On the Inductive Biases of Deep Generative Models

Fabian Falck

Oriel College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas 2024

To my parents

Acknowledgements

Studying towards my DPhil at Oxford has been an incredible learning and life experience. I am deeply grateful to those who enabled this, and who made it such a fantastic time.

First and foremost, I would like to thank my supervisors Chris Holmes and Arnaud Doucet for their guidance throughout this journey. I am thankful for all their expertise and deep interest for machine learning and statistics they shared with me. They always believed in me, encouraged me to pursue my ideas, gave me the freedom to be curious, and made me eager and excited to produce the best research I can. I learned so much from them, both technically but also in what it means to be a good researcher. Thank you for all your support and advice!

I would like to thank Matthew Willetts and Saifuddin Syed for being exceptional mentors during my DPhil. Matthew's work on probabilistic generative models inspired my research interest early on. He spent countless hours with me discussing research, both high- and low-level, and shared with me his wisdom of the field. I am also thankful to Saif for the numerous in-depth research discussions we had together, and for the brilliant research advice he gave me.

Thank you to all my collaborators, both those contributing to the papers constituting this thesis and beyond. It has been such a pleasure working with all of them. In particular, I would like to thank my co-first authors Haoting Zhang, Christopher Williams, and Ziyu Wang: the papers I present in this thesis would not have been possible without them. I learned so much from them, and thank them for all the time we spent and fought together. I would also like to thank George Nicholson for all his positivity, encouragement and diligence throughout the years working with him.

I would like to thank the senior tutors at Oriel college, Mike Spivey and Irina Voiculescu, for their confidence in me. Teaching computer science undergraduate students through my role as a Graduate Teaching and Research Scholar at the college was a very enriching experience. I learned a lot from the interactions with all the brilliant Oriel students I taught during two and a half years.

I would like to thank the Wellcome Trust and the Health Data Research UK-The Alan Turing Institute Wellcome PhD programme for funding and supporting my PhD studies, and the directors of the programme, Christopher Yau, Peter Diggle, Ioanna Manolopoulou and Max Little, for their support. I would also like to thank and acknowledge partial funding through the Enrichment Scheme of The Alan Turing Institute. Thank you to the Alan Turing's Research Engineering team, the Biomedical Research Computing Team at the University of Oxford and the Baskerville Advanced Research Computing team at University of Birmingham for providing excellent computational resources and support throughout my DPhil.

Thank you to everyone in the OxCSML group and the Department of Statistics: Oscar Clivio, Jin Xu, Guneet Singh Dhillon, Amitis Shidani, Tim Reichelt, Brieuc Lehmann, Sam Dauncey, Linying Yang, Vik Shirvaikar, Andrew Campbell, Angus Phillips, Michael Hutchinson, Muhammad Faaiz Taufiq, Desi Ivanova, Sahra Ghalebikesabi, Anna Menacher, Cong Lu, Ning Miao, Tyler Farghly, Bobby He, Emilien Dupont, James Thornton, senior members George Deligiannidis and Habib Ganjgahi, and all those who I missed. Beyond Oxford, thank you to Vincent Jeanselme, Dominic Danks, and my collaborators in the Oxford-Novartis collaboration, particularly Xuan Zhu, Matthias Kormaksson, Marc Vandemeulebroecke and Tom Nichols. They all made this DPhil such a pleasurable time on a daily basis. I would also like to thank John McManigle for open-sourcing his LaTeX template which I used for this thesis.

Thank you to Jan Gasthaus, Richard Kurle, Abdul Fatir Ansari, Niranjani Prasad and Ted Meeds, and all my other colleagues at Amazon Science in Berlin and Microsoft Research Cambridge where I had the pleasure of interning during two wonderful summers.

Thank you to Angela for her support, especially before conference deadlines and during challenging times, which was invaluable to me on this journey.

Lastly, I would like to thank my parents. I am here because of them. My mother and father gave me the foundation, and an approach to life of being eager to learn and do your best which got me where I am today. Or to put it in my father's words: "Wer viel weiß hat mehr vom Leben."

Abstract

Generative modelling is an important machine learning paradigm for approximating high-dimensional distributions of multi-modal data. Generative probabilistic models learn a compressed latent representation and allow to sample from the approximated distribution, achieving state-of-the-art performance across numerous tasks. Many generative models feature latent variables which allow them to learn a hierarchical representation capturing structure in data or exploit its multi-resolution property. Others can be flexibly conditioned on available data at inference time. However, the inductive biases why generative models work well, and their critical components such as their neural architecture are often poorly understood.

In this thesis, we develop, characterise and probe the inductive biases of four major generative model classes: Variational Autoencoders (VAEs), Hierarchical VAEs (HVAEs), diffusion models, and autoregressive models or Large Language Models (LLMs). First, we propose Multi-Facet Clustering Variational Autoencoders (MFCVAE), a novel generative model which can uncover a multi-partition structure in data. Second, we identify state-of-the-art HVAEs as discretisations of an underlying multi-resolution diffusion process. Third, we analyse why U-Nets are a go-to architecture for the generative modelling of images and for diffusion models in particular, characterising them as learning the coefficients on truncated, finite-dimensional function spaces via preconditioning. Fourth, for LLMs, we analyse the hypothesis that in-context learning with i.i.d. data follows Bayesian principles. The goal of this thesis is to understand fundamental design principles and properties of deep generative models, which may characterise existing and inspire novel inductive biases.

Contents

List of Figures

List of Tables

List of Abbreviations

AI . . . . . Artificial Intelligence

BIVA . . . . . Bidirectional-Inference Variational Autoencoder

CI . . . . . Confidence Interval

c.i.d. . . . . . Conditionally Identically Distributed

CNN . . . . . Convolutional Neural Network

DDIM . . . . . Denoising Diffusion Implicit Models

DDPM . . . . . Denoising Diffusion Probabilistic Models

DWT . . . . . Discrete Wavelet Transform

ELBO . . . . . Evidence Lower Bound

EM . . . . . Expectation-Maximization

FID . . . . . Fréchet inception distance

FNO . . . . . Fourier Neural Operator

GAN . . . . . Generative Adversarial Network

GMM . . . . . Gaussian Mixture Model

GPT . . . . . Generative Pre-trained Transformer

GPU . . . . . Graphics Processing Unit

HMM . . . . . Hidden Markov Model

HVAE . . . . . Hierarchical Variational Autoencoder

ICL . . . . . In-Context Learning

i.i.d. . . . . . Independent and Identically Distributed

IQR . . . . . Inter-Quartile Range

JPEG . . . . . Joint Photographic Experts Group

KL . . . . . Kullback–Leibler divergence

LLM . . . . . Large Language Model

LPIPS . . . . . Learned Perceptual Image Patch Similarity
LSTM . . . . . Long Short-Term Memory
LTVAE . . . . . Latent Tree Variational Autoencoder
LVAE . . . . . Ladder Variational Autoencoder
MC . . . . . Monte Carlo
MFCVAE . . . . . Multi-Facet Clustering Variational Autoencoder
MLE . . . . . Maximum Likelihood Estimation
MoG . . . . . Mixture-of-Gaussians
MRI . . . . . Magnetic Resonance Imaging
Multi-ResNet . . . . . Multi-Resolution Residual Network
NaN . . . . . Not a Number
NLL . . . . . Negative Log-Likelihood
NLP . . . . . Natural Language Processing
NVAE . . . . . Nouveau VAE
ODE . . . . . Ordinary Differential Equation
OOD . . . . . Out-of-Distribution
PDE . . . . . Partial Differential Equation
RCT . . . . . Randomised Control Trial
ReLU . . . . . Rectified Linear Unit
ResNet . . . . . Residual Network
RGB . . . . . Red, Green, Blue
r-MSE . . . . . Rollout Mean-Squared Error
SAE . . . . . Stacked Autoencoder
SDE . . . . . Stochastic Differential Equation
SNR . . . . . Signal-to-Noise Ratio
VaDE . . . . . Variational Deep Embedding
VAE . . . . . Variational Autoencoder
VDM . . . . . Variational Diffusion Models
VDVAE . . . . . Very Deep Variational Autoencoder
VLAE . . . . . Variational Ladder Autoencoder
WS-VDVAE . . . . . Weight-Sharing Very Deep Variational Autoencoder

1

Introduction

Contents

All information, matter and life in the universe have been generated following stochastic processes and probabilistic decisions. Whether it is the formation and folding of proteins, the interaction of cells, how pixels are composed in an image, or how text is written by humans, all these have in common that their physical and cognitive mechanisms follow a generative process. The modelling of such processes with machine learning is the topic of this thesis. Building and characterising *generative models*, which aim to capture the generative process of observables, may not only let us mimic the generative process of nature and human-made artifacts, it may also allow us to understand their inherent structure, intervene on them, and compress the information for subsequent use.

Generative models have emerged as one of the most successful and capable Artificial Intelligence (AI) technologies of present time. They are showing large potential as general-purpose systems across numerous applications. Systems such as Chat Generative Pre-trained Transformer (ChatGPT) [176] can predict text given a prompt in almost all areas of human knowledge available on the internet and beyond. In the sciences, generative models can discover new proteins or materials

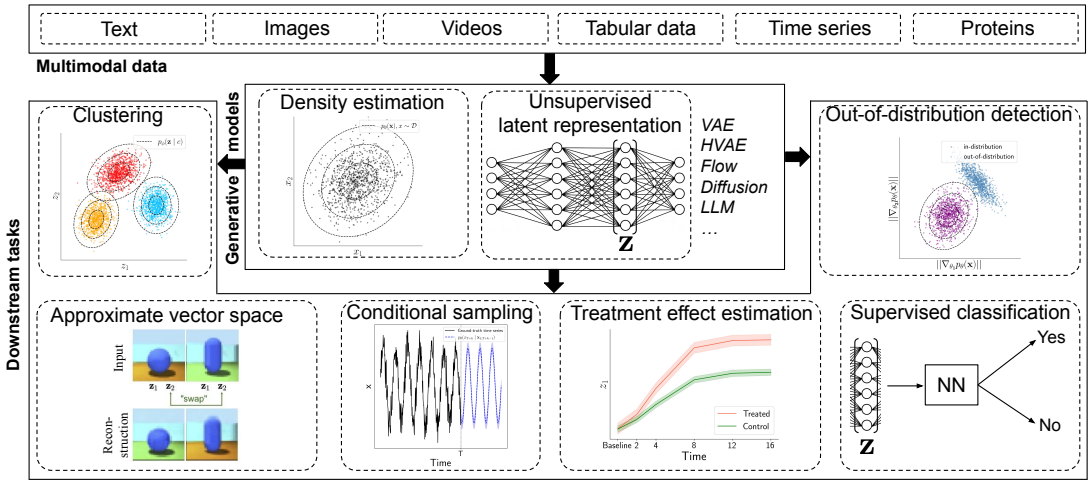


Figure 1.1: *Generative models*: Learning unsupervised representations via density estimation for downstream tasks. Illustrated are several examples of data modalities and downstream tasks, some of which are a topic of this thesis.

[112, 161, 234, 252]. In healthcare, multi-task systems learn powerful representations from a range of data sources and can be tailored to specific applications, such as radiology report generation [13, 198]. Remarkably, generative models are perhaps the first AI technology with global impact and use, while remaining a frontier of machine learning research.

Generative models can be characterised by two properties (see Figure 1.1): First, they learn a compressed, unsupervised, and sometimes structured and interpretable *latent representation*. Second, they perform (conditional) *density estimation* of the data distribution, which enables the generation or sampling of new examples. These trained, large-scale generative models (sometimes referred to as foundation models [21]) and their representations are subsequently useful in a myriad of downstream tasks, such as clustering [67], out-of-distribution (OOD) detection [48] and conditional generation, which all fundamentally depend on the learned distributions and obtained latent representations.

The remarkable success of generative models lies in their ability to represent the structure and patterns in a dataset in an unsupervised way: they are capable of learning these patterns without a supervision or reward signal, merely by predicting

the data itself. In later training stages, they may be fine-tuned with supervised or reinforcement learning to improve or guide the learned representation [11, 184, 264]. Importantly, certain generative models such as Large Language Models (LLMs) [176, 189] and diffusion models [196] are capable of processing vast amounts of high-dimensional data, such as text, images or scientific information, allowing them to scale with data in both size and corresponding performance [94]. Their scaling behaviour is especially enabled by the transformer architecture [226] and the U-Net [197], two core neural network building blocks of these generative models. This renders them a powerful learning paradigm in a digitised world where data from multiple modalities and sources becomes abundant.

However, in spite of the enormous potential and success of generative models, we lack understanding *why* generative models work so well. Even though latent variable models allow us to impose structure into a model, this structure is often very simple and serves the generative performance of the model rather than its purpose to explore and understand the data at hand. Multi-Facet Clustering Variational Autoencoders, which we will introduce in Chapter 2, close this gap for multi-partition data. Furthermore, even though many generative models are perceived as different classes and hence fundamentally different, we often fail to see their similarities and relatedness. In Chapter 3, we shed light on hierarchical VAEs (HVAEs), highlighting their connections with diffusion models. Moreover, the neural network used within generative models is a key component of their inductive bias that is highly tuned experimentally, yet often lacks theoretical grounding. In Chapter 4, we characterise U-Nets, an important neural architecture in diffusion models, as preconditioned models which learn the coefficients on truncated, finite-dimensional function spaces (e.g. Haar wavelet spaces). Lastly, autoregressive models and LLMs can be flexibly conditioned on a dataset at inference time via so-called in-context learning. While previous work claims that this conditioning is following Bayesian principles, in Chapter 5 we contradict this hypothesis. We

refer to the respective introduction sections in Chapters 2 to 5 for an extended motivation and overview of the presented work.

1.1 Background

1.1.1 What is a Generative Model?

We begin by defining generative models in machine learning. A *generative model* is a joint probability distribution $p_\theta(\boldsymbol{x}|\boldsymbol{c})$, where $\boldsymbol{x} \in \mathcal{X}$ is an observed variable (e.g. an image, or the tokens of a piece of text), $\boldsymbol{c} \in C$ is a conditioning variable (e.g. a prompt in chat systems), and θ are learnable parameters [167, p.765]. We call $p_\theta(\boldsymbol{x}|\boldsymbol{c})$ an unconditional generative model if $\boldsymbol{c} = \emptyset$, and a conditional generative model otherwise. Chapters 2 to 4 mostly focuses on unconditional generative modelling, while Chapter 5 analyses conditioning in LLMs. There are two distinguishing points of generative machine learning models compared to discriminative models (such as those solving a regression or classification task): First, the variables $\boldsymbol{x}$ (and optionally $\boldsymbol{c}$) are typically high-dimensional. Second, in generative models we assume there could be multiple outputs conditional on $\boldsymbol{c}$ which a generative model correctly assigns a high probability to, while in discriminative models there typically exists exactly one ground-truth output or label, which may be a set [167, p. 768]. Even though these two points and the boundaries they draw are not strictly defined, the usage of the terms generative versus discriminative models is typically clear from the task context.

There are different classes of generative models which can be characterised by several properties (closely following [167, p.765]), which all contribute towards their unique inductive bias:

- Discrete or continuous: generative models may define a discrete or continuous distribution $p_\theta(\boldsymbol{x}|\boldsymbol{c})$. This typically depends on whether its training data is

discrete (e.g. text) or continuous (e.g. images). In this thesis we focus on human artifacts like images and text which are abundantly available as datasets. We note that in the context of diffusion models, discrete and continuous may also refer to the time variable [209], which is to be distinguished from the use here.

- Density: most generative models allow the point-wise evaluation of the probability density $p_\theta(\boldsymbol{x}|\boldsymbol{c})$. This evaluation can either be exact or approximate (e.g. the Evidence Lower-Bound (ELBO) in VAEs). However, models such as Generative Adversarial Networks (GANs) [74] do not allow for point-wise evaluation and rather learn this density implicitly.

- Training: the evaluation of the density of a generative model is crucial for their training, i.e. fitting the parameters θ. Most methods perform Maximum Likelihood Estimation (MLE), while some might only compute an approximation thereof as they approximate $p_\theta(\boldsymbol{x}|\boldsymbol{c})$. Other deviations from strict MLE include weighting certain terms in the training loss [67], or the addition of a regularisation term.

- Sampling: by definition, generative models are able to generate samples $\boldsymbol{x} \sim p_\theta(\boldsymbol{x}|\boldsymbol{c})$ from its learned density. The computational cost and time to sample can vary a lot, and depends on several factors, such as the number of latent variables, the size of the neural network applied, and the amount of computation that can be parallelised. On the latter point, autoregressive models sample each dimension in the data $\boldsymbol{x}$ sequentially, while diffusion models require the sequential sampling of a chain of latent variables before a data sample can be obtained. This renders autoregressive and diffusion models to be in general slower samplers than other classes of generative models like VAEs, at the potential benefit of higher sampling quality.

- Latent variables: many popular generative models (including VAEs, HVAEs and diffusion models) are so-called latent variable models: they introduce

parameterised conditional distributions over latent variables $\boldsymbol{z}$ which are used to evaluate the density $p_\theta(\boldsymbol{x}|\boldsymbol{c})$. In some cases (e.g. in VAEs), the latent variables serve the purpose of obtaining a compressed representation.

- Neural architecture: the neural network architecture is a crucial building block in generative models. Together with their training, it is the key component differentiating modern generative machine learning models from classical, statistical (Bayesian) generative models [20]. The neural architecture captures numerous advances of deep learning, which are crucial for the success of generative models. Neural architectures are parameterised, deterministic and learnable functions. In Chapters 3 and 4, we will investigate and rigorously define the U-Net, a neural architecture particularly successful for modelling images, videos and PDEs.

In this thesis, we focus on four classes of generative models: VAEs [121], HVAEs [34, 221], the hierarchical extension of VAEs, diffusion models [92, 206] and autoregressive models (or LLMs) [189]. We introduce the core components of each model class below in their unconditional form, referring to the cited seminal papers and a reference book [167], which we closely follow, and the respective thesis chapters and appendices for further details. In particular, Sections 2.5, 3.4, C.3.1, and D.4 review the work related to the main thesis chapters and the Appendices provide additional background material.

1.1.2 Variational Autoencoders

A *Variational Autoencoder (VAE)* [121, 167] is a latent variable model consisting of three components: a likelihood $p_\theta(\mathbf{x}|\vec{\mathbf{z}})$, a prior $p_\theta(\vec{\mathbf{z}})$ and an approximate posterior $q_\phi(\vec{\mathbf{z}}|\mathbf{x})$, where $\mathbf{x}$ is observed and $\vec{\mathbf{z}}$ are the latent variables. In Chapter 2, $\vec{\mathbf{z}}$ are a set of continuous and discrete latent variables (and we will use distinct variables for both), while most often VAEs consist of continuous latent variables $\vec{\mathbf{z}}$ only. A plain VAE has

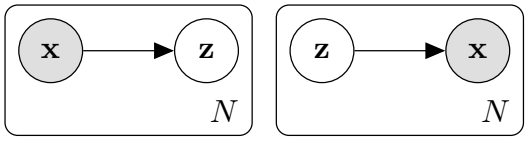


Figure 1.2: Graphical model of a plain VAE. [Left] Amortised variational posterior $q_\phi(\vec{\mathbf{z}}|\mathbf{x})$. [Right] Generative model $p_\theta(\mathbf{x}, \vec{\mathbf{z}}) = p_\theta(\vec{\mathbf{z}}|\mathbf{x})p_\theta(\vec{\mathbf{z}})$.

a single, high-dimensional latent variable $\vec{\mathbf{z}} = \mathbf{z}$, yet for consistency with hierarchical VAEs (see §1.1.3) we will use the notation $\vec{\mathbf{z}}$. We illustrate a plain VAE in Fig. 1.2.

The likelihood and prior form the generative model $p_\theta(\mathbf{x}, \vec{\mathbf{z}})$ as

$$p_\theta(\mathbf{x}, \vec{\mathbf{z}}) = p_\theta(\vec{\mathbf{z}}|\mathbf{x})p_\theta(\vec{\mathbf{z}}). \tag{1.1}$$

In plain VAEs, the prior $p_\theta(\vec{\mathbf{z}})$ is typically Gaussian with a diagonal covariance structure, optionally with learnable parameters. The likelihood $p_\theta(\mathbf{x}|\vec{\mathbf{z}})$ is chosen in a data-dependent way, for instance as (isotropic) Gaussian or Bernoulli distributed, and is typically assumed to be independent over the dimensions of $\mathbf{x}$. Furthermore, the likelihood $p_\theta(\mathbf{x}|\vec{\mathbf{z}})$ is parameterised by a neural network f_θ. In plain VAEs, f_θ receives a sample $\vec{\mathbf{z}}$ as input, and outputs some or all of the parameters of the distribution $p_\theta(\mathbf{x}|\vec{\mathbf{z}})$ (e.g. the mean, if the likelihood is isotropic Gaussian with a variance defined by a hyperparameter).

To generate samples, we only require the prior and the likelihood, drawing samples from them sequentially as $\mathbf{z} \sim p_\theta(\mathbf{z}), \mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$. However, to train a VAE, we need to be able to infer the latent variables $\vec{\mathbf{z}}$. Unfortunately, the posterior distribution $p(\vec{\mathbf{z}}|\mathbf{x})$ is computationally intractable. We therefore resort to parameterising the posterior distribution with a neural network g_ϕ with parameters ϕ, a concept called amortized variational inference [50]. We obtain an approximate posterior distribution $q_\phi(\vec{\mathbf{z}}|\mathbf{x}) \approx p(\vec{\mathbf{z}}|\mathbf{x})$.

To train a VAE (and an HVAE), one maximises an ELBO with respect to parameters ϕ and θ via stochastic gradient descent using the reparametrization trick [121],

$$\log p(\mathcal{D}) \geq \mathcal{L}(\mathcal{D}; \phi, \theta) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\underbrace{\left[\mathbb{E}_{\vec{\mathbf{z}}\sim q_\phi(\vec{\mathbf{z}}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\vec{\mathbf{z}})\right]\right]}_{\text{Reconstruction loss}} - \underbrace{\mathrm{KL}[q_\phi(\vec{\mathbf{z}}|\mathbf{x})||p_\theta(\vec{\mathbf{z}})]}_{\text{Prior loss}}, \tag{1.2}$$

where $\mathcal{D}$ denotes the training dataset. The ELBO is a lower bound approximation of the log data marginal likelihood $\log p(\mathbf{x})$.

1.1.3 Hierarchical Variational Autoencoders

A *hierarchical Variational Autoencoder (HVAE)* [1] is a VAE where latent variables are separated into L groups $\vec{\mathbf{z}} = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_L)$ which conditionally depend on each other. L is sometimes referred to as stochastic depth. For convenience of notation, we write the observed variable $\mathbf{x}$ as $\mathbf{z}_0$, i.e. $\mathbf{x} \equiv \mathbf{z}_0$. In HVAEs, latent variables typically follow a 'bow tie', U-Net [197] type architecture with an information bottleneck [217] such that $\dim(\mathbf{z}_{l+1}) \leq \dim(\mathbf{z}_l)$ for all $l = 0, \ldots, L-1$. Latent variables 'live' on multiple resolutions, either decreasing steadily in dimension [151, 207], or step-wise every few stochastic layers [34, 221]. We consider this multi-resolution property an important characteristic of HVAEs. It distinguishes HVAEs from other deep generative models, in particular vanilla diffusion models where latent and data variables are of equal dimension [92].

To instantiate $q_\phi(\vec{\mathbf{z}}|\mathbf{x})$ and $p_\theta(\vec{\mathbf{z}})$, numerous conditional structures of the latent variables exist in HVAEs, and we review them in §3.4. In Chapter 3, we follow [34, 119, 221]: the latent variables in the prior and approximate posterior are estimated in the same order, from $\mathbf{z}_L$ to $\mathbf{z}_1$, conditioning 'on all previous latent variables', i.e.

$$p_\theta(\vec{\mathbf{z}}) = p_\theta(\mathbf{z}_L) \prod_{l=1}^{L-1} p_\theta(\mathbf{z}_l|\mathbf{z}_{>l}), \qquad (1.3) \qquad q_\phi(\vec{\mathbf{z}}|\mathbf{x}) = q_\phi(\mathbf{z}_L|\mathbf{x}) \prod_{l=1}^{L-1} q_\phi(\mathbf{z}_l|\mathbf{z}_{>l}, \mathbf{x}). \qquad (1.4)$$

We visualise the graphical model of this HVAE in Fig. 1.3. Recent HVAEs [34, 221] capture this dependence on all previous latent variables $\mathbf{z}_{>l}$ in their residual state as shown in §3.2.3, which imposes the conditional structure. This

[1] We closely follow the introduction of hierarchical VAEs in [34, §2.2].

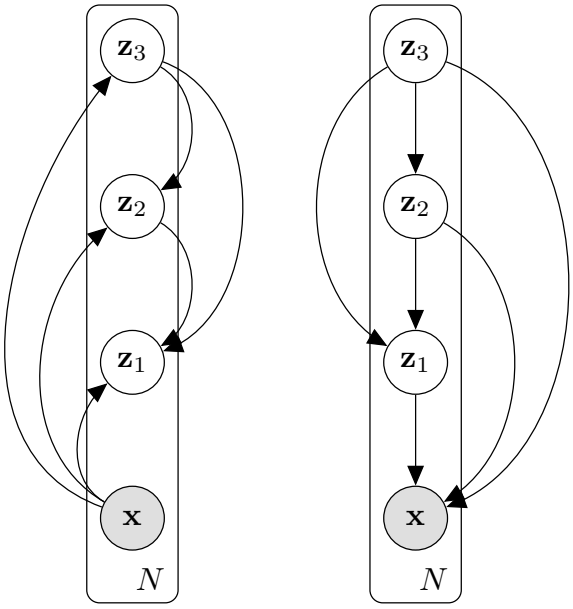


Figure 1.3: Conditional structure in state-of-the-art HVAE models (VDVAE [34] / NVAE [221]) with $L = 3$. [Left] Amortised variational posterior $q_\phi(\vec{\mathbf{z}} \mid \mathbf{x})$. [Right] Generative model $p_\theta(\mathbf{x}, \vec{\mathbf{z}})$.

implies a first-order Markov chain conditional on the previous residual state. First-order Markov processes have shown great success empirically, such as in LSTMs [93]. Further, note that in all previous work on HVAEs, the neural networks corresponding to the l-th stochastic layer which estimate the inference and generative distributions, respectively, do not share parameters with those estimating other stochastic layers $l' \neq l$.

Intuitively, the conditional structure of HVAEs together with a U-Net architecture imposes an inductive bias on the model to learn a hierarchy of latent variables where each level corresponds to a different level of abstraction. In Chapter 3, we characterise this intuition via the regularisation property of U-Nets in §3.2.2.

The distributions over the latent variables in both the inference and generative model are Gaussian with mean $\boldsymbol{\mu}$ and a diagonal covariance matrix $\boldsymbol{\Sigma}$, i.e. for all $l = 1, \ldots, L$,

$$q_\phi(\mathbf{z}_l|\mathbf{z}_{>l}, \mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}_{l,\phi}, \boldsymbol{\Sigma}_{l,\phi}), \tag{1.5}$$

$$p_\theta(\mathbf{z}_l|\mathbf{z}_{>l}) \sim \mathcal{N}(\boldsymbol{\mu}_{l,\theta}, \boldsymbol{\Sigma}_{l,\theta}), \tag{1.6}$$

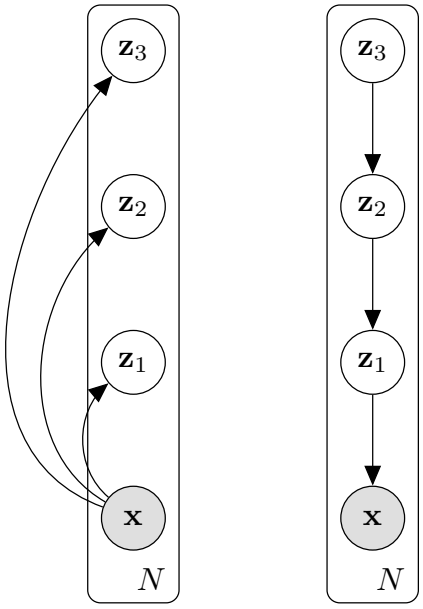


Figure 1.4: Graphical model of a diffusion model (DDPM [92]) with $L = 3$ timesteps. [Left] Amortised variational posterior $q_\phi(\vec{\mathbf{z}} \mid \mathbf{x})$. [Right] Generative model $p_\theta(\mathbf{x}, \vec{\mathbf{z}})$.

where mean and variances are estimated by neural networks with parameters ϕ and θ corresponding to stochastic layer l. Note that $p_\theta(\mathbf{z}_L|\mathbf{z}_{>l}) = p_\theta(\mathbf{z}_L)$, where the top-down block estimating $p_\theta(\mathbf{z}_L)$ receives the zero-vector as input, $q_\phi(\mathbf{z}_L|\mathbf{z}_{>L}, \mathbf{x}) = q_\phi(\mathbf{z}_L|\mathbf{x})$, meaning that we infer without conditioning on other latent groups at the L-th step. Further, for modelling images, VDVAE chooses $p_\theta(\mathbf{x}|\vec{\mathbf{z}})$ to be a mixture of discretised logistics likelihood.

1.1.4 Diffusion Models

A *diffusion model* is the third class of generative models discussed in this thesis, which is possibly the state-of-the-art generative model for images, proteins and other modalities. Diffusion models were first proposed in the seminal paper by Sohl-Dickstein et al. [206], and were scaled up to become a popular approach through the work of Ho et al. [92] (and others) whose formulation we closely follow. Just like in HVAEs, diffusion models introduce a hierarchy of latent variables $\vec{\mathbf{z}}$ which conditionally depend on each other and is learned through a variational lower-bound of the negative log-likelihood $-\log p_\theta(\boldsymbol{x})$ of the observations.

In the context of diffusion models, we differentiate two types of inference: the forward (or approximate posterior) $q(\vec{\boldsymbol{z}}|\boldsymbol{x})$ and the backward (or generative) process $p(\vec{\boldsymbol{z}}, \boldsymbol{x})$ (see Fig. 1.4 for an illustration of the graphical model). On a high-level, the idea of diffusion models is to induce a deterministic, highly structured forward process which adds noise to the data until all signal is fully corrupted, and learn its reverse process parameterised by a neural network.

The forward process is governed by a Markov chain. We can write the transitions of the forward process as

$$q(\boldsymbol{z}_l|\boldsymbol{z}_{l-1}) = \mathcal{N}(\sqrt{1-\beta_l}\boldsymbol{z}_{l-1}, \beta_l \boldsymbol{I}), \tag{1.7}$$

where the parameters β_l govern the forward noising process, and we again use the notation $\boldsymbol{z}_0 \equiv \boldsymbol{x}$. These parameters are normally[2] fixed, such that the forward process contains no learnable parameters. We can compute the distribution $q(\boldsymbol{z}_l|\boldsymbol{x})$ in closed form as

$$q(\boldsymbol{z}_l|\boldsymbol{x}) = \mathcal{N}(\sqrt{\overline{\alpha}_l}\boldsymbol{x}, (1-\overline{\alpha}_l)\boldsymbol{I}), \tag{1.8}$$

where $\alpha_l = 1-\beta_l$ and $\overline{\alpha}_l = \prod_{s=1}^{l} \alpha_s$. Eq. (1.8) allows for fast inference in the forward process. We can further compute the posterior distribution of the forward process as

$$q(\boldsymbol{z}_{l-1}|\boldsymbol{z}_l, \boldsymbol{z}_0) = \mathcal{N}(\tilde{\boldsymbol{\mu}}_l(\boldsymbol{z}_l, \boldsymbol{z}_0), \tilde{\beta}_l(\overline{\alpha}_{l-1}, \overline{\alpha}_l, \beta_l)\boldsymbol{I}) \tag{1.9}$$

where $\tilde{\boldsymbol{\mu}}$ and $\tilde{\beta}_l$ are functions of $\boldsymbol{z}_l$ and $\boldsymbol{z}_0$, and $\overline{\alpha}_{l-1}$, $\overline{\alpha}_l$ and β_l, respectively. Both can be computed in closed form (see [92] Appendix A for details).

The backward process is defined as

$$p_\theta(\boldsymbol{z}_{l-1}|\boldsymbol{z}_l) = \mathcal{N}(\mu_\theta(\boldsymbol{z}_l, l), \boldsymbol{\Sigma}_\theta(\boldsymbol{z}_l, l)), \tag{1.10}$$

where $\boldsymbol{\mu}_\theta$ is a neural network with parameters θ, typically a U-Net for images, which receives $\boldsymbol{z}_l$ and the current time step l as input. With this forward and

[2] See [120] for a counterexample.

backward process, we can maximise the ELBO

$$\log p(\mathcal{D}) \geq \mathcal{L}(\mathcal{D};\theta) \tag{1.11}$$

$$= \mathbb{E}_{x\sim\mathcal{D}}\Big[-\mathrm{KL}[q(\boldsymbol{z}_L|\boldsymbol{x})||p(\boldsymbol{z}_L)] - \sum_{l>1}\underbrace{\mathrm{KL}[q(\boldsymbol{z}_{l-1}|\boldsymbol{z}_l,\boldsymbol{x})||p_\theta(\boldsymbol{z}_{l-1}|\boldsymbol{z}_l)]}_{\mathcal{L}_{l-1}} + \log p_\theta(\boldsymbol{x}|\boldsymbol{z}_1)\Big]. \tag{1.12}$$

One can further make the simplifying assumption that $\boldsymbol{\Sigma}_\theta(\boldsymbol{z}_l, l) = \sigma_l^2\boldsymbol{I}$, where $\sigma_l^2 = \frac{1-\overline{\alpha}_{l-1}}{1-\overline{\alpha}_l}\beta_l$. With this choice, the term $\mathcal{L}_{l-1}$ simplifies to

$$\mathcal{L}_{l-1} = \mathbb{E}_{q(z_0,z_l)}\left[\frac{1}{2\sigma_l^2}\|\tilde{\boldsymbol{\mu}}_l(\boldsymbol{z}_l,\boldsymbol{z}_0) - \boldsymbol{\mu}_\theta(\boldsymbol{z}_l,l)\|^2\right] + C, \tag{1.13}$$

where C is a constant independent of the parameters θ. We can hence interpret the loss—in this simplifying form—as a regression of the mean of the variational posterior distribution in the forward process.

Given a trained diffusion model, i.e. having learned the parameters θ with a variant of stochastic gradient descent, we can now draw samples from this model. Most sampling algorithms correspond to discretisations of viewing the reverse process as a Stochastic Differential Equation (SDE) or an Ordinary Differential Equation (ODE) (see [209] for details on the continuous-time formulation of diffusion models). One particularly popular ODE discretisation is proposed in DDIM [208].

Diffusion models share many parallels with HVAEs. They are hierarchical latent variable models with a very similar structure and loss function [117]. They use a U-Net architecture as their backbone [197]. They both discretise an SDE or ODE in the backward process (see Chapter 3 for details on HVAEs). However, they have several notable differences [150]: First, the latent dimension $\dim(\boldsymbol{z}_l)$ in diffusion models is the same as the data dimension in all latent variables. In contrast, in HVAEs, this is not the case: there are typically several resolutions, each corresponding to a different latent dimension. Second, the encoder or forward process in diffusion models is fixed, interpolating between the data distribution and

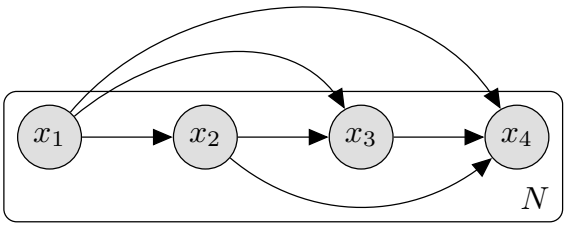


Figure 1.5: Graphical model of an autoregressive model with data dimension $D = 4$.

typically a standard Gaussian distribution, and can be computed in closed form. This allows to train diffusion models by sampling timesteps l (uniformly at random) and computing the corresponding loss terms at these timesteps, which is not feasible in HVAEs. HVAEs learn the forward process, and there is no imposed structure as in the fixed forward process of diffusion models, rendering HVAEs the more flexible model class in this regard. Third, diffusion models use weight-sharing across all timesteps l. HVAEs, however, typically use different neural networks across the depth dimension l to parameterise the variational posterior and generative distribution. Chapter 3 will explore a specific form of weight-sharing in HVAEs.

1.1.5 Autoregressive models

An *autoregressive model* [167, p.811] learns to model the data by factorising its joint distribution $p(\boldsymbol{x})$ as

$$p_\theta(\boldsymbol{x}) = \prod_{i=1}^{D} p_\theta(x_i|\boldsymbol{x}_{1:i-1}), \tag{1.14}$$

where $x_i \in \mathcal{X}$ indicates the i-th observation, and $\boldsymbol{x}_{1:i-1}$ indicates x_1 to x_{i-1}. As before, we may alternatively condition on an input $\boldsymbol{c}$ modelling $p_\theta(\boldsymbol{x}|\boldsymbol{c})$. In the context of LLMs, we refer to x_i as a token and the optional input $\boldsymbol{c}$ as the prompt. An autoregressive model hence learns the conditional distributions $p_\theta(x_i|\boldsymbol{x}_{1:i-1}, \boldsymbol{c})$ with amortized weights θ, which become more complex as i increases since the model is conditioned on more observations. We illustrate the graphical model of an autoregressive model in Fig. 1.5.

Perhaps the most popular instantiation of autoregressive models are LLMs [189] such as the GPT-4 [1] or the Llama [218] model families. In these state-of-the-art autoregressive models, the conditional distributions $p_\theta(x_i|\boldsymbol{x}_{1:i-1}, \boldsymbol{c})$ follow a categorical distribution. Furthermore, the transformer [226] neural architecture plays a crucial role in achieving the remarkable performance of modern LLMs. Earlier autoregressive models include WaveNet [222], mostly for modelling time series, and PixelCNN [223].

Autoregressive models are typically (in the first stage) trained with MLE using a cross-entropy loss. It is worth noting that just like in diffusion models, autoregressive models share their weights θ across all steps i. Furthermore, as the number of conditional distributions in Eq. (1.14) scales linearly with the data dimension D, autoregressive models can be slow during inference. This is a potential advantage of latent variable models with L latents as they often achieve excellent performance with $L \ll D$.

1.2 Papers

1.2.1 Papers constituting this thesis

This thesis is formed of four papers, presented in Chapters 2 to 5 and Appendices A to D. Their content is—subject to few rearrangements within the thesis—presented without significant modifications, but has been reformatted and edited where adequate. Each paper is self-contained, providing a separate introduction, background material, related work and notation. I list these papers below. The symbol * indicates equal contribution.

Multi-facet clustering variational autoencoders. **Fabian Falck***, Haoting Zhang*, Matthew Willetts, George Nicholson, Christopher Yau, Chris Holmes. Advances in Neural Information Processing Systems (NeurIPS) 2021 [67].

A Multi-Resolution Framework for U-Nets with Applications to Hierarchical VAEs. **Fabian Falck***, Christopher Williams*, Dominic Danks, George Deligiannidis, Christopher Yau, Chris Holmes, Arnaud Doucet, Matthew Willetts. Advances in Neural Information Processing Systems (NeurIPS) 2022 (oral) [66].

A Unified Framework for U-Net Design and Analysis. Christopher Williams*, **Fabian Falck***, George Deligiannidis, Chris Holmes, Arnaud Doucet, Saifuddin Syed. Advances in Neural Information Processing Systems (NeurIPS) 2023 [239].

Is In-Context Learning in Large Language Models Bayesian? A Martingale Perspective. **Fabian Falck***, Ziyu Wang*, Chris Holmes. International Conference on Machine Learning (ICML) 2024. Also presented at Secure and Trustworthy Large Language Models workshop at ICLR 2024 (oral) [65].

In the following, I detail my contributions and the contributions of my co-authors for each of paper, augmenting the Statements of Authorship presented at the end of each chapter.

'Multi-facet clustering variational autoencoders' (Chapter 2 and Appendix A): Haoting Zhang and I jointly led the project. I had the initial idea for the project, which originated from a literature review and discussions with Haoting Zhang. Haoting Zhang and I jointly developed the method, with crucial contributions and ideas from Matthew Willetts. George Nicholson also contributed to aspects of and discussions on the method. Haoting Zhang led the theoretical derivations with my support, and contributions from George Nicholson and Christopher Yau. I implemented large parts of the code base, and demonstrated proof-of-concept experiments on a first dataset. Haoting Zhang made major contributions to the code base. Haoting Zhang and I jointly conducted all experiments, and wrote the paper. Matthew Willetts and all other co-authors edited the paper. Matthew Willetts and Chris Holmes advised the project and provided crucial methodological guidance.

'A Multi-Resolution Framework for U-Nets with Applications to Hierarchical VAEs' (Chapter 3 and Appendix B): I initiated the project and led its early stages. I developed the initial project ideas based on a literature review and discussions with Dominic Danks, Matthew Willetts and Chris Holmes. Christopher Williams led the development of the theoretical framework with my support, and contributions from Arnaud Doucet and the other co-authors. I led the development of the code base with contributions from Dominic Danks, and performed all experiments. In particular, I demonstrated proof-of-concept experiments for the effectiveness of weight-sharing in HVAEs. Christopher Williams and I jointly wrote the paper, Arnaud Doucet and all other co-authors edited it. Matthew Willetts, Arnaud Doucet, Chris Holmes, George Deligiannidis and Christopher Yau advised the project and provided crucial methodological guidance.

'A Unified Framework for U-Net Design and Analysis' (Chapter 4 and Appendix C): I initiated the project. Christopher Williams and I jointly led the project. Christopher Williams and I jointly developed the initial project ideas and motivation around the Multi-ResNet. Saifuddin Syed, Christopher Williams and I then jointly developed the methodological ideas in the paper. Christopher Williams led the derivation of theoretical results with the support of Saifuddin Syed and me. I implemented the large majority of the code base, and performed all experiments except one. Christopher Williams, and Saifuddin Syed and I jointly wrote the paper, Arnaud Doucet and all other co-authors edited it. Saifuddin Syed, Arnaud Doucet, George Deligiannidis and Chris Holmes advised the project and provided crucial methodological guidance.

'Is In-Context Learning in Large Language Models Bayesian?' (Chapter 5 and Appendix D): Chris Holmes had the core idea for the project. I implemented the initial code base, and demonstrated proof-of-concept experiments which verified our early hypotheses. Ziyu Wang led the development of the theoretical results with my support. I implemented the Llama and Mistral models, Ziyu Wang implemented the GPT models. Ziyu Wang implemented large parts of the late stages of the

code base and the experimental figures with my support, and performed many large-scale experiments and ablations. Ziyu Wang and I jointly wrote the paper, Chris Holmes edited it. Chris Holmes advised the project.

1.2.2 Other papers during my DPhil

Below I list papers published during my DPhil candidacy which were ommitted from this thesis in chronological order.

Machine Learning for Health (ML4H) 2020: Advancing Healthcare for All. Suproteem K. Sarkar, Subhrajit Roy, Emily Alsentzer, Matthew B. A. McDermott, **Fabian Falck**, Ioana Bica, Griffin Adams, Stephen Pfohl, Brett Beaulieu-Jones, Tristan Naumann, Stephanie L. Hyland. Proceedings of Machine Learning Research.

Ivy: Templated Deep Learning for Inter-Framework Portability. Daniel Lenton, Fabio Pardo, **Fabian Falck**, Stephen James, Ronald Clark. Arxiv preprint.

Identification of Underlying Disease Domains by Longitudinal Latent Factor Analysis for Secukinumab Treated Patients in Psoriatic Arthritis and Rheumatoid Arthritis Trials. Xuan Zhu, **Fabian Falck**, Sahra Ghalebikesabi, Matthias Kormaksson, Marc Vandemeulebroecke, Cong Zhang, Luis Santos, Chun Hei Kwok, Dominique West, Ann-Marie Mallon, Ruvie Martin, Aimee Readie, Kunal Gandhi, Gregory Ligozio, George Nicholson. American College of Rheumatology (ACR) Convergence 2021.

Machine Learning for Health (ML4H) 2021. Subhrajit Roy, Stephen Pfohl, Girmaw Abebe Tadesse, Luis Oala, **Fabian Falck**, Yuyin Zhou, Liyue Shen, Ghada Zamzmi, Purity Mugambi, Ayah Zirikly, Matthew BA McDermott, Emily Alsentzer. Proceedings of Machine Learning Research.

Neural Score Matching for High-Dimensional Causal Inference. Oscar Clivio, **Fabian Falck**, George Deligiannidis, Brieuc Lehmann, Chris Holmes. Artificial Intelligence and Statistics (AISTATS) 2022. Also presented at American Causal Inference Conference 2022.

Approximations to the Fisher Information Metric of Deep Generative Models for Out-Of-Distribution Detection. Sam Dauncey, Chris Holmes, Christopher Williams, **Fabian Falck**. Transactions on Machine Learning Research 2024. Earlier presented as 'On Gradients of Deep Generative Models for Representation-Invariant Anomaly Detection' at Trustworthy ML workshop, ICLR 2023.

A Critical Review of Causal Reasoning Benchmarks for Large Language Models. Linying Yang, Vik Shirvaikar, Oscar Clivio, **Fabian Falck**. 'Are LLMs Causal Parrots' workshop at AAAI 2024 (oral).

A framework for longitudinal latent factor modelling of treatment response in clinical trials. **Fabian Falck***, Xuan Zhu*, Sahra Ghalebikesabi, Matthias Kormaksson, Marc Vandemeulebroecke, Cong Zhang, Ruvie Martin, Stephen Gardiner, Chun Hei Kwok, Dominique M. West, Luis Santos, Chengcheng Tian, Yu Pang, Aimee Readie, Gregory Ligozio, Kunal K. Gandhi, Tom Nichols, Ann-Marie Mallon, Luke Kelly, David Ohlssen, George Nicholson. Journal of Biomedical Informatics.

Identifying treatment response subgroups in observational time-to-event data. Vincent Jeanselme, Chang Ho Yoon, **Fabian Falck**, Brian Tom, Jessica Barrett. Proceedings of the Fifth Machine Learning for Health Symposium, PMLR 297:55-75.

1.3 Thesis outline

The remainder of this thesis is organised as follows: In Chapter 2, we introduce Multi-Facet Clustering Variational Autoencoders (MFCVAE), a generative model

able to learn and capture multiple partitions of high-dimensional data. This paper presents a fully unsupervised, multi-partition clustering algorithm trained end-to-end. MFCVAE can be used for exploratory analysis to identify structure in data by identifying and disentangling abstract concepts.

In Chapter 3, we present a multi-resolution framework for U-Nets, and apply it to characterise the inductive bias of HVAEs. We identify U-Nets as learning the coefficients on finite-dimensional function spaces, which are truncations of an infinite-dimensional function space, and show a connection between average pooling and Haar wavelet truncation. We then leverage this framework by characterising state-of-the-art HVAE models as discretisations of an underlying (continuous) multi-resolution diffusion process. We also show how one can exploit this understanding: HVAEs learn an approximate representation of time in their residual state such that a time dependency can be removed in the HVAE cells through weight-sharing, improving parameter efficiency.

In Chapter 4, we expand our understanding of U-Nets. We identify preconditioning as the core design principle of U-Nets, provide a rigorous definition for U-Nets, and uncover their connection to ResNets. We analyse why U-Nets are a useful inductive bias in diffusion models, where they are a go-to architecture. We show how novel U-Nets can be designed to encode function constraints of a problem, such as the boundary conditions of a PDE. We also propose Multi-ResNets, a U-Net with a parameter-free encoder consisting of wavelet projections, and show how this neural network can achieve competitive and sometimes superior performance compared to classical U-Net architectures with the same number of parameters.

In Chapter 5, we shift our focus to LLMs, an instantiation of autoregressive models. We analyse the hypothesis whether in-context learning in LLMs under i.i.d. data follows Bayesian principles. We explore this question from a new angle, by measuring violations of what we call the martingale property (or conditionally identically distributed, c.i.d.), a fundamental property of Bayesian learning systems.

All information an LLM has is in its pretrained weights and the prompt. Therefore, an LLM should produce the same prediction for all future tokens when averaging over all previous tokens the LLM may have predicted. It is this intuition that the martingale property formalises. We investigate the martingale property on state-of-the-art LLMs, and find violations of the martingale property particularly when increasing the sample horizon, falsifying the hypothesis.

Appendices A to D augment the respective main texts of the presented papers in Chapters 2 to 5.

In Chapter 6, we conclude the thesis, discussing the contributions made and providing an outlook for future research.

2

Multi-Facet Clustering Variational Autoencoders

Abstract

Work in deep clustering focuses on finding a *single* partition of data. However, high-dimensional data, such as images, typically feature *multiple* interesting characteristics one could cluster over. For example, images of objects against a background could be clustered over the shape of the object and separately by the colour of the background. In this paper, we introduce *Multi-Facet Clustering Variational Autoencoders (MFCVAE)*, a novel class of variational autoencoders with a hierarchy of latent variables, each with a Mixture-of-Gaussians prior, that learns multiple clusterings simultaneously, and is trained fully unsupervised and end-to-end. MFCVAE uses a progressively-trained ladder architecture which leads to highly stable performance. We provide novel theoretical results for optimising the ELBO analytically with respect to the categorical variational posterior distribution, correcting earlier influential theoretical work. On image benchmarks, we demonstrate that our approach separates out and clusters over different aspects of the data in a disentangled manner. We also show other advantages of our model: the compositionality of its latent space and that it provides controlled generation of samples.

Contents

2.1 Introduction

Clustering is the task of finding structure by partitioning samples in a finite, unlabeled dataset according to statistical or geometric notions of similarity [84, 168, 244].For example, we might group items along axes of empirical variation in the data, or maximise internal homogeneity and external separation of items within and between clusters with respect to a specified distance metric. The choice of similarity measure and how one consequently validates clustering quality is fundamentally a subjective one: it depends on what is useful for a particular task [229, 244]. In this work, we are interested in uncovering abstract, latent characteristics/facets/aspects/levels of the data to understand and characterise the data-generative process. We further assume a fully exploratory, unsupervised setting without prior knowledge on the data, which could be exploited while fitting the clustering algorithm, and in particular without given ground-truth partitions at training time.

When being faced with high-dimensional data such as images, speech or electronic health records, items typically have more than one abstract characteristic. Consider the example of the MNIST dataset [133]: MNIST images possess at least two such characteristics: The digit class, which might impose the largest amount of statistical variation, and the style of the digit (e.g. stroke width). This naturally raises a question: by which characteristic is a clustering algorithm supposed to partition the data? In MNIST, both digit class and (the sub-categories of) style would be perfectly reasonable candidates to answer this question. In our exploratory setting described above, there is not one "correct" partition of the data.

Deep learning based clustering algorithms, so-called *deep clustering*, were particularly successful in recent years in dealing with high-dimensional data by compressing the inputs into a lower-dimensional latent space in which clustering is computationally tractable [4, 163]. However, almost all of these deep clustering algorithms find only a *single* partition of the data, typically the one corresponding to the given

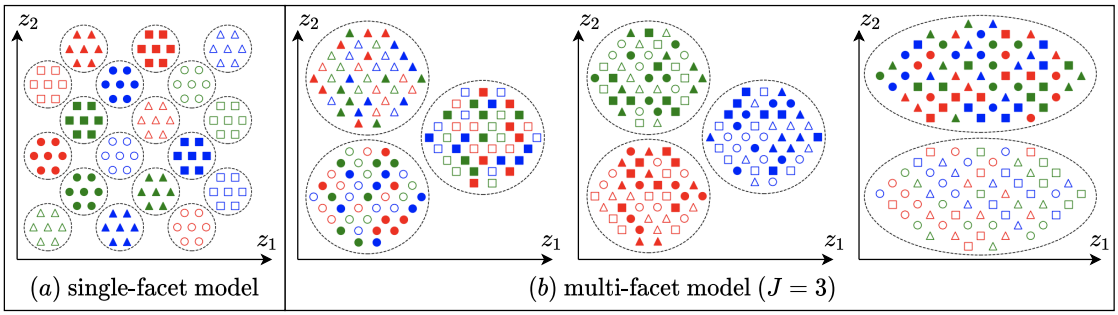


Figure 2.1: Latent space of a (a) single-facet model and a (b) multi-facet model ($J = 3$) with two dimensions (z_1, z_2) per facet. Both models perfectly separate the abstract characteristics of the data. However, the multi-facet model disentangles them into three sensible partitions (one per facet) and its required clusters scale linearly as opposed to exponentially w.r.t. the number of aspects in the data.

class label in a supervised dataset [97, 108, 115, 165, 201, 242, 246, 247]. When evaluating their model, said approaches validate clustering performance by treating the one supervision label (e.g. digit class in the case of MNIST) as the de-facto "ground truth clustering". We argue that restricting our view to a single facet C_1 rather than all or at least multiple facets $(C_1, C_2, \ldots, C_J)$ is an arbitrary, incomplete choice of formulating the problem of clustering a high-dimensional dataset.

To this end, we propose *Multi-Facet Clustering Variational Autoencoders (MFCVAE)*, a principled, probabilistic model which finds multiple characteristics of the data simultaneously through its multiple Mixtures-of-Gaussians (MoG) prior structure. Our contributions are as follows: (a) *Multi-Facet Clustering Variational Autoencoders (MFCVAE)*, a novel class of probabilistic deep learning models for unsupervised, multi-facet clustering in high-dimensional data that can be optimised end-to-end. (b) Novel theoretical results for the optimisation of the corresponding ELBO, correcting and extending an influential, related paper for the single-facet case. (c) Demonstrating MFCVAE's stable empirical performance in terms of multi-facet clustering of various levels of abstraction, compositionality of facets, generative, unsupervised classification, and diversity of generation.

2.2 Multi-facet clustering

High-dimensional data are inherently structured according to a number of abstract characteristics, and in an exploratory setting, it is clear that arbitrarily clustering by one of them is insufficient. However, the question remains whether these multiple facets should also be explicitly *represented* by the model. In particular, one might argue that a single partition could be used to represent all cross-combinations[1] of facets $\mathcal{C} = C_1 \times C_2 \times \cdots \times C_J$ where $C_j = \{1, 2, \ldots, K_j\}$, as in Fig. 2.1 (a). In this work, we explain that explicitly representing and clustering by multiple facets, as we do in MFCVAE and illustrated in Fig. 2.1 (b), has the following four properties that are especially desirable in an unsupervised learning setting:

(a) Discovering a multi-facet structure. We adopt a probabilistically principled, unsupervised approach, specifying an independent, multiple Mixtures of Gaussians (MoG) prior on the latent space. This induces a disentangled representation across facets, meaning that in addition to examples assigned to certain clusters being homogeneous, the facets (and their corresponding clusters) represent different abstract characteristics of the data (such as digit class or digit style). Because of this multi-facet structure, the total number of clusters required to represent a given multi-partition structure of the data scales *linearly* w.r.t. the number of data characteristics. In comparison, the number of clusters required in a single-facet model scales *exponentially* (see Fig. 2.1, and Appendix A.1 for details).

(b) Compositionality of facets. A multi-facet model has a compositional advantage: different levels of abstraction of the data are represented in separate latent variables. As we will show, this allows qualitatively diverse characteristics to be meaningfully combined.

[1]Note that in practice, not all cross-combinations of facets might be present. For example, in a dataset like MNIST, one might not observe 'right-tilted zeros', even though we observe 'right-tilted' digits and 'zeros'.

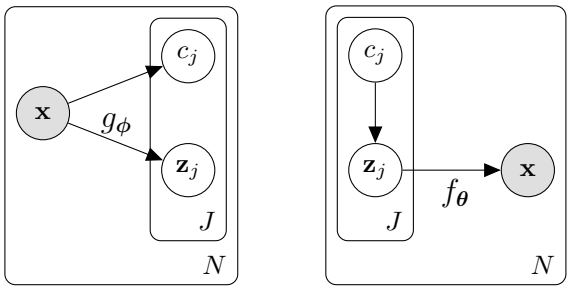


Figure 2.2: Graphical model of MFCVAE. [Left] Variational posterior, $q_\phi(\vec{\mathbf{z}}, \mathbf{c}|\mathbf{x})$. [Right] Generative model, $p_\theta(\mathbf{x}, \vec{\mathbf{z}}, \mathbf{c})$.

(c) Generative, unsupervised classification. Our method joins a myriad of single-facet clustering models in being able to accurately identify known class structures given by the label in standard supervised image datasets. However, in contrast to previous work, we are also able find interesting characteristics in other facets with homogeneous clusters. We stress that while we compare generative classification performance against other models to demonstrate statistical competitiveness, this task is *not* the main motivation for our fully unsupervised model.

(d) Diversity of generated samples. In a generative sense, the structure of the latent space allows us to compose new, synthetic examples by a set of J pairs of (continuous, discrete) latent variables. We can in particular intervene on each facet separately. This yields a rich set of options and fine-grained control for interventions and the diversity of generated examples.

We illustrate these four properties in our experiments in Section 2.4.

2.3 Multi-Facet Clustering Variational Autoencoders

Our model comprises J latent facets, each learning its own unique clustering of samples via a Mixture-of-Gaussians (MoG) distribution:

$$c_j \sim \mathrm{Cat}(\boldsymbol{\pi}_j), \quad \mathbf{z}_j \mid c_j \sim \mathcal{N}(\boldsymbol{\mu}_{c_j}, \boldsymbol{\Sigma}_{c_j}) \tag{2.1}$$

where $\boldsymbol{\pi}_j$ is the jth facet's K_j-dimensional vector of mixing weights, and $(\boldsymbol{\mu}_{c_j}, \boldsymbol{\Sigma}_{c_j})$ are the mean and covariance of the c_jth mixture component in facet j ($\boldsymbol{\Sigma}_{c_j}$ can be either diagonal or full).

The multi-facet generative model (Fig. 2.2 [Right]) is thus structured as

$$p_\theta(\mathbf{x}, \vec{\mathbf{z}}, \mathbf{c}) = p_\theta(\mathbf{x}|\vec{\mathbf{z}})p_\theta(\vec{\mathbf{z}}|\mathbf{c})p_\theta(\mathbf{c}) = p_\theta(\mathbf{x}|\vec{\mathbf{z}}) \prod_{j=1}^{J} p_\theta(\mathbf{z}_j|c_j)p_\theta(c_j), \tag{2.2}$$

where $\mathbf{c} = \{c_1, c_2, ..., c_J\}$, $\vec{\mathbf{z}} = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_J\}$, and $p_\theta(\mathbf{x}|\vec{\mathbf{z}})$ is a user-defined likelihood model, for example a product of Bernoulli or Gaussian distributions, which is parameterised with a deep neural network $f(\vec{\mathbf{z}}; \theta)$. Importantly, this structure in Eq. (2.2) encodes prior independence across facets, i.e. $p_\theta(\vec{\mathbf{z}}, \mathbf{c}) = \prod_j p_\theta(\mathbf{z}_j, c_j)$, thereby encouraging facets to learn clusterings that span distinct subspaces of $\vec{\mathbf{z}}$. The overall marginal prior $p_\theta(\vec{\mathbf{z}})$ can be interpreted as a product of independent MoGs.

2.3.1 VaDE tricks

To train this model, we wish to optimise the evidence lower bound (ELBO) of the data marginal likelihood using an amortised variational posterior $q_\phi(\vec{\mathbf{z}}, \mathbf{c}|\mathbf{x})$ (Fig. 2.2 [Left]), parameterised by a neural network $g(\mathbf{x}; \phi)$, within which we will perform Monte Carlo (MC) estimation where necessary to approximate expectations

$$\log p(\mathcal{D}) \geq \mathcal{L}(\mathcal{D}; \theta, \phi) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[\mathbb{E}_{q_\phi(\vec{\mathbf{z}},\mathbf{c}|\mathbf{x})}[\log \frac{p_\theta(\mathbf{x}, \vec{\mathbf{z}}, \mathbf{c})}{q_\phi(\vec{\mathbf{z}}, \mathbf{c}|\mathbf{x})}]\right]. \tag{2.3}$$

What should we choose for $q_\phi(\vec{\mathbf{z}}, \mathbf{c}|\mathbf{x})$? Training deep generative models with discrete latent variables can be challenging, as reparameterisation tricks so far developed, such as the Gumbel-Softmax trick [105, 152], necessarily introduce bias into the optimisation, and become unstable when a discrete latent variable has a high cardinality. Our setting where we have multiple discrete latent variables is even more challenging. First, the bias from using the Gumbel-Softmax trick compounds when there is a hierarchy of dependent latent variables, leading to poor optimisation

[142]. Second, we cannot necessarily avail ourselves of advances in obtaining good estimators for discrete latent variables as either they do not carry over to the hierarchical case [76], or are restricted to binary latent variables [185]. Third, we wish for light-weight optimisation, avoiding the introduction of additional neural networks whenever possible as this simplifies both training and neural specification.

Thus, we sidestep these problems, bias from relaxations of discrete variables *and* the downsides of additional amortised-posterior neural networks for the discrete latent variables, by developing the hierarchical version of the *VaDE trick*. This trick was first developed for clustering VAEs with a *single* Gaussian mixture in the generative model [108]. Informally, the idea (for a single-facet model) is to define a Bayes-optimal posterior for the discrete latent variable using the responsibilities of the constituent components of the mixture model; these responsibilities are calculated using samples taken from the amortised posterior for the continuous latent variable.

Estimating the ELBO for models of this form does not require us to take MC samples from discrete distributions—the data likelihood is conditioned only on the continuous latent variable $\vec{\mathbf{z}}$, which we sample using the reparameterization trick [121], and the posterior for $\vec{\mathbf{z}}$ is conditioned only on $\mathbf{x}$. Thus, when calculating the ELBO, we can cheaply marginalise out discrete latent variables where needed. In other words, we do not have to perform multiple forward passes through the decoder as neither it nor the $\vec{\mathbf{z}}$ samples we feed it depend on c.

As it is fundamental to our method, we now briefly recapitulate the original VaDE trick for VAEs with a single latent mixture (correcting a misapprehension in the original form of this idea) and will then cover our hierarchical extension[2].

Single-Facet VaDE Trick: Consider a single facet model, so the generative model is $p_\theta(\mathbf{x}, \mathbf{z}, c) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z}|c)p_\theta(c)$. Introduce a posterior $q_\phi(\mathbf{z}, c|\mathbf{x}) = q_\phi(\mathbf{z}|\mathbf{x})q_\phi(c|\mathbf{x})$

[2]We note that the original VaDE paper, besides the misapprehension discussed in Section 2.3.1 and Appendix A.2.1, proposed a highly complex training algorithm with various pre-training heuristics which we significantly simplified while maintaining or increasing performance (details in Appendix A.4.5).

where $q_\phi(\mathbf{z}|\mathbf{x})$ is a multivariate Gaussian with diagonal covariance and we assume independence of $\mathbf{z}$ and c given $\mathbf{x}$. The ELBO for this model for one datapoint is

$$\mathcal{L}(\mathbf{x};\theta,\phi) = \mathbb{E}_{q_\phi(\mathbf{z},c|\mathbf{x})}[\log \frac{p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z}|c)p_\theta(c)}{q_\phi(\mathbf{z}|\mathbf{x})q_\phi(c|\mathbf{x})}] = \mathbb{E}_{q_\phi(\mathbf{z},c|\mathbf{x})}[\log \frac{p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})p_\theta(c|\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})q_\phi(c|\mathbf{x})}], \tag{2.4}$$

where we have chosen to rewrite the generative model factorisation, $p_\theta(\mathbf{z}) = \sum_c p_\theta(\mathbf{z}|c)p_\theta(c)$ is the marginal mixture of Gaussians, and $p_\theta(c|\mathbf{z}) = p_\theta(\mathbf{z}|c)p_\theta(c)/p_\theta(\mathbf{z})$ is the Bayesian posterior for c.

Expanding out the ELBO, we get

$$\mathcal{L}(\mathbf{x};\theta,\phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - \mathrm{KL}\left[q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})\right] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \mathrm{KL}\left[q_\phi(c|\mathbf{x})||p_\theta(c|\mathbf{z})\right]. \tag{2.5}$$

We can *define* $q_\phi(c|\mathbf{x})$ such that $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \mathrm{KL}\left[q_\phi(c|\mathbf{x})||p_\theta(c|\mathbf{z})\right]$ is *minimal*, by construction, which is the case if we choose $q_\phi(c|\mathbf{x}) \propto \exp\left(\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(c|\mathbf{z})\right)$ as we will show in Theorem 2.1. This means that we can simply use samples from the posterior for $\mathbf{z}$ to define the posterior for c, using Bayes' rule within the latent mixture model.

Remark: We note, however, that in the original description of this idea in [108], it was claimed that $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \mathrm{KL}\left[q_\phi(c|\mathbf{x})||p_\theta(c|\mathbf{z})\right]$ could, in general, be set to *zero*, which is not the case. Rather, this KL can be minimised, in general, to a *non-zero* value. We discuss this misapprehension in more detail and why the empirical results in [108] are still valid in Appendix A.2.1.

Theorem 2.1. *(Single-Facet VaDE Trick) For any probability distribution $q_\phi(\mathbf{z}|\mathbf{x})$, the distribution $q_\phi(c|\mathbf{x})$ that minimises $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \mathrm{KL}\left[q_\phi(c|\mathbf{x})||p_\theta(c|\mathbf{z})\right]$ in (2.5) is*

$$\underset{q_\phi(c|\mathbf{x})}{\operatorname{argmin}} \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \mathrm{KL}\left[q_\phi(c|\mathbf{x})||p_\theta(c|\mathbf{z})\right] = \boldsymbol{\pi}(c|q_\phi(\mathbf{z}|\mathbf{x})) \tag{2.6}$$

with the minimum value attained being

$$\min_{q_\phi(c|\mathbf{x})} \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \mathrm{KL}\left[q_\phi(c|\mathbf{x})||p_\theta(c|\mathbf{z})\right] = -\log Z(q_\phi(\mathbf{z}|\mathbf{x})) \tag{2.7}$$

$$\textit{where} \qquad \boldsymbol{\pi}(c|q_\phi(\mathbf{z}|\mathbf{x})) := \frac{\exp\left(\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p(c|\mathbf{z})\right)}{Z(q_\phi(\mathbf{z}|\mathbf{x}))} \quad \textit{for } c = 1, \ldots, K \tag{2.8}$$

$$Z(q_\phi(\mathbf{z}|\mathbf{x})) := \sum_{c=1}^{K} \exp\left(\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p(c|\mathbf{z})\right) \ . \tag{2.9}$$

Proof: See Appendix A.2.1. □

Multi-facet VaDE Trick: In this work, we consider the case of having J facets, each with its own pair of variables $(\mathbf{z}_j, c_j)$. Perhaps surprisingly, we do *not* have to make a mean-field assumption *between* the J facets for $\mathbf{c}$ once we have made one for $\vec{\mathbf{z}}$. In other words, once we have chosen that $q_\phi(\vec{\mathbf{z}}, \mathbf{c}|\mathbf{x}) = q_\phi(\mathbf{c}|\mathbf{x}) \prod_{j=1}^{J} q_\phi(\mathbf{z}_j|\mathbf{x})$, where $q_\phi(\mathbf{z}_j|\mathbf{x})$ is defined to be a multivariate Gaussian with diagonal covariance for each j, the optimal $q_\phi(\mathbf{c}|\mathbf{x})$ similarly factorises[3]. We formalise this:

Theorem 2.2. *(Multi-Facet VaDE Trick for factorized $q_\phi(\vec{\mathbf{z}}|\mathbf{x})$, $p(\vec{\mathbf{z}}, \mathbf{c})$) For any factorized probability distribution $q_\phi(\vec{\mathbf{z}}|\mathbf{x}) = \prod_j q_\phi(\mathbf{z}_j|\mathbf{x})$, the distribution $q_\phi(\mathbf{c}|\mathbf{x})$ that minimises $\mathbb{E}_{q_\phi(\vec{\mathbf{z}}|\mathbf{x})} \mathrm{KL}\left[q_\phi(\mathbf{c}|\mathbf{x})||p_\theta(\mathbf{c}|\vec{\mathbf{z}})\right]$ under factorized prior $p(\vec{\mathbf{z}}, \mathbf{c}) = \prod_j p(\mathbf{z}_j, c_j)$ of (2.2) is*

$$\operatorname*{argmin}_{q_\phi(\mathbf{c}|\mathbf{x})} \mathbb{E}_{q_\phi(\vec{\mathbf{z}}|\mathbf{x})} \mathrm{KL}\left[q_\phi(\mathbf{c}|\mathbf{x})||p_\theta(\mathbf{c}|\vec{\mathbf{z}})\right] = \prod_j \boldsymbol{\pi}_j(c_j|q_\phi(\mathbf{z}_j|\mathbf{x})) \tag{2.10}$$

where the minimum value is attained at

$$\min_{q_\phi(\mathbf{c}|\mathbf{x})} \mathbb{E}_{q_\phi(\vec{\mathbf{z}}|\mathbf{x})} \mathrm{KL}\left[q_\phi(\mathbf{c}|\mathbf{x})||p_\theta(\mathbf{c}|\vec{\mathbf{z}})\right] = -\sum_j \log Z_j(q_\phi(\mathbf{z}_j|\mathbf{x})) \tag{2.11}$$

$$\textit{where} \qquad \boldsymbol{\pi}_j(c_j|q_\phi(\mathbf{z}_j|\mathbf{x})) := \frac{\exp(\mathbb{E}_{q_\phi(\mathbf{z}_j|\mathbf{x})} \log p_\theta(c_j|\mathbf{z}_j))}{Z_j(q_\phi(\mathbf{z}_j|\mathbf{x}))}, \textit{ for } c_j = 1, \ldots, K_j \tag{2.12}$$

$$Z_j(q_\phi(\mathbf{z}_j|\mathbf{x})) := \sum_{c_j=1}^{K_j} \exp(\mathbb{E}_{q_\phi(\mathbf{z}_j|\mathbf{x})} \log p_\theta(c_j|\mathbf{z}_j)) \ . \tag{2.13}$$

Proof: See Appendix A.2.2. □

[3]We also provide the VaDE trick for the general form of the posterior for $\vec{\mathbf{z}}$, i.e. without assuming the factorisation $q_\phi(\vec{\mathbf{z}}|\mathbf{x}) = \prod_{j=1}^{J} q_\phi(\mathbf{z}_j|\mathbf{x})$, in Appendix A.2.3.

Note that we use Eq. (2.12) as the probability distribution of assigning input $\mathbf{x}$ to clusters of facet j.

Armed with these theoretical results, we can now write the ELBO for our model, with the optimal posterior for $\mathbf{c}$, in a form that trivially admits stochastic estimation and does not necessitate extra recognition networks for $\mathbf{c}$,

$$\begin{aligned}\mathcal{L}^{\text{MFCVAE}}(\mathcal{D};\theta,\phi) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\Bigg[& \mathbb{E}_{q_\phi(\vec{\mathbf{z}}|\mathbf{x})} \log p_\theta(\mathbf{x}|\vec{\mathbf{z}}) \\ & - \sum_{j=1}^{J} \Big[\mathbb{E}_{q_\phi(c_j|\mathbf{x})} \operatorname{KL}(q_\phi(\mathbf{z}_j|\mathbf{x})||p_\theta(\mathbf{z}_j|c_j)) + \operatorname{KL}(q_\phi(c_j|\mathbf{x})||p(c_j))\Big]\Bigg]\end{aligned} \tag{2.14}$$

where the optimal $q_\phi(c_j|\mathbf{x})$ is given by Eq. (2.12) for each j.

To obtain the posterior distributions for $\mathbf{c}$, we take MC samples from $q_\phi(\vec{\mathbf{z}}|\mathbf{x})$ and use these to construct the posterior as in Eq. (2.12). We found one MC sample ($L = 1$; for each facet and for each $\mathbf{x}$) to be sufficient. We derive the complete MC estimator which we use as the loss function of our model and ablations on two alternative forms in Appendix A.3.

2.3.2 Neural implementation and training algorithm

It is worth pausing here to consider what neural architecture best suits our desire for learning multiple disentangled facets, and then further how we can best train our model to robustly elicit from it well-separated facets. In the introduction, we discussed the different plausible ways to cluster high-dimensional data, such as in MNIST digits by stroke thickness and class identity. These different aspects intuitively correspond to different levels of abstraction about the image. It is thus natural that these levels would be best captured by different depths of the neural networks in each amortised posterior. These ideas have motivated the use of *ladder networks* in deep generative models that aim to learn different facets of the input data into different layers of latent variables. Here, we take inspiration

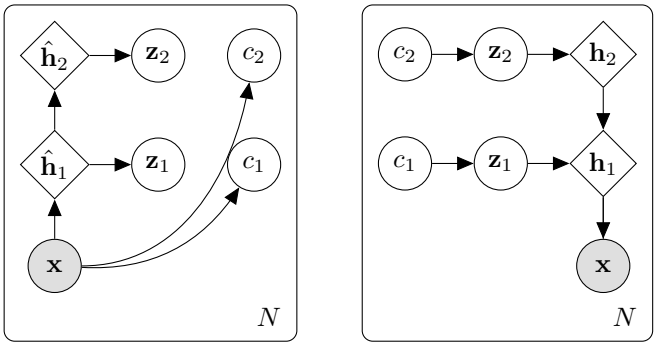


Figure 2.3: Ladder-MFCVAE architecture. [Left] Variational posterior. [Right] Generative model.

from *Variational Ladder Autoencoders (VLAEs)* [259]: A VLAE architecture has a deterministic "backbone" in both the recognition and generative model. The different layers of latent variables branch out from these at different depths along. This inductive bias naturally leads to stratification and does so without having to bear the computational cost of training a completely separate encoder (say) for each layer. Here, we use this ladder architecture for MFCVAE, as illustrated in Fig. 2.3, and refer to Appendix A.4.2 for further implementation details.

Further, we found *progressive training* [139], previously shown to help VLAEs learn layer-by-layer disentangled representations, to be of great use in making each facet consistently represent the same aspects of data. The general idea of progressive training is to start with training a single facet (typically the one corresponding to the deepest recognition and generative neural networks) for a certain number of epochs, and progressively and smoothly loop in the other facets one after the other. We discuss the details of our progressive training schedule in Appendix A.4.3. We find that both the VLAE architecture and progressive training are jointly important to stabilise training and get robust qualitative and quantitative results as we show in Appendix A.5.1.

2.4 Experiments

In the following, we demonstrate the usefulness of our model and its prior structure in four experimental analyses: (a) discovering a multi-facet structure (b) compositionality of latent facets (c) generative, unsupervised classification, and (d) diversity of generated samples from our model. We train our model on three image datasets: MNIST [133], 3DShapes (two configurations) [29] and SVHN [171]. We refer to Appendices A.4 and A.5 for experimental details and further results. We also provide our code implementing MFCVAE, using *PyTorch Distributions* [180], and reproducing our results at `https://github.com/FabianFalck/mfcvae`.

2.4.1 Discovering a multi-facet structure

We start by demonstrating that our model can discover a multi-facet structure in data. Fig. 2.4 visualises input examples representative of clusters in a two-facet ($J = 2$) model. For each facet j, input examples $\boldsymbol{x}$ with latent variable $\boldsymbol{z}_j$ are assigned to latent cluster $c_j = \text{argmax}_{c_j} \boldsymbol{\pi}_j(c_j|q_\phi(\mathbf{z}_j|\mathbf{x}))$ according to Eq. (2.12). Surprisingly, we find that we can represent the two most striking data characteristics—digit class and style (mostly in the form of stroke width, e.g. 'bold', 'thin') in MNIST, object shape and floor colour in 3DShapes (configuration 1), and digit class and background colour in SVHN—in two separate facets of the data. In each facet, clusters are homogeneous w.r.t. a value from the represented characteristic. When comparing our results on MNIST with LTVAE [138], the model closest to ours in its attempt to learn a clustered latent space of multiple facets, LTVAE struggles to separate data characteristics into separate facets (c.f. [138] Fig. 5; in particular, both facets learn digit class, i.e. this characteristic is not properly disentangled between facets), whereas MFCVAE better isolates the two.

To quantitatively assess the degree of disentanglement in the learned multi-facet structure of our model, we perform a set of supervised experiments. For each

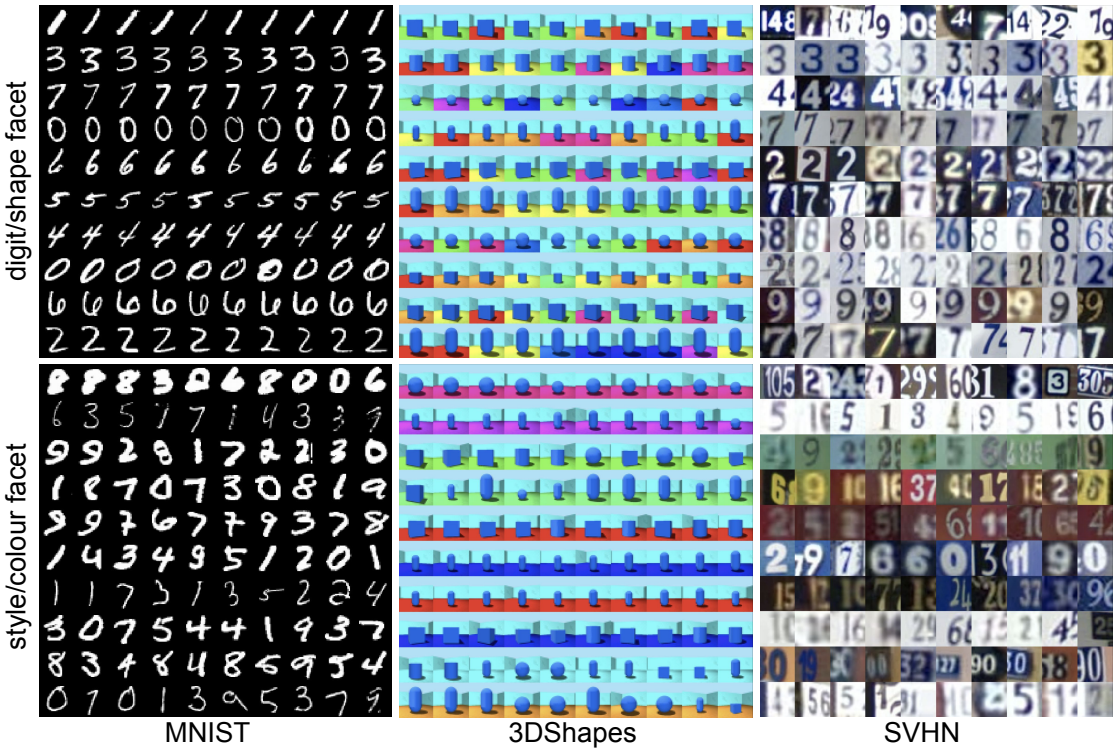


Figure 2.4: Input examples for clusters of MFCVAE with two-facets ($J = 2$) trained on MNIST, 3DShapes and SVHN. Clusters (rows) in each facet j are sorted in decreasing order by the average assignment probability of test inputs over each cluster. Inputs (columns) are sorted in decreasing order by their assignment probability $\max_{c_j} \boldsymbol{\pi}_j(c_j|q_\phi(\mathbf{z}_j|\mathbf{x}))$. We visualise the first 10 clusters and inputs from the test set (see Appendix A.5.3 for all clusters).

dataset, we formulate three classification tasks, for which we use latent embeddings $\mathbf{z}_1$, $\mathbf{z}_2$ and $\vec{\mathbf{z}}$, respectively, sampled from their corresponding amortised posterior, as inputs, and the label present in the dataset (e.g. digit class in MNIST) as the target. For each task and dataset, we train (on the training inputs) a multi-layer perceptron of one hidden layer with 100 hidden units and a ReLU activation, and an output layer followed by a softmax activation, which are the default hyperparameters in the Python package `sklearn`. Table 2.1 shows test accuracy of these experiments. We find that the supervised classifiers predict the supervised label with high accuracy when presented with latent embeddings which we found to cluster the abstract characteristic corresponding to this label, or with the concatenation of both latent embeddings. However, when presented with latent embeddings corresponding to the "non-label" facet, the classifier should—if facets are strongly disentangled—

Table 2.1: Supervised classification experiment to assess the disentanglement of MFCVAE's multi-facet structure on all three datasets. Values report test accuracy in %. Error bars are the sample standard deviation across 3 runs.

	MNIST	3DShapes config. 1		3DShapes config. 2		SVHN
	digit class	object shape	floor colour	object shape	wall colour	digit class
$\mathbf{z}_1$	17.34 (0.24)	95.00 (0.45)	20.00 (0.68)	98.26 (0.16)	73.40 (1.48)	69.46 (0.36)
$\mathbf{z}_2$	94.95 (0.04)	32.43 (1.38)	100.00 (0.00)	24.41 (1.34)	100.00 (0.00)	22.30 (0.16)
$\vec{\mathbf{z}}$	95.27 (0.07)	95.18 (0.42)	100.00 (0.00)	98.19 (0.30)	99.97 (0.06)	70.39 (0.29)

not be presented with useful information to learn the supervised mapping, and this is indeed what we find, observing significantly worse performance. This demonstrates the multi-facet structure of the latent space, which learns separate abstract characteristics of the data.

2.4.2 Compositionality of latent facets

A unique advantage of the prior structure of MFCVAE compared to other unsupervised generative models, say a VAE with an isotropic Gaussian prior, is that it allows different abstract characteristics to be composed in the separated latent space. Here, we show how this enables interventions on a per-facet basis, illustrated with a two-facet model where style/colour is learned in one facet and digit/shape is learned in the other facet. Let us have two inputs $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ assigned to two different style clusters according to Eq. (2.12) (and two different digit clusters). For both inputs, we obtain their latent representation $\tilde{\mathbf{z}}_j$ as the modes of $q_\phi(\mathbf{z}_j|\mathbf{x})$, respectively. Now, we swap the style/colour facet's representation, i.e. $\tilde{\mathbf{z}}_1$ of both inputs for MNIST, and $\tilde{\mathbf{z}}_2$ of both inputs for 3DShapes and SVHN, and pass these together with their unchanged digit/shape representation ($\tilde{\mathbf{z}}_2$ for MNIST and $\tilde{\mathbf{z}}_1$ for 3DShapes and SVHN) through the decoder $f(\vec{\mathbf{z}};\theta)$ to get reconstructions $\hat{\mathbf{x}}^{(1)} = f(\{\tilde{\mathbf{z}}_1^{(1)}, \tilde{\mathbf{z}}_2^{(2)}\};\theta)$ and $\hat{\mathbf{x}}^{(2)} = f(\{\tilde{\mathbf{z}}_1^{(2)}, \tilde{\mathbf{z}}_2^{(1)}\};\theta)$ which we visualise in Fig. 2.5 (see Appendix A.5.4 for a more rigorous explanation of this swapping procedure).

Surprisingly, by construction of this intervention in our multi-facet model, we

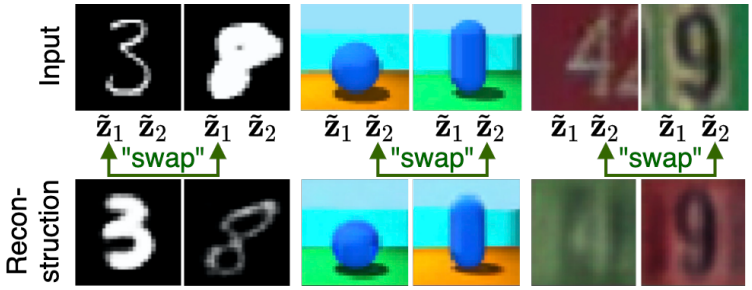


Figure 2.5: Reconstructions of two input examples when swapping their latent style/colour.

observe reconstructions that "swap" their style/background colour, yet in most cases preserve their digit/shape. This intervention is successful across a wide set of clusters on MNIST and 3DShapes. It works less so on SVHN where we hypothesise that this is due to the much more diverse dataset and (consequently) the model reaching a worse fit (see Section 2.4.3). We show further examples including failure cases in Appendix A.5.4 which show that our model learns a multi-facet structure allowing complex inventions.

2.4.3 Generative, unsupervised classification

Recall our fully unsupervised, exploratory setting of clustering where the goal is to identify and characterise multiple meaningful latent structures *de novo*. In practice, we have no ground-truth data partition—if labels were available, the task would be better formulated as a supervised classification in the first place. That said, it is often reasonable to assume that the class label in a supervised dataset represents a semantically meaningful latent structure that contributes to observed variation in the data. Indeed, this assumption underlies the common approach for benchmarking clustering models on labelled data: the class label is hidden during training; afterwards it is revealed as a pseudo ground-truth partitioning of the data for assessing clustering "accuracy". MFCVAE aims to capture multiple latent structures and can be deployed as a multi-facet generative classifier, as distinct from standard single-facet discriminative classifiers [168, p.30]. But we emphasise that

Table 2.2: Unsupervised clustering accuracy (%) of single-facet (SF) and multi-facet (MF), generative (G) and non-generative (NG) models on the test set. Error bars (if available) are the sample standard deviation across multiple runs. Results marked with $^{\eta}$ do not provide error bars.

Method	MNIST	SVHN
DEC ([242]; SF; NG)	84.3 $^{\eta}$	11.9 (0.4)
VaDE ([108]; MLP; SF; G)	94.46 $^{\eta}$; 89.09 (3.32)	27.03 (1.53)
VaDE ([108]; conv.; SF; G)	92.65 (1.14)	30.80 (1.99)
IMSAT ([97]; SF; NG)	98.4 (0.4)	57.3 (3.9)
ACOL-GAR ([115]; SF; NG)	98.32 (0.08)	76.80 (1.30)
VLAC ([236]; MF; G)	-	37.8 (2.2)
LTVAE ([138]; MF; G)	86.3	-
MFCVAE (ours; MF; G)	92.02 (3.18)	56.25 (0.93)

high classification accuracy is attained as a by-product, and is *not* our core goal—we do not explicitly target label accuracy, nor does high label accuracy necessarily correspond to the "best" multi-facet clustering.

Following earlier work, in Table 2.2, we report classification performance on MNIST and SVHN in terms of *unsupervised clustering accuracy* on the test set, which intuitively measures homogeneity w.r.t. a set of ground-truth clusters in each facet (see Appendix A.5.5 for a formal definition). We compare our method against commonly used single-facet (SF) and multi-facet (MF), generative (G) and non-generative (NG) deep clustering approaches (we use results as reported) of both deterministic and probabilistic nature. We report the mean and standard deviation (if available) of accuracy over T runs with different random seeds, where $T = 10$ for MFCVAE. For VaDE [108], we report results from the original paper, and our two implementations, one with a multi-layer perceptron encoder and decoder architecture, one using convolutional layers. Models marked with $^{\eta}$ state that they instead report the best result obtained from R restarts with different random seeds (DEC: $R = 20$, VaDE: $R = 10$). Both of these types of reporting in previous work—not providing error bars over several runs and picking the best run (while not providing error bars)—ignore stability of the model w.r.t. initialisation. We further discuss this issue and the importance of stability in deep clustering approaches in Appendix A.5.1.

MFCVAE is able to recover the assumed ground-truth clustering stably. It achieves competitive performance compared to other probabilistic deep clustering models, but is clearly outperformed by ACOL-GAR on SVHN, a single-facet, non-generative and deterministic model which does not possess three of the four properties demonstrated in Sections 2.4.1, 2.4.4) and 2.4.2). Besides the results presented in the table, we also note that MFCVAE performs strongly on 3DShapes, obtaining $99.46\% \pm 1.10\%$ for floor colour and $88.47\% \pm 1.82\%$ for object shape on configuration 1, and $100.00\% \pm 0.00\%$ for wall colour and $90.05\% \pm 2.65\%$ for object shape on configuration 2. Lastly, it is worth noting that we report classification performance for the same hyperparameter configurations and training runs of our model that are used in all experimental sections and in particular for Fig. 2.4, 2.5 and 2.6, i.e. our trained model has a pronounced multi-facet characteristic. In contrast, while it is somewhat unclear, LTVAE seems to report its clustering performance when trained with only a single facet, not when performing multi-facet clustering [138].

2.4.4 Diversity of generated samples

We lastly show that MFCVAE enables diverse generation of synthetic examples for each given cluster in the different facets, as a downstream task in addition to clustering. To obtain synthetic examples for a cluster c_j in facet j, we sample $\mathbf{z}_j$ from $p(\mathbf{z}_j|c_j)$, and sample $\mathbf{z}_{j'}$ from $p(\mathbf{z}_{j'})$ for all other facets $j' \neq j$. We then take the modes of $p_\theta(\mathbf{x}|\vec{\mathbf{z}})$ where $\vec{\mathbf{z}} = (\mathbf{z}_1, \ldots, \mathbf{z}_j, \ldots, \mathbf{z}_J)$ as the generated images. Fig. 2.6 shows synthetic examples generated from the models ($J = 2$) trained on MNIST, 3DShapes and SVHN.

For all three datasets, we observe synthetic samples that are homogeneous w.r.t. the characteristic value (e.g. 'red background') of a cluster in the chosen facet (as we sample this continuous latent variable from the conditional distribution), but heterogeneous and diverse w.r.t. all other facets (as we sample all other continuous latent variables from their marginal distribution). For example, on MNIST, when

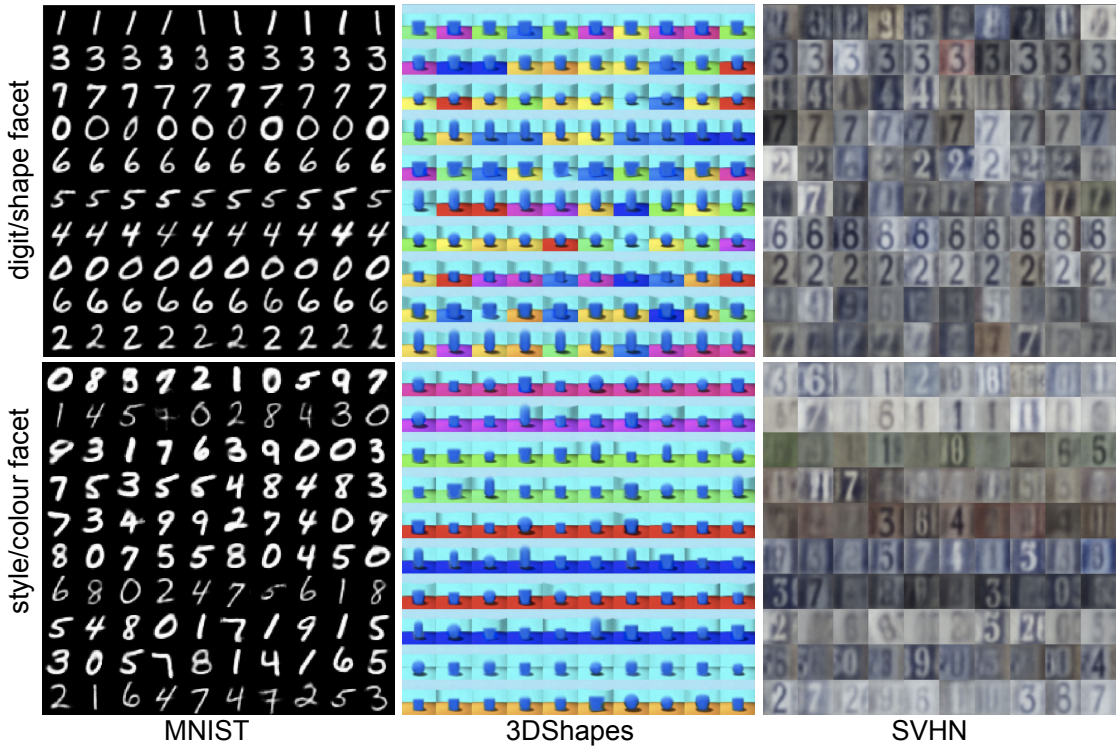


Figure 2.6: Synthetic samples generated from MFCVAE with two facets ($J = 2$) trained on MNIST, 3DShapes, and SVHN. For each cluster c_j in facet j, $\mathbf{z}_j$ is sampled from $p(\mathbf{z}_j|c_j)$ and $\mathbf{z}_{j'}$ is sampled from $p(\mathbf{z}_{j'})$ for the other facet $j' \neq j$. Each row corresponds to 10 random samples from a cluster. Clusters (rows) are sorted and selected (and are from the same trained model) as in Fig. 2.4 (see Appendix A.5.6 for visualisation of all clusters and comparison with LTVAE).

fixing a cluster in the digit facet, we observe generated samples that have the same digit class (e.g. all '1'), but are diverse in style (e.g. different 'stroke width'). Conversely, when fixing a cluster in the style facet, we get samples homogeneous in style, but heterogeneous in digit class. Likewise, on 3DShapes, fixing a cluster in the wall colour facet produces generations diverse in shape, but having the same wall color, and conversely when fixing the shape facet. Besides, in all clusters, generated samples are diverse w.r.t. other factors of variation on 3DShapes, such as orientation and scale. On SVHN, while less strong than in Fig. 2.4, these patterns extend here to the two facets style (background colour is particularly distinct) and digit class. These results are consistent with and underline the observed disentanglement of facets that we found in our previous experimental analyses. We also compare sample generation performance between MFCVAE and LTVAE and assess the

diversity of generations quantitatively in Appendix A.5.6.

2.5 Related work

Within the deep generative framework, various deep clustering models have been proposed. VaDE [108] is the most important prior work, a probabilistic model which has been highly influential in deep clustering. Related approaches, GM-VAEs [58] and GM-DGMs [170, 237, 238], have similar overall performance and explicitly represent the discrete clustering latent variable during training. Non-parametric approaches include DLDPMMs [170], and HDP-VAEs [75]. Further, many non-generative methods for clustering have been proposed that use neural components [58, 97, 115, 165, 201, 242, 246]. All these approaches, however, propose single-facet models.

Hierarchical VAEs can both be a way to learn more powerful models [34, 119, 151, 207, 221], but can also enable to separate out representations where each layer of latent variables represents a different aspect of the data. Variational Ladder Autoencoders (VLAEs) [259] aim to do the latter: to learn independent sets of latent variables, each representing some part of the data; but each group of latent variables within this set has a $\mathcal{N}(\mathbf{0}, \mathbf{1})$ prior, so it does not perform clustering. Recently, progressive training for VLAEs has been proposed [139] which sharpens the separation between layers. Here, we also mention disentanglement methods [31, 90, 116, 158] which likewise attempt to find separated latent variables. However, rather than discovering facets through the prior and a hierarchical structure, these techniques attempt to find statistically-independent representations via regularisation, leading the loss to deviate from the ELBO. Unfortunately, these methods require lucky selection of hyperparameters to work [147, 195], and do not provide a clustered latent space.

In the VAE literature more broadly, Joy et al. proposed a VAE model for semi-supervised learning which, in the unsupervised mode of operation, is similar to MFCVAE [111]. One important difference is their explicit paramterisation of the posterior of the categorical given the continuous latent, which is superfluous in MFCVAE by application of the VaDE trick. Furthermore, their expeirmental focus lies on semi-supervised learning, and their neural architecture and training algorithm are not 'hierarchy-inducing' as in MFCVAE.

Learning multiple clusterings simultaneously has been studied in the case of low-dimensional datasets [44, 49, 166, 187] under the names *alternative clusterings* and *non-redundant clustering*. However, when it comes to clustering high-dimensional data like images, approaches are rare. The recently proposed LTVAE [138] aims to perform this task, proposing a variational autoencoder with a latent tree model prior for a set of continuous latent variables $\vec{\mathbf{z}}$, of which each $\mathbf{z}_j$ has a GMM prior. The neural components are trained via stochastic gradient ascent under the ELBO; this is interleaved with a heuristic (hill-climbing) search algorithm to grow or prune the tree structure and message-passing to learn its nodes' GMM parameters of the current structure of the tree prior in a manner reminiscent of SVAEs [110], rendering the entire training algorithm *not* end-to-end differentiable (in contrast to MFCVAE). LTVAE learns multiple clusterings over the data, however, lacks a proper disentanglement of facets, as discussed in Section 2.4.1.

2.6 Conclusion

We introduced Multi-Facet Clustering Variational Autoencoders (MFCVAE), a novel class of probabilistic deep learning models for unsupervised, multi-partition clustering in high-dimensional data which is end-to-end differentiable. We provided novel theoretical results for optimising its ELBO, correcting and extending an influential related paper for the single-facet case. We demonstrated MFCVAE's empirical performance in terms of multi-facet clustering of various levels of abstraction, and

the usefulness of its prior structure for composing, classifying and generating samples, achieving state-of-the-art performance among deep probabilistic multi-facet models.

An important limitation of our work shared with many other deep clustering algorithms is the lack of a procedure to find good hyperparameters through a metric known at training time. Future work should explore: MFCVAE with $J > 2$; automatic tuning of hyperparameters J and K_j; application to large-scale datasets of other modalities; and regularising the model facet-wise to further enforce disentangled representations in the latent space [182]. While we successfully stabilised model training, further work will be key to harness the full potential of deep clustering models.

Acknowledgments and Disclosure of Funding

FF and HZ acknowledge the receipt of studentship awards from the Health Data Research UK-The Alan Turing Institute Wellcome PhD Programme in Health Data Science (Grant Ref: 218529/Z/19/Z). HZ acknowledges the receipt of Wellcome Cambridge Trust Scholarship. MW is grateful for the support of UCL Computer Science and The Alan Turing Institute. GN acknowledges support from the Medical Research Council Programme Leaders award MC_UP_A390_1107. CY is funded by a UKRI Turing AI Fellowship (Ref: EP/V023233/1). CH acknowledges support from the Medical Research Council Programme Leaders award MC_UP_A390_1107, The Alan Turing Institute, Health Data Research, U.K., and the U.K. Engineering and Physical Sciences Research Council through the Bayes4Health programme grant.

The authors report no competing interests.

We thank Tomas Lazauskas, Jim Madge and Oscar Giles from the Alan Turing Institute's Research Engineering team for their help and support. We thank Adam Huffman, Jonathan Diprose, Geoffrey Ferrari and Colin Freeman from the

Biomedical Research Computing team at the University of Oxford for their help and support. We thank Angela Wood and Ben Cairns for their support and useful discussions.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	Multi-facet clustering variational autoencoders
Publication Status	x Published □ Accepted for Publication □ Submitted for Publication □ Unpublished and unsubmitted work written in a manuscript style
Publication Details	Multi-facet clustering variational autoencoders. Fabian Falck*, Haoting Zhang*, Matthew Willetts, George Nicholson, Christopher Yau, Chris Holmes. Advances in Neural Information Processing Systems (NeurIPS) 2021. * indicates equal contribution.

Student Confirmation

Student Name:	Fabian Falck		
Contribution to the Paper	I follow the CRediT contributor role taxonomy (Brand et al., 2015). Fabian Falck: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing - Original Draft, Writing - Review & Editing, Visualization Haoting Zhang: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing - Original Draft, Writing - Review & Editing, Visualization Matthew Willetts: Conceptualization, Methodology, Resources, Formal analysis, Writing - Review & Editing, Supervision George Nicholson: Methodology, Formal analysis, Writing - Review & Editing Christopher Yau: Methodology, Formal analysis, Writing - Review & Editing Chris Holmes: Methodology, Resources, Writing - Review & Editing, Supervision, Project administration, Funding acquisition		
Signature	Fabian Falck	Date	September 29, 2024

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof. Chris Holmes		
Supervisor comments		
Signature	Date	November 27, 2024

This completed form should be included in the thesis, at the end of the relevant chapter.

3

A Multi-Resolution Framework for U-Nets with Applications to Hierarchical VAEs

Abstract

U-Net architectures are ubiquitous in state-of-the-art deep learning, however their regularisation properties and relationship to wavelets are understudied. In this paper, we formulate a multi-resolution framework which identifies U-Nets as finite-dimensional truncations of models on an infinite-dimensional function space. We provide theoretical results which prove that average pooling corresponds to projection within the space of square-integrable functions and show that U-Nets with average pooling implicitly learn a Haar wavelet basis representation of the data. We then leverage our framework to identify state-of-the-art hierarchical VAEs (HVAEs), which have a U-Net architecture, as a type of two-step forward Euler discretisation of multi-resolution diffusion processes which flow from a point mass, introducing sampling instabilities. We also demonstrate that HVAEs learn a representation of time which allows for improved parameter efficiency through weight-sharing. We use this observation to achieve state-of-the-art HVAE performance with half the number of parameters of existing models, exploiting the properties of our continuous-time formulation.

Contents

3.1 Introduction

U-Net architectures are extensively utilised in modern deep learning models. First developed for image segmentation in biomedical applications [197], U-Nets have been widely applied for text-to-image models [199], image-to-image translation [103], image restoration [156, 250], super-resolution [202], and multiview learning [216], amongst other tasks [132]. They also form a core building block as the neural architecture of choice in state-of-the-art generative models, particularly for images, such as HVAEs [34, 86, 125, 221] and diffusion models [52, 56, 92, 120, 172, 196, 199, 208, 209]. In spite of their empirical success, it is poorly understood why U-Nets work so well, and what regularisation they impose.

In likelihood-based generative modelling, various model classes are competing for superiority, including normalizing flows [91, 118], autoregressive models [35, 223], diffusion models, and hierarchical variational autoencoders (HVAEs), the latter two of which we focus on in this work. HVAEs form groups of latent variables with a conditional dependence structure, use a U-Net neural architecture, and are trained with the typical VAE ELBO objective (for a detailed introduction to HVAEs, see Section 1.1.3). HVAEs show impressive synthesis results on facial images, and yield competitive likelihood performance, outperforming autoregressive models, VAEs and flow models on computer vision benchmarks [34, 221]. HVAEs have undergone a journey of design iterations and architectural improvements in recent years, for example the introduction a deterministic backbone [67, 194, 260] and ResNet elements [87, 119] with shared parameters between the inference and generative model parts. There has also been a massive increase in the number of latent variables and overall stochastic depth, as well as the use of different types of residual cells in the decoder [34, 221] (see §3.4 and Fig. B.1 for a detailed discussion). However, a theoretical understanding of these choices is lacking. For instance, it has not been shown why a residual backbone may be beneficial, or what the specific cell structures in VDVAE [34] and NVAE [221] correspond to, or how they could be improved.

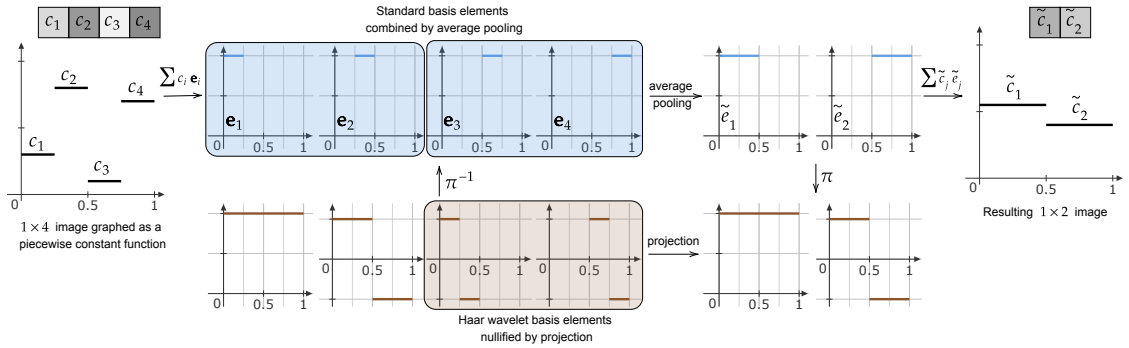


Figure 3.1: U-Nets with average pooling learn a Haar wavelet basis representation of the data.

In this paper we provide a theoretical framework for understanding the latent spaces in U-Nets, and apply this to HVAEs specifically. Doing so allows us to relate HVAEs to diffusion processes, and also to motivate a new type of piecewise time-homogenenous model which demonstrates state-of-the-art performance with approximately half the number of parameters of a VDVAE [34]. More formally, our contributions are as follows: **(a)** We provide a multi-resolution framework for U-Nets. We formally define U-Nets as acting over a multi-resolution hierarchy of $L^2([0,1]^2)$. We prove that average pooling is a conjugate operation to projection in the Haar wavelet basis within $L^2([0,1]^2)$. We use this insight to show how U-Nets with average pooling implicitly learn a Haar wavelet basis representation of the data (see Fig. 3.1), helping to characterise the regularisation within U-Nets. **(b)** We apply this framework to state-of-the-art HVAEs as an example, identifying their residual cell structure as a type of two-step forward Euler discretisation of a multi-resolution diffusion bridge. We uncover that this diffusion process flows from a point mass, which causes instabilities, for instance during sampling, and identify parameter redundancies through our continuous-time formulation. Our framework both allows us to understand the heuristic choices of existing work in HVAEs and enables future work to optimise their design, for instance their residual cell. **(c)** In our experiments, we demonstrate these sampling instabilities and train HVAEs with the largest stochastic depth ever achieved, yielding state-of-the-art performance with half the number of parameters by exploiting our theoretical insights. We explain these results by uncovering that HVAEs secretly represent time in their

state and show that they use this information during training. We finally provide extensive ablation studies which, for instance, rule out other potential factors which correlate with stochastic depth, show the empirical gain of multiple resolutions, and find that Fourier features (which discrete-time diffusion models can strongly benefit from [120]) do not improve performance in the HVAE setting.

3.2 The Multi-Resolution Framework

A grayscale image with infinite resolution can be thought of as the graph[1] of a two-dimensional function over the unit square. To store these infinitely-detailed images in computers, we project them to some finite resolution. These projections can still be thought of as the graphs of functions with support over the unit square, but they are piecewise constant on finitely many intervals or 'pixels', e.g. 512^2 pixels, and we store the function values obtained at these pixels in an array or 'grid'. The relationship between the finite-dimensional version and its infinitely-fine counterpart depends entirely on how we construct this projection to preserve the details we wish to keep. One approach is to prioritise preserving the large-scale details of our images, so unless closely inspected, the projection is indistinguishable from the original. This can be achieved with a multi-resolution projection [47] of the image. In this section we introduce a *multi-resolution framework* for constructing neural network architectures that utilise such projections, prove what regularisation properties they impose, and show as an example how HVAEs with a U-Net [197] architecture can be interpreted in our framework. Proofs of all theorems in the form of an extended exposition of our framework can be found in Appendix B.1.

[1] For a function $f(\cdot)$, its graph is the set $\bigcup_{x\in[0,1]^2}\{x, f(x)\}$.

3.2.1 Multi-Resolution Framework: Definitions and Intuition

What makes a multi-resolution projection good at prioritising large-scale details can be informally explained through the following thought experiment. Imagine we have an image, represented as the graph of a function, and its finite-dimensional projection drawn on a wall. We look at the wall, start walking away from it and stop when the image and its projection are indistinguishable by eye. The number of steps we took away from the wall can be considered our measure of 'how far away' the approximation is from the underlying function. The goal of the multi-resolution projection is therefore to have to take as few steps away as possible. The reader is encouraged to physically conduct this experiment with the examples provided in Appendix B.2.1. We can formalise the aforementioned intuition by defining a *multi-resolution hierarchy* [47] of subspaces we may project to:

Definition 3.1. [*Daubechies (1992)* [47]] Given a nested sequence of *approximation spaces* $\cdots \subset V_1 \subset V_0 \subset V_{-1} \subset \cdots$, $\{V_{-j}\}_{j\in\mathbb{Z}}$ is a *multi-resolution hierarchy* of the function space $L^2(\mathbb{R}^m)$ if: **(A1)** $\overline{\bigcup_{j\in\mathbb{Z}} V_{-j}} = L^2(\mathbb{R}^m)$; **(A2)** $\bigcap_{j\in\mathbb{Z}} V_{-j} = \{0\}$; **(A3)** $f(\cdot) \in V_{-j} \Leftrightarrow f(2^j\cdot) \in V_0$; **(A4)** $f(\cdot) \in V_0 \Leftrightarrow f(\cdot - n) \in V_0$ for $n \in \mathbb{Z}$. For a compact set $\mathbb{X} \subset \mathbb{R}^m$, a *multi-resolution hierarchy* of $L^2(\mathbb{X})$ is $\{V_{-j}\}_{j\in\mathbb{Z}}$ as defined above, restricting functions in V_{-j} to be supported on $\mathbb{X}$.

In Definition 3.1, the index j references how many steps we took in our thought experiment, so negative j corresponds to 'zooming in' on the images. The original image[2] is a member of $L^2([0,1]^2)$, the space of square-integrable functions on the unit square, and its finite projection to $2^j \cdot 2^j$ many pixels is a member of V_{-j}. Images can be represented as piecewise continuous functions in the subspaces $V_{-j} = \{f \in L^2([0,1]) \mid f|_{[2^{-j}\cdot k, 2^{-j}\cdot(k+1))} = c_k,\, k \in \{0,\ldots,2^j-1\},\, c_k \in \mathbb{R}\}$. The nesting property $V_{-j+1} \subset V_{-j}$ ensures that any image with $(2^{j-1})^2$ pixels can also

[2]We here focus on grayscale, squared images for simplicity, but note that our framework can be seamlessly extended to colour images with a Cartesian product $L^2([0,1]^2) \times L^2([0,1]^2) \times L^2([0,1]^2)$, and other continuous signals such as time series.

be represented by $(2^j)^2$ pixels, but at a higher resolution. Assumption (**A1**) states that with infinitely many pixels, we can describe any infinitely detailed image. In contrast, (**A2**) says that with no pixels, we cannot approximate any images. Assumptions (**A3**) and (**A4**) allow us to form a basis for images in any V_{-j} if we know the basis of V_0. One basis made by extrapolating from V_0 in this way is known as a *wavelet basis* [47]. Wavelets have proven useful for representing images, for instance in the JPEG standard [215], and are constructed to be orthonormal.

Now suppose we have a probability measure ν_∞ over infinitely detailed images represented in $L^2([0,1]^2)$ and wish to represent it at a lower resolution. Similar to how we did for infinitely detailed images, we want to project the measure ν_∞ to a lower dimensional measure ν_j on the finite dimensional space V_{-j}. In extension to this, we want the ability to reverse this projection so that we may sample from the lower dimensional measure and create a generative model for ν_∞. We would like to again prioritise the presence of large-scale features of the original image within the lower dimensional samples. We do this by constructing a *multi-resolution bridge* from ν_∞ to ν_j, as defined below.

Definition 3.2. Let $\mathbb{X} \subset \mathbb{R}^m$ be compact, $\{V_{-j}\}_{j=0}^\infty$ be a multi-resolution hierarchy of $L^2(\mathbb{X}) = \overline{\bigcup_{j\in\mathbb{N}_0} V_{-j}}$ and $V_0 = \{0\}$. If $\mathbb{D}(L^2(\mathbb{X}))$ is the space of probability measures over $L^2(\mathbb{X})$, then a family of probability measures $\{\nu_t\}_{t\in[0,1]}$ on $L^2(\mathbb{X})$ is a *multi-resolution bridge* if:

(i) there exist increasing times $\mathcal{I} := \{t_j\}_{j\in\mathbb{N}_0}$ where $t_0 = 0$, $\lim_{j\to\infty} t_j = 1$, such that $s \in [t_j, t_{j+1})$ implies $\text{supp}(\nu_s) \subset V_{-j}$, i.e $\nu_s \in \mathbb{D}(V_{-j})$; and,

(ii) for $s \in (0,1)$, the mapping $s \mapsto \nu_s$ is continuous for $s \in (t_j, t_{j+1})$ for some j.

The continuous time dependence in Definition 3.2 plays a movie of the measure ν_0 supported on V_0 growing to ν_∞, a measure on images with infinite resolution. At a time interval $[t_j, t_{j+1})$, the space V_{-j} which the measure is supported on is fixed. We may therefore define a finite-dimensional model transporting probability measures

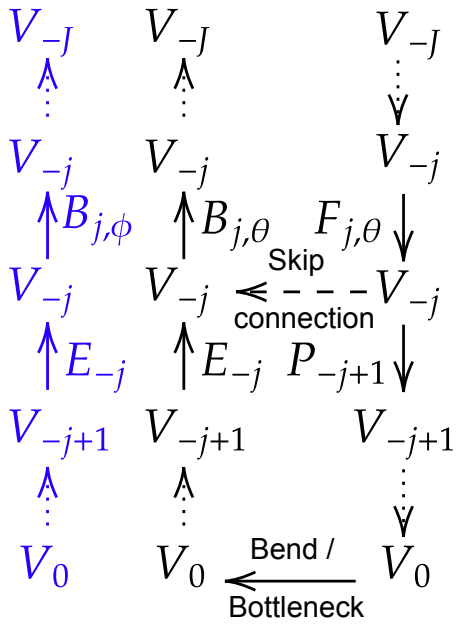


Figure 3.2: A U-Net in our multi-resolution framework. See Appendix B.2.2 for details.

within V_{-j}, but at t_{j+1} the support flows over to V_{-j-1}. Given a multi-resolution hierarchy, we may glue these finite models, each acting on a disjoint time interval, together in a unified fashion. In Theorem 3.1 we show this for the example of a continuous-time multi-resolution diffusion process truncated up until some time $t_J = T \in (0, 1)$ and in the *standard basis* discussed in §3.2.2, which will be useful when viewing HVAEs as discretisations of diffusion processes on functions in §3.2.3.

Theorem 3.1. *Let $B_j : [t_j, t_{j+1}) \times \mathbb{D}(V_{-j}) \mapsto \mathbb{D}(V_{-j})$ be a linear operator (such as a diffusion transition kernel, see Appendix B.1) for $j < J$ with coefficients $\mu^{(j)}, \sigma^{(j)} : [t_j, t_{j+1}) \times V_{-j} \mapsto V_{-j}$, and define the natural extensions within V_{-J} in bold, i.e. $\boldsymbol{B}_j \coloneqq B_j \oplus \boldsymbol{I}_{V_{-j}^{\perp}}$. Then the operator $\boldsymbol{B} : [0, T] \times \mathbb{D}(V_{-J}) \mapsto \mathbb{D}(V_{-J})$ and the coefficients $\boldsymbol{\mu}, \boldsymbol{\sigma} : [0, T] \times V_{-J} \mapsto V_{-J}$ given by*

$$\boldsymbol{B} \coloneqq \sum_{j=0}^{J} \mathbb{1}_{[t_j, t_{j+1})} \cdot \boldsymbol{B}_j, \quad \boldsymbol{\mu} \coloneqq \sum_{j=0}^{J} \mathbb{1}_{[t_j, t_{j+1})} \cdot \boldsymbol{\mu}^{(j)}, \quad \boldsymbol{\sigma} \coloneqq \sum_{j=0}^{J} \mathbb{1}_{[t_j, t_{j+1})} \cdot \boldsymbol{\sigma}^{(j)},$$

induce a multi-resolution bridge of measures from the dynamics for $t \in [0, T]$ and on the standard basis as $dZ_t = \boldsymbol{\mu}_t(Z_t)dt + \boldsymbol{\sigma}_t(Z_t)dW_t$ (see Appendix B.1.4 for details) for $Z_t \in V_{-j}$ for $t \in [t_j, t_{j+1})$, i.e. a multi-resolution diffusion process.

The concept of a multi-resolution bridge will become important in Section 3.2.2 where we will show that current U-Net bottleneck structures used for unconditional sampling impose a multi-resolution bridge on the modelled densities. To preface this, we here provide a description of a U-Net within our framework, illustrated in 3.2. Consider $B_{j,\theta}, F_{j,\theta} : \mathbb{D}(V_{-j}) \to \mathbb{D}(V_{-j})$ as the forwards and backwards passes of a U-Net on resolution j. Further, let $P_{-j+1} : \mathbb{D}(V_{-j}) \to \mathbb{D}(V_{-j+1})$ and $E_{-j} : \mathbb{D}(V_{-j+1}) \to \mathbb{D}(V_{-j})$ be the projection (here: average pooling) and embedding maps (e.g. interpolation), respectively. When using an L^2-reconstruction error, a U-Net [197] architecture implicitly learns a sequence of models $\mathbf{B}_{j,\phi} : \mathbb{D}(V_{-j+1}) \times \mathbb{D}(V_{-j+1}^{\perp}) \mapsto \mathbb{D}(V_{-j})$ due to the orthogonal decomposition $V_{-j} = V_{-j+1} \oplus U_{-j+1}$ where $U_{-j+1} \coloneqq V_{-j} \cap V_{-j+1}^{\perp}$. The backwards operator for the U-Net has a (bottleneck) input from $\mathbb{D}(V_{-j+1})$ and a (skip) input yielding information from $\mathbb{D}(V_{-j+1}^{\perp})$. A simple *bottleneck* map $U_{j,\theta} : \mathbb{D}(V_{-j}) \to \mathbb{D}(V_{-j})$ (without skip connection) is given by

$$U_{j,\theta} \coloneqq B_{j,\theta} \circ E_{-j} \circ P_{-j+1} \circ F_{j,\theta}, \tag{3.1}$$

and a U-Net bottleneck with skip connection is

$$\mathbf{U}_{j,\phi} \coloneqq B_{j,\phi}(E_{-j} \circ P_{-j+1} \circ F_{j,\theta}, F_{j,\theta}). \tag{3.2}$$

In HVAEs, the map $\mathbf{U}_{j,\phi} : \mathbb{D}(V_{-j}) \to \mathbb{D}(V_{-j})$ is trained to be the identity by minimising reconstruction error, and further shall approximate $U_{j,\theta} \approx \mathbf{U}_{j,\phi}$ via a KL divergence. The L^2-reconstruction error for $\mathbf{U}_{j,\phi}$ has an orthogonal partition of the inputs from $V_{-j+1} \times V_{-j}$, hence the only new subspace added is U_{-j+1}. As each orthogonal U_{-j+1} is added sequentially in HVAEs, the skip connections induce a multi-resolution structure of this hierarchical neural network structure. What we will investigate in Theorem 3.3 is the regularisation imposed on this partitioning by enforcing $U_{j,\theta} \approx \mathbf{U}_{j,\phi}$, as is often enforced for generative models with VAEs.

$$\begin{array}{ccc} (V_{-j}, \mathbf{E}_j) & \xrightarrow{\text{pool}_{-j,-j+1}} & (V_{-j+1}, \mathbf{E}_{j-1}) \\ \uparrow{\scriptstyle \pi_j^{-1}} & & {\scriptstyle \pi_{j-1}}\downarrow \\ (V_{-j}, \mathbf{\Psi}_j) & \xrightarrow[\text{proj}_{V_{-j+1}}]{} & (V_{-j+1}, \mathbf{\Psi}_{j-1}) \end{array}$$

Figure 3.3: The function space V_{-j} remains the same, but the basis changes under π_j.

3.2.2 The regularisation property imposed by U-Net architectures with average pooling

Having defined U-Net architectures within our multi-resolution framework, we are now interested in the regularisation they impose. We do so by analysing a U-Net when skip connections are absent, so that we may better understand what information is transferred through each skip connection when they are present. In practice, a pixel representation of images is used when training U-Nets, which we henceforth call the *standard basis* (see B.1.2, Eq. (B.9)). The standard basis is not convenient to derive theoretical results. It is instead preferable to use a basis natural to the multi-resolution bridge imposed by a U-Net with a corresponding projection operation, which for average pooling is the *Haar (wavelet) basis* [81] (see Appendix B.1.2). The Haar basis, like a Fourier basis, is an orthonormal basis of $L^2(\mathbb{X})$ which has desirable L^2-approximation properties. We formalise this in Theorem 3.2 which states that the dimension reduction operation of average pooling in the standard basis is a conjugate operation to co-ordinate projection within the Haar basis (details are provided in Appendix B.1.2).

Theorem 3.2. *Given V_{-j} as in Definition 3.1, let $x \in V_{-j}$ be represented in the standard basis $\mathbf{E}_j$ and Haar basis $\mathbf{\Psi}_j$. Let $\pi_j : \mathbf{E}_j \mapsto \mathbf{\Psi}_j$ be the change of basis map illustrated in Fig. 3.3, then we have the conjugacy $\pi_{j-1} \circ pool_{-j,-j+1} = proj_{V_{-j+1}} \circ \pi_j$.*

Theorem 3.2 means that if we project an image from V_{-j} to V_{-j+1} in the Haar wavelet basis, we can alternatively view this as changing to the standard basis via π_j^{-1}, performing average pooling, and reverting back via π_{j-1} (see Figure 3.3). This

is important because the Haar basis is orthonormal, which in Theorem 3.3 allows us to precisely quantify what information is lost with average pooling.

Theorem 3.3. *Let* $\{V_{-j}\}_{j=0}^{J}$ *be a multi-resolution hierarchy of* V_{-J} *where* $V_{-j} = V_{-j+1} \oplus U_{-j+1}$*, and further, let* $F_{j,\phi}$*,* $B_{j,\theta} : \mathbb{D}(V_{-j}) \mapsto \mathbb{D}(V_{-j})$ *be such that* $B_{j,\theta}F_{j,\phi} = I$ *with parameters* ϕ *and* θ*. Define* $\boldsymbol{F}_{j_1|j_2,\phi} \coloneqq \boldsymbol{F}_{j_1,\phi} \circ \cdots \circ \boldsymbol{F}_{j_2,\phi}$ *by* $\boldsymbol{F}_{j,\phi} : \mathbb{D}(V_{-j}) \mapsto \mathbb{D}(V_{-j+1})$ *where* $\boldsymbol{F}_{j,\phi} \coloneqq proj_{V_{-j+1}} \circ F_{j,\phi}$*, and analogously define* $\boldsymbol{B}_{j_1|j_2,\theta}$ *with* $\boldsymbol{B}_{j,\theta} \coloneqq B_{j,\theta} \circ embd_{V_{-j}}$*. Then, the sequence* $\{\boldsymbol{B}_{1|j,\theta}(\boldsymbol{F}_{1|J,\phi}\nu_J)\}_{j=0}^{J}$ *forms a discrete multi-resolution bridge between* $\boldsymbol{F}_{1|J,\phi}\nu_J$ *and* $\boldsymbol{B}_{1|J,\theta}\boldsymbol{F}_{1|J,\phi}\nu_J$ *at times* $\{t_j\}_{j=1}^{J}$*, and*

$$\sum_{j=0}^{J} \mathbb{E}_{X_{t_j}\sim\nu_j} \left\| \mathrm{proj}_{U_{-j+1}} X_{t_j} \right\|_2^2 / \left\| \boldsymbol{F}_{j|J,\phi} \right\|_2^2 \leq (\mathcal{W}_2(\boldsymbol{B}_{1|J,\theta}\boldsymbol{F}_{1|J,\phi}\nu_J, \nu_J))^2, \tag{3.3}$$

where $\mathcal{W}_2$ *is the Wasserstein-2 metric and* $\left\| \boldsymbol{F}_{j|J,\phi} \right\|_2$ *is the Lipschitz constant of* $\boldsymbol{F}_{j|J,\phi}$*.*

Theorem 3.3 states that the bottleneck component of a U-Net pushes the latent data distribution to a finite multi-resolution basis, specifically a Haar basis when average pooling is used. To see this, note that the RHS of Eq. (3.3) is itself upper-bounded by the L^2-reconstruction error. This is because the Wasserstein-2 distance finds the infinimum over all possible couplings between the data and the 'reconstruction' measure, hence any coupling (induced by the learned model) bounds it. Note that models using a U-Net, for instance HVAEs or diffusion models, either directly or indirectly optimise for low reconstruction error in their loss function. The LHS of Eq. (3.3) represents what percentage of our data enters the orthogonal subspaces $\{U_{-j}\}_{j=0}^{J}$ which are (by Theorem 3.2) discarded by the bottleneck structure when using a U-Net architecture with average pooling. Theorem 3.3 thus shows that as we minimise the reconstruction error during training, we minimise the percentage of our data transported to the orthogonal sub-spaces $\{U_{-j}\}_{j=0}^{J}$. Consequently, the bottleneck architecture implicitly decomposes our data into a Haar wavelet decomposition, and when the skip connections are absent (like in a traditional auto-encoder) our network learns to compress the discarded

subspaces U_{-j}. This characterises the regularisation imposed by a U-Net in the absence of skip connections.

These results suggest that U-Nets with average pooling provide a direct alternative to Fourier features [120, 190, 213, 230] which impose a Fourier basis, an alternative orthogonal basis on $L^2(\mathbb{X})$, as with skip connections the U-Net adds each subspace U_{-j} sequentially. However, unlike Fourier bases, there are in fact a multitude of wavelet bases which are all encompassed by the multi-resolution framework, and in particular, Theorem 3.3 pertains to all of them for the bottleneck structure. This opens the door to exploring conjugacy operations beyond average pooling induced by other wavelet bases optimised for specific data types.

3.2.3 Example: HVAEs as Diffusion Discretisations

To show what practical inferences we can derive from our multi-resolution framework, we apply it to analyse state-of-the-art HVAE architectures (see Appendix 1.1.3 for an introduction), identifying parameter redundancies and instabilities. Here and in our experiments, we focus on VDVAEs [34]. We provide similar results for Markovian HVAEs [28, 207] and NVAEs [221] (see § 3.4) in Appendix B.1.5.

We start by inspecting VDVAEs. As we show next, we can tie the computations in VDVAE cells to the (forward and backward) operators $F_{j,\phi}$ and $B_{j,\theta}$ within our framework and identify them as a type of two-step forward Euler discretisation of a diffusion process. When used with a U-Net, as is done in VDVAE [34], this creates a *multi-resolution diffusion bridge* by Theorem 3.4.

Theorem 3.4. *Let $t_J \coloneqq T \in (0, 1)$ and consider (the p_θ backward pass) $\boldsymbol{B}_{\theta,1|J} : \mathbb{D}(V_{-J}) \mapsto \mathbb{D}(V_0)$ given in multi-resolution Markov process in the standard basis:*

$$dZ_t = (\overleftarrow{\mu}_{1,t}(Z_t) + \overleftarrow{\mu}_{2,t}(Z_t))dt + \overleftarrow{\sigma}_t(Z_t)dW_t, \tag{3.4}$$

where $proj_{U_{-j}} Z_{t_j} = 0$, $\|Z_t\|_2 > \|Z_s\|_2$ with $0 \leq s < t \leq T$ and for a measure $\nu_J \in \mathbb{D}(V_{-J})$ we have X_T, $Z_0 \sim \boldsymbol{F}_{\phi,J|1}\nu_J = \delta_{\{0\}}$. Then, VDVAEs approximates this

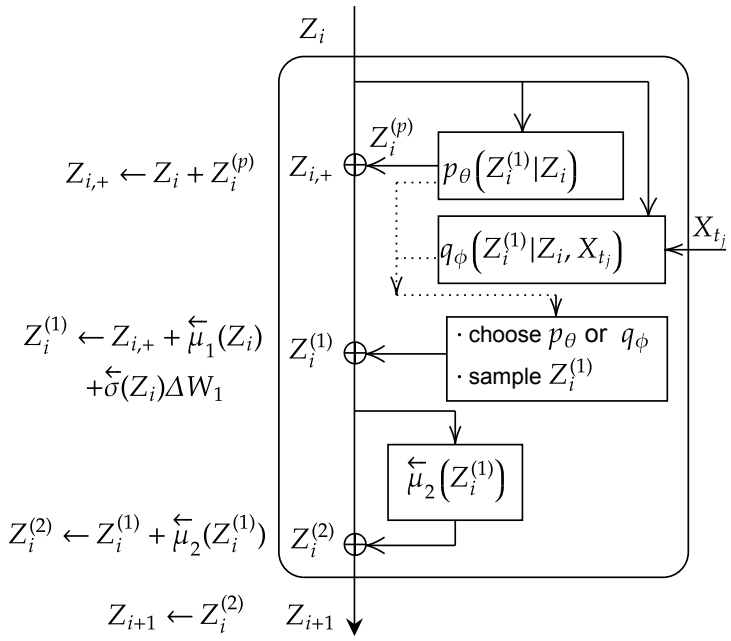


Figure 3.4: The VDVAE [34] cell is a type of two-step forward Euler discretisation of the continuous-time diffusion process in Eq. 3.4. See Fig. B.1 for similar schemas on NVAE [221] and Markovian HVAE [28, 207].

process, and its residual cells are a type of two-step forward Euler discretisation of this Stochastic Differential Equation (SDE).

To better understand Theorem 3.4, we visualise its residual cell structure of VDVAEs and the corresponding discretisation steps in Fig. 3.4, and together those of NVAEs and Markovian HVAEs in Appendix B.1.5, Fig. B.1. Note that this process is Markov and increasing in the Z_i variables. Similar processes have been empirically observed as efficient first-order approximates to higher-order chains, for example the memory state in LSTMs [93]. Further, VDVAEs and NVAEs are even claimed to be high-order chains (see Eqs. (2,3) in [34] and Eq. (1) in [221]), despite only approximating this with an accumulative process.

To show how VDVAEs impose the growth of the Z_t, we prove that the bottleneck component of VDVAE's U-Net enforces $Z_0 = 0$. This is done by identifying that the measure ν_0, which a VDVAE connects to the data ν_∞ via a multi-resolution bridge, is a point mass on the zero function. Consequently the backward pass

must grow from this, and the network learns this in a monotonic manner as we later confirm in our experiments (see §3.3.2).

Theorem 3.5. *Consider the SDE in Eq.* (3.4)*, trained through the ELBO in Eq.* (1.2)*. Let* $\tilde{\nu}_J$ *denote the data measure and* $\nu_0 = \delta_{\{0\}}$ *be the initial multi-resolution bridge measure imposed by VDVAE. If* $q_{\phi,j}$ *and* $p_{\theta,j}$ *are the densities of* $B_{\phi,1|j}\boldsymbol{F}_{J|1}\tilde{\nu}_J$ *and* $B_{\theta,1|j}\nu_0$ *respectively, then a VDVAE optimises the boundary condition* $\min_{\theta,\phi} KL(q_{\phi,0,1}||q_{\phi,0}p_{\theta,1})$*, where a double index indicates the joint distribution.*

Theorem 3.5 states that the VDVAE architecture forms multi-resolution bridge with the dynamics of Eq. (3.4), and connects our data distribution to the trivial measure on V_0: a Dirac mass at 0 as the pooling here cascades completely to V_0. From this insight, we can draw conclusions on instabilities and on parameter redundancies of this HVAE cell.

The imposed ν_0 is disastrously unstable as it enforces a data set, with potentially complicated topology to derive from a point-mass in U_{-j} at each $t = t_j$, and we observe the resulting sampling instability in our experiments in §3.3.3. We note that similar arguments are applicable in settings without a latent hierarchy imposed by a U-Net, see for instance [42]. The VDVAE architecture does, however, bolster this rate through the $Z_{i,+}^{(\sigma)}$ term, which is absent in NVAEs [221], in the discretisation steps of the residual cell. We empirically observe this controlled backward error in Fig. 3.6 [Right]. We refer to Fig. B.1 for a detailed comparison of HVAE cells and their corresponding discretisation of the coupled SDE in Eq. (3.4).

Moreover, the current form of VDVAEs is over-parameterised and not informed by this continuous-time formulation. The continuous time analogue of VDVAEs [34] in Theorem 3.4 has time dependent coefficients $\overleftarrow{\mu}_{t,1}$, $\overleftarrow{\mu}_{t,2}$, $\overleftarrow{\sigma}_t$. We hypothesise that the increasing diffusion process in Z_i implicitly encodes time. Hence, explicitly representing this in the model, for instance via ResNet blocks with independent parameterisations at every time step, is redundant, and a time-homogeneous model

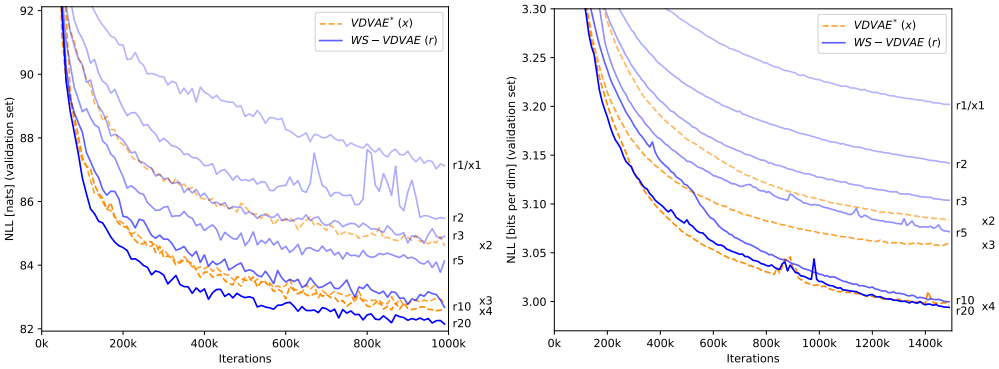


Figure 3.5: A small-scale study on parameter efficiency of HVAEs. We compare models with with 1,2,3 and 4 parameterised blocks per resolution ($\{x1, x2, x3, x4\}$) against models with a single parameterised block per resolution weight-shared $\{2, 3, 5, 10, 20\}$ times ($\{r2, r3, r5, r10, r20\}$). We report NLL ($\downarrow$) measured on the validation set of MNIST [left] and CIFAR10 [right]. NLL performance increases with more weight-sharing repetitions and surpasses models without weight-sharing but with more parameters.

(see Appendix B.1.6 for a precise formulation)—practically speaking, performing weight-sharing across time time steps/layers—has the same expressivity, but requires far fewer parameters than the state-of-the-art VDVAE. It is worth noting that such a time-homogeneous model would make the parameterisation of HVAEs more similar to the recently popular diffusion models [92, 206] which perform weight-sharing across all time steps.

3.3 Experiments

In the following we probe the theoretical understanding of HVAEs gained through our framework, demonstrating its utility in four experimental analyses: (a) Improving parameter efficiency in HVAEs, (b) Time representation in HVAEs and how they make use of it, (c) Sampling instabilities in HVAEs, and (d) Ablation studies.

We train HVAEs using VDVAE [34] as the basis model on five datasets: MNIST [133], CIFAR10 [127], two downsampled versions of ImageNet [36, 55], and CelebA [145], splitting each into a training, validation and test set (see Appendix B.4 for details). In general, reported numeric values refer to Negative Log-Likelihood (NLL) in nats

Table 3.1: A large-scale study of parameter efficiency in HVAEs. We compare our runs of VDVAE with original hyperparameters [34] (`VDVAE`*) against our weight-shared VDVAE (`WS-VDVAE`). While `WS-VDVAE`s have improved parameter efficiency by a factor of 2, they reach similar NLL as `VDVAE`* with the simple modification inspired by our framework (weight sharing). We note that a parameter count cannot be provided for VDM [120] as the code is not public and the manuscript does not specify it.

Dataset	Method	Type	#Params	NLL ↓
MNIST 28×28	`WS-VDVAE` (ours)	VAE	**232k**	≤ 79.98
	`VDVAE`* (ours)	VAE	339k	≤ 80.14
	NVAE [221]	VAE	33m	≤ 78.01
CIFAR10 32 × 32	`WS-VDVAE` (ours)	VAE	**25m**	≤ 2.88
	`WS-VDVAE` (ours)	VAE	39m	≤ 2.83
	`VDVAE`* (ours)	VAE	39m	≤ 2.87
	NVAE [221]	VAE	131m	≤ 2.91
	VDVAE [34]	VAE	39m	≤ 2.87
	VDM [120]	Diff	–	≤ 2.65
ImageNet 32 × 32	`WS-VDVAE` (ours)	VAE	**55m**	≤ 3.68
	`WS-VDVAE` (ours)	VAE	85m	≤ 3.65
	`VDVAE`* (ours)	VAE	119m	≤ 3.67
	NVAE [221]	VAE	268m	≤ 3.92
	VDVAE [34]	VAE	119m	≤ 3.80
	VDM [120]	Diff	–	≤ 3.72
CelebA 64 × 64	`WS-VDVAE` (ours)	VAE	**75m**	≤ 2.02
	`VDVAE`* (ours)	VAE	125m	≤ 2.02
	NVAE [221]	VAE	153m	≤ 2.03

(MNIST) or bits per dim (all other datasets) on the test set at model convergence, if not stated otherwise. We note that performance on the validation and test set have similar trends in general. An optional *gradient checkpointing* implementation to trade in GPU memory for compute is discussed in Appendix B.6. Appendices B.6 and B.7 define the HVAE models we train, i.e. $p_\theta(\mathbf{z}_L), p_\theta(\mathbf{z}_l|\mathbf{z}_{>l}), q_\phi(\mathbf{z}_L|\mathbf{x}), q_\phi(\mathbf{z}_l|\mathbf{z}_{>l}, \mathbf{x})$ and $p_\theta(\mathbf{x}|\vec{\mathbf{z}})$, and present additional experimental details and results. We provide our PyTorch code base at https://github.com/FabianFalck/unet-vdvae (see Appendix B.3 for details).

3.3.1 "More from less": Improving parameter efficiency in HVAEs

In §3.2.3, we hypothesised that a time-homogeneous model has the same expressivity as a model with time-dependent coefficients, yet uses much less parameters. We start demonstrating this effect by weight-sharing ResNet blocks across time on a small scale. In Fig. 3.5, we train HVAEs on MNIST and CIFAR10 with $\{1, 2, 3, 4\}$ ResNet blocks (referred to as {`x1`, `x2`, `x3`, `x4`}) in each resolution with spatial dimensions $\{32^2, 16^2, 8^2, 4^2, 1^2\}$ (`VDVAE`*), and compare their performance when weight-sharing a single parameterised block per resolution $\{2, 3, 5, 10, 20\}$ times (referred to as {`r2`,`r3`,`r5`,`r10`,`r20`}; `WS-VDVAE`), excluding projection and embedding blocks. As hypothesised by our framework, yet very surprising in HVAEs, NLL after 1m iterations measured on the validation set gradually increases the more often blocks are repeated even though all weight-sharing models have an identical parameter count to the $x1$ model (MNIST: 107k, CIFAR10: 8.7m). Furthermore, the weight-sharing models often outperform or reach equal NLLs compared to `x2`, `x3`, `x4`, all of which have more parameters (MNIST: 140k; 173k; 206k. CIFAR10: 13.0m; 17.3m; 21.6m), yet fewer activations, latent variables, and number of timesteps at which the coupled SDE in Eq. (3.4) is discretised.

We now scale these findings up to large-scale hyperparameter configurations. We train VDVAE closely following the state-of-the-art hyperparameter configurations in [34], specifically with the same number of parameterised blocks and without weight-sharing (`VDVAE`*), and compare them against models with weight-sharing (`WS-VDVAE`) and fewer parameters, i.e. fewer parameterised blocks, in Table 3.1. On all four datasets, the weight-shared models achieve similar NLLs with fewer parameters compared to their counterparts without weight-sharing: We use 32%, 36%, 54%, and 40% less parameters on the four datasets reported in Table 3.1, respectively. For the larger runs, weight-sharing has diminishing returns on NLL as these already have many discretisation steps. To the best of our knowledge, our models achieve a new state-of-the-art performance in terms of NLL compared to any HVAE on CIFAR10,

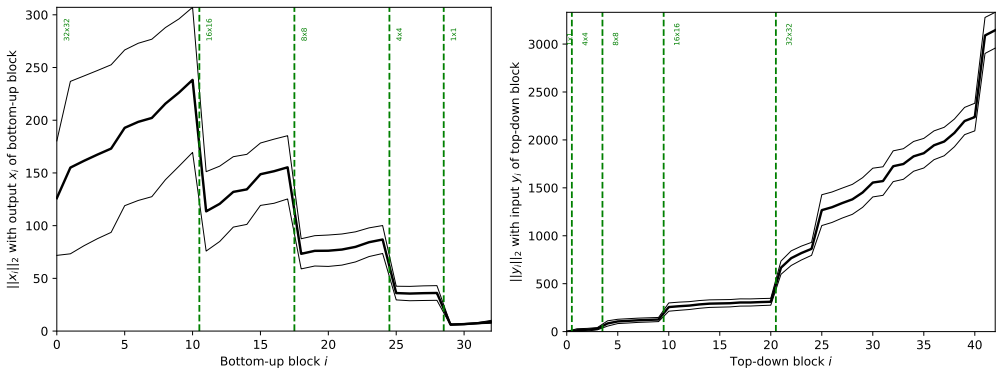


Figure 3.6: HVAEs secretly represent a notion of time: We measure the L_2-norm of the residual state for the [Left] forward/bottom-up pass and the [Right] backward/top-down pass over 10 batches with 100 data points each. In both plots, the thick, central line refers to the average and the thin, outer lines refer to ± 2 standard deviations.

ImageNet32 and CelebA. Furthermore, our `WS-VDVAE` models have stochastic depths of 57, 105, 235, 125, respectively, the highest ever trained. In spite of these results, it is worth noting that current HVAEs, and VDVAE in particular remains notoriously unstable to train, partly due to the instabilities identified in Theorem 3.5, and finding the right hyperparameters helps, but cannot solve this.

3.3.2 HVAEs secretly represent time and make use of it

In §3.3.1, we showed how we can exploit insight on HVAEs through our framework to make HVAEs more parameter efficient. We now want to explain and understand this behavior further. In Fig. 3.6, we measure $\|Z_i\|_2$, the L_2-norm of the residual state at every backward/top-down block with index i, over several batches for models trained on MNIST (see Appendix B.7.2 for the corresponding figure of the forward/bottom-up pass, and similar results on CIFAR10 and ImageNet32). On average, we experience an increase in the state norm across time in every resolution, interleaved by discontinuous 'jumps' at the resolution transitions (projection or embedding) where the dimension of the residual state changes. This supports our claim in §3.2 that HVAEs discretise multi-resolution diffusion processes which are increasing in the Z_i variables, and hence learn to represent a notion of time in their residual state.

Table 3.2: NLL of HVAEs with and without normalisation of the residual state Z_i. The symbol ✗ indicates deteriorated training due to numerical instabilities.

Residual state	NLL
MNIST	
Normalised (✗)	≤ 464.68
Non-normalised	≤ 81.69
CIFAR10	
Normalised (✗)	≤ 6.80
Non-normalised	≤ 2.93
ImageNet	
Normalised	≤ 6.76
Non-normalised	≤ 3.68

It is now straightforward to ask how HVAEs benefit from this time representation during training: As we show in Table 3.2, when normalising the state by its norm at every forward and backward block during training, i.e. forcing a "flat line" in Fig. 3.6, learning deteriorates after a short while, resulting in poor NLL results compared to the runs with a regular, non-normalised residual state. This evidence confirms our earlier stated hypothesis: The time representation in ResNet-based HVAEs encodes information which recent HVAEs heavily rely on during learning.

3.3.3 Sampling instabilities in HVAEs

High fidelity unconditional samples of faces, e.g. from models trained on CelebA, cover the front pages of state-of-the-art HVAE papers [34, 221]. Here, we question whether face datasets are an appropriate benchmark for HVAEs. In Theorem 3.5, we identified the aforementioned state-of-the-art HVAEs as flow from a point mass, hypothesising instabilities during sampling. And indeed, when sampling from our trained `VDVAE`* with state-of-the-art configurations, we observe high fidelity and diversity samples on MNIST and CelebA, but unrecognisable, yet diverse samples on CIFAR10, ImageNet32 and ImageNet64, in spite of state-of-the-art test set NLLs (see Fig. 3.7 and Appendix B.7.3). We argue that MNIST and CelebA, i.e. numbers

CIFAR10 ($t = 0.9$) ImageNet32 ($t = 1$) ImageNet64 ($t = 0.9$) MNIST ($t = 0.9$) CelebA ($t = 0.5$)

Figure 3.7: Unconditional samples (not cherry-picked) of `VDVAE`*. While samples on MNIST and CelebA demonstrate high fidelity and diversity, samples on CIFAR10, ImageNet32 and ImageNet64 are diverse, but are unrecognisable, demonstrating the instabilities identified by Theorem 3.1. Temperatures t are tuned for maximum fidelity.

and faces, have a more uni-modal nature, and are in this sense easier to learn for a discretised multi-resolution process flowing to a point mass, which is uni-modal, than the other "in-the-wild", multi-modal datasets. Trying to approximate the latter with the, in this case unsuitable, HVAE model leads to the sampling instabilities observed.

3.3.4 Ablation studies

We conducted several ablation studies which support our experimental results and further probe our multi-resolution framework for HVAEs. In this section we note key findings—a detailed account of all ablations can be found in Appendix B.7.4. In particular, we find that the number of latent variables, which correlates with stochastic depth, does not explain the performance observed in §3.3.1, supporting our claims. We further show that Fourier features do not provide a performance gain in HVAEs, in contrast to state-of-the-art diffusion models, where they can significantly improve performance [120]. This is consistent with our framework's finding that a U-Net architecture with pooling is already forced to learn a Haar wavelet basis representation of the data, hence introducing another basis does not add value. We also demonstrate that residual cells are crucial for the performance of HVAEs as they are able to approximate the dynamics of a diffusion process and impose an SDE structure into the model, empirically compare a multi-resolution bridge to a single-resolution model, and investigate synchronous vs. asynchronous processing in time between the forward and backward pass.

3.4 Related work

U-Nets. A U-Net [197] is an autoencoding architecture with multiple resolutions where skip connections enable information to pass between matched layers on opposite sides of the autoencoder's bottleneck. These connections also smooth out the network's loss landscape [136]. In the literature, U-Nets tend to be convolutional, and a wide range of different approaches have been used for up-sampling and down-sampling between resolutions, with many using average pooling for the down-sampling operation [56, 92, 120, 208, 209]. In this work, we focus on U-Nets as operators on measures interleaved by average pooling as the down-sampling operation (and a corresponding inclusion operation for up-sampling), and we formally characterise U-Nets in Section 3.2.1 and Appendix B.2.2. Prior to our work, the dimensionality-reducing bottleneck structure of U-Nets was widely acknowledged as being useful, however it was unclear what regularising properties a U-Net imposes. We provided these in §3.2.

HVAEs. The evolution of HVAEs can be seen as a quest for a parameterisation with more expressiveness than single-latent-layer VAEs [121], while achieving stable training dynamics that avoid common issues such as posterior collapse [24, 207] or exploding gradients. Early HVAEs such as LVAE condition each latent variable directly on only the previous one by taking samples forward [28, 207]. Such VAEs suffer from stability issues even for very small stochastic depths. *Nouveau VAEs (NVAE)* [221] and *Very Deep VAEs (VDVAE)* [34] combine the improvements of several earlier HVAE models (see Section 1.1.3 for details), while scaling up to larger stochastic depths. Both use ResNet-based backbones, sharing parameters between the generative and recognition parts of the model. VDVAE is the considerably simpler approach, in particular avoiding common tricks such as a warm-up deterministic autoencoder training phase or data-specific initialisation. VDVAE achieves a stochastic depth of up to 78, improving performance with more ResNet blocks. Worth noting is that while LVAE and NVAE use convolutions with appropriately

chosen stride to jump between resolutions, VDVAE use average pooling. In all HVAEs to date, a theoretical underpinning which explains architectural choices, for instance the choice of residual cell, is missing, and we provided this in Section §3.2.3.

3.5 Conclusion

In this work, we introduced a multi-resolution framework for U-Nets. We provided theoretical results which uncover the regularisation property of the U-Nets bottleneck architecture with average pooling as implicitly learning a Haar wavelet representation of the data. We applied our framework to HVAEs, identifying them as multi-resolution diffusion processes flowing to a point mass. We characterised their backward cell as a type of two-step forward Euler discretisations, providing an alternative to score-matching to approximate a continuous-time diffusion process [52, 209], and observed parameter redundancies and instabilities. We verified the latter theoretical insights in both small- and large-scale experiments, and in doing so trained the deepest ever HVAEs. We explained these results by showing that HVAEs learn a representation of time and performed extensive ablation studies.

An important limitation is that the proven regularisation property of U-Nets is limited to using average pooling as the down-sampling operation. Another limitation is that we only applied our framework to HVAEs, though it is possible to apply it to other model classes. It could also be argued that the lack of exhaustive hyperparameter optimisation performed is a limitation of the work as it may be possible to obtain improved results. We demonstrate, however, that simply adding weight-sharing while using the hyperparameter settings given in the original VDVAE paper [34] leads to state-of-the-art performance with improved parameter efficiency, and hence view it as a strength of our results.

Acknowledgments and Disclosure of Funding

Fabian Falck acknowledges the receipt of studentship awards from the Health Data Research UK-The Alan Turing Institute Wellcome PhD Programme in Health Data Science (Grant Ref: 218529/Z/19/Z), and the Enrichment Scheme of The Alan Turing Institute under the EPSRC Grant EP/N510129/1. Chris Williams acknowledges support from the Defence Science and Technology (DST) Group and from a ESPRC DTP Studentship. Dominic Danks is supported by a Doctoral Studentship from The Alan Turing Institute under the EPSRC Grant EP/N510129/1. Christopher Yau is funded by a UKRI Turing AI Fellowship (Ref: EP/V023233/1). Chris Holmes acknowledges support from the Medical Research Council Programme Leaders award MC_UP_A390_1107, The Alan Turing Institute, Health Data Research, U.K., and the U.K. Engineering and Physical Sciences Research Council through the Bayes4Health programme grant. Arnaud Doucet acknowledges support of the UK Defence Science and Technology Laboratory (Dstl) and EPSRC grant EP/R013616/1. This is part of the collaboration between US DOD, UK MOD and UK EPSRC under the Multidisciplinary University Research Initiative. Arnaud Doucet also acknowledges support from the EPSRC grant EP/R034710/1. Matthew Willetts is grateful for the support of UCL Computer Science and The Alan Turing Institute.

The authors report no competing interests.

The three compute clusters used in this work were provided by the Alan Turing Institute, the Oxford Biomedical Research Computing (BMRC) facility, and the Baskerville Tier 2 HPC service (https://www.baskerville.ac.uk/) which we detail in the following. First, this research was supported in part through computational resources provided by The Alan Turing Institute under EPSRC grant EP/N510129/1 and with the help of a generous gift from Microsoft Corporation. Second, we used the Oxford BMRC facility, a joint development between the Wellcome Centre for Human Genetics and the Big Data Institute supported by Health Data Research

UK and the NIHR Oxford Biomedical Research Centre. The views expressed are those of the author(s) and not necessarily those of the NHS, the NIHR or the Department of Health. Third, Baskerville was funded by the EPSRC and UKRI through the World Class Labs scheme (EP/T022221/1) and the Digital Research Infrastructure programme (EP/W032244/1) and is operated by Advanced Research Computing at the University of Birmingham.

We thank Tomas Lazauskas, Jim Madge and Oscar Giles from the Alan Turing Institute's Research Engineering team for their help and support. We thank Adam Huffman, Jonathan Diprose, Geoffrey Ferrari and Colin Freeman from the Biomedical Research Computing team at the University of Oxford for their help and support. We thank Haoting Zhang (University of Cambridge) for valuable comments on the implementation; Huiyu Wang (Johns Hopkins University) for a useful discussion on gradient checkpointing; and Ruining Li and Hanwen Zhu (University of Oxford) for kindly proofreading the manuscript.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	A Multi-Resolution Framework for U-Nets with Applications to Hierarchical VAEs
Publication Status	x Published □ Accepted for Publication □ Submitted for Publication □ Unpublished and unsubmitted work written in a manuscript style
Publication Details	A Multi-Resolution Framework for U-Nets with Applications to Hierarchical VAEs. Fabian Falck*, Christopher Williams*, Dominic Danks, George Deligiannidis, Christopher Yau, Chris Holmes, Arnaud Doucet, Matthew Willetts. Advances in Neural Information Processing Systems (NeurIPS) 2022 (oral). * indicates equal contribution.

Student Confirmation

Student Name:	Fabian Falck
Contribution to the Paper	I follow the CRediT contributor role taxonomy (Brand et al., 2015). Fabian Falck: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing - Original Draft, Writing - Review & Editing, Visualization Christopher Williams: Conceptualization, Methodology, Formal analysis, Investigation, Writing - Original Draft, Writing - Review & Editing, Visualization Dominic Danks: Software, Investigation, Writing - Review & Editing George Deligiannidis: Writing - Review & Editing, Supervision, Funding acquisition Christopher Yau: Writing - Review & Editing, Supervision, Funding acquisition Chris Holmes: Methodology, Resources, Writing - Review & Editing, Supervision, Project administration, Funding acquisition Arnaud Doucet: Methodology, Writing - Review & Editing, Supervision, Project administration, Funding acquisition Matthew Willetts: Conceptualization, Methodology, Resources, Writing - Review & Editing, Supervision, Project administration

Signature	Fabian Falck	Date	September 29, 2024

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof. Arnaud Doucet		
Supervisor comments I agree with the description of the contributions.		
 Signature	Date	25/11/2024

This completed form should be included in the thesis, at the end of the relevant chapter.

4

A Unified Framework for U-Net Design and Analysis

Abstract

U-Nets are a go-to neural architecture across numerous tasks for continuous signals on a square such as images and Partial Differential Equations (PDE), however their design and architecture is understudied. In this paper, we provide a framework for designing and analysing general U-Net architectures. We present theoretical results which characterise the role of the encoder and decoder in a U-Net, their high-resolution scaling limits and their conjugacy to ResNets via preconditioning. We propose Multi-ResNets, U-Nets with a simplified, wavelet-based encoder without learnable parameters. Further, we show how to design novel U-Net architectures which encode function constraints, natural bases, or the geometry of the data. In diffusion models, our framework enables us to identify that high-frequency information is dominated by noise exponentially faster, and show how U-Nets with average pooling exploit this. In our experiments, we demonstrate how Multi-ResNets achieve competitive and often superior performance compared to classical U-Nets in image segmentation, PDE surrogate modelling, and generative modelling with diffusion models. Our U-Net framework paves the way to study the theoretical properties of U-Nets and design natural, scalable neural architectures for a multitude of problems beyond the square.

Contents

4.1 Introduction

U-Nets (see Figure 4.1) are a central architecture in deep learning for continuous signals. Across many tasks as diverse as image segmentation [132, 175, 197, 204, 263], Partial Differential Equation (PDE) surrogate modelling [79, 211] and score-based diffusion models [52, 92, 172, 196, 208], U-Nets are a go-to architecture yielding state-of-the-art performance. In spite of their enormous success, a framework for U-Nets which characterises for instance the specific role of the encoder and decoder in a U-Net or which spaces these operate on is lacking. In this work, we provide such a framework for U-Nets. This allows us to design U-Nets for data beyond a square domain, and enable us to incorporate prior knowledge about a problem, for instance a natural basis, functional constraints, or knowledge about its topology, into the neural architecture.

The importance of preconditioning. We begin by illustrating the importance of the core design principle of U-Nets: *preconditioning*. Preconditioning informally means that initialising an optimisation problem with a 'good' solution greatly benefits learning [5, 6]. Consider a synthetic example using ResNets [87] which are natural in the context of U-Nets as we will show in §4.2.3: we are interested in learning a ground-truth mapping $w : V \mapsto W$ and $w(v) = v^2$ over $V = [-1, 1]$ and $W = \mathbb{R}$ using a ResNet $R^{\text{res}}(v) = R^{\text{pre}}(v) + R(v)$ where $R^{\text{pre}}, R : V \mapsto W$. In Figure 4.2 [Left] we learn a standard ResNet with $R^{\text{pre}}(v) = v$ on a grid of values from $V \times W$, i.e. with inputs $v_i \in V$ and regression labels $w_i = w(v_i) = v_i^2$. In contrast, we train a ResNet with $R^{\text{pre}}(v) = |v|$ [Right] with the same number of parameters and iterations. Both networks have been purposely designed to be weakly expressive (see Appendix C.2.5 for details). The standard ResNet [Left] makes a poor approximation of the function, whilst the other ResNet's [Right] approximation is nearly perfect. This is because $R^{\text{pre}}(v) = |v|$ is a 'good' initial guess or *preconditioner* for $w(v) = v^2$, but $R^{\text{pre}}(v) = v$ is a 'bad' one. This shows

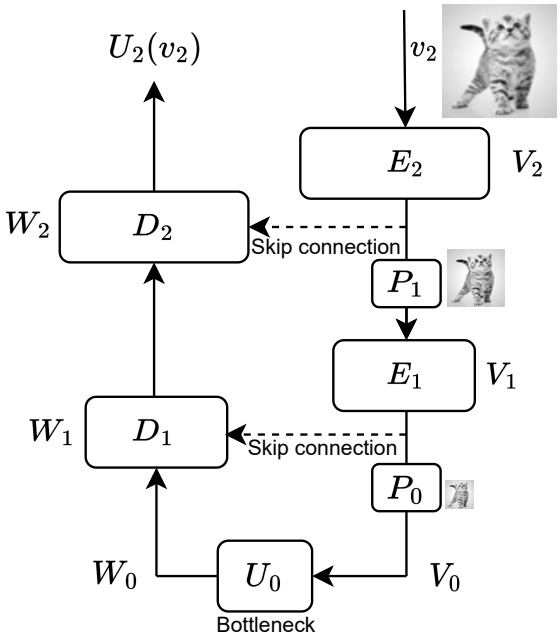


Figure 4.1: A resolution 2 U-Net (Def. 4.1). If $E_i = \mathrm{Id}_{V_i}$, this is a Multi-ResNet (see Def. 4.3).

the importance of encoding good preconditioning into a neural network architecture and motivates us studying how preconditioning is used in U-Net design.

In this paper, we propose a mathematical framework for designing and analysing general U-Net architectures. We begin with a comprehensive definition of a U-Net which characterises its components and identifies its self-similarity structure which is established via preconditioning. Our theoretical results delineate the role of the encoder and decoder and identify the subspaces they operate on. We then focus on ResNets as natural building blocks for U-Nets that enable flexible preconditioning from lower-resolutions inputs. Our U-Net framework paves the way to designing U-Nets which can model distributions over complicated geometries beyond the square, for instance CW-complexes or manifolds, or diffusions on the sphere [51], without any changes to the diffusion model itself (see Appendix C.1). It allows us to enforce problem-specific constraints through the architecture, such as boundary conditions of a PDE or a natural basis for a problem. We also analyse why U-Nets with average pooling are a natural inductive bias in diffusion models.

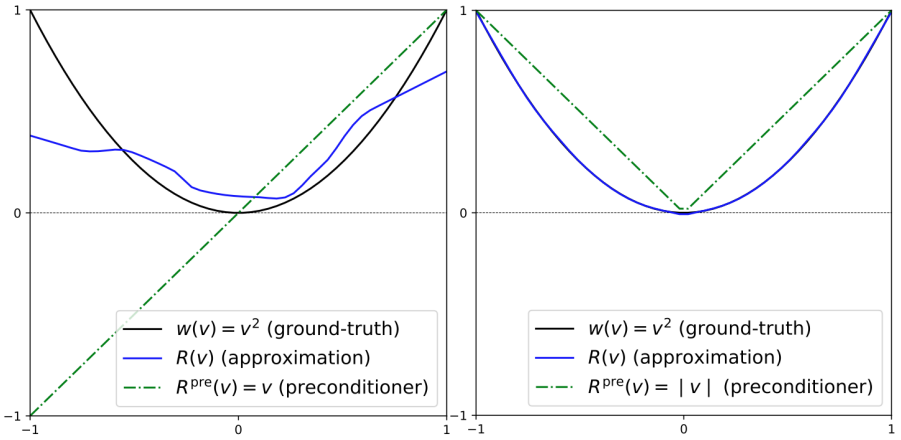


Figure 4.2: The importance of *preconditioning*.

More specifically, our *contributions* are as follows: (a) We provide the first rigorous definition of U-Nets, which enables us to identify their self-similarity structure, high-resolution scaling limits, and conjugacy to ResNets via preconditioning. (b) We present Multi-ResNets, a novel class of U-Nets over a hierarchy of orthogonal wavelet spaces of $L^2(\mathbb{X})$, for compact domain $\mathbb{X}$, with no learnable parameters in its encoder. In our experiments, Multi-ResNets yield competitive and often superior results when compared to a classical U-Net in PDE modelling, image segmentation, and generative modelling with diffusion models. We further show how to encode problem-specific information into a U-Net. In particular, we design U-Nets incorporating a natural basis for a problem which enforces boundary conditions on the elliptic PDE problem, and design and demonstrate proof-of-concept experiments for U-Nets with a Haar wavelet basis over a triangular domain. (c) In the context of diffusion models, we analyse the forward process in a Haar wavelet basis and identify how high-frequency information is dominated by noise exponentially faster than lower-frequency terms. We show how U-Nets with average pooling exploit this observation, explaining their go-to usage.

4.2 U-Nets: Neural networks via subspace preconditioning

The goal of this section is to develop a mathematical framework for U-Nets which introduces the fundamental principles that underpin its architecture and enables us to design general U-Net architectures. All theoretical results are proven in Appendix C.1. We commence by defining the U-Net.

4.2.1 Anatomy of a U-Net

Definition 4.1. U-Net. Let V and W be measurable spaces. A *U-Net* $\mathcal{U} = (\mathcal{V}, \mathcal{W}, \mathcal{E}, \mathcal{D}, \mathcal{P}, U_0)$ comprises six components:

1. *Encoder subspaces:* $\mathcal{V} = (V_i)_{i=0}^{\infty}$ are nested subsets of V such that $\lim_{i\to\infty} V_i = V$.
2. *Decoder subspaces:* $\mathcal{W} = (W_i)_{i=0}^{\infty}$ are nested subsets of W such that $\lim_{i\to\infty} W_i = W$.
3. *Encoder operators:* $\mathcal{E} = (E_i)_{i=1}^{\infty}$ where $E_i : V_i \mapsto V_i$ denoted $E_i(v_i) = \tilde{v}_i$.
4. *Decoders operators:* $\mathcal{D} = (D_i)_{i=1}^{\infty}$ where $D_i : W_{i-1} \times V_i \mapsto W_i$ at resolution i denoted $D_i(w_{i-1}|\tilde{v}_i)$. The $\tilde{v}_i$ component is called the *skip connection*.
5. *Projection operators:* $\mathcal{P} = (P_i)_{i=0}^{\infty}$, where $P_i : V \mapsto V_i$, such that $P_i(v_i) = v_i$ for $v_i \in V_i$.
6. *Bottleneck*: U_0 is the mapping $U_0 : V_0 \mapsto W_0$, enabling a compressed representation of the input.

The *U-Net of resolution* i is the mapping $U_i : V_i \mapsto W_i$ defined through the recursion (see Figure 4.3):

$$U_i(v_i) = D_i(U_{i-1}(P_{i-1}(\tilde{v}_i))|\tilde{v}_i), \qquad i = 1, 2, \ldots. \tag{4.1}$$

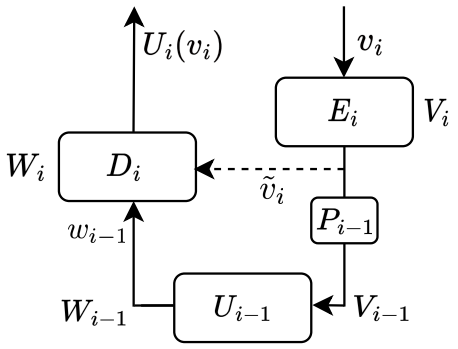


Figure 4.3: Recursive structure of a U-Net.

We illustrate the Definition of a U-Net in Figure 4.1. Definition 4.1 includes a wide array of commonly used U-Net architectures, from the seminal U-Net [197], through to modern adaptations used in large-scale diffusion models [52, 92, 172, 196, 208], operator learning U-Nets for PDE modelling [79, 211], and custom designs on the sphere or graphs [71, 258]. Our framework also comprises models with multiple channels (for instance RGB in images) by choosing V_i and W_i for a given resolution i to be direct sums $V_i = \bigoplus_{k=1}^{M} V_{i,k}$ and $W_i = \bigoplus_{k=1}^{M} W_{i,k}$ for M channels when necessary[1]. Remarkably, despite their widespread use, to the best of our knowledge, our work presents the first formal definition of a U-Net. Definition 4.1 not only expands the scope of U-Nets beyond problems confined to squared domains, but also naturally incorporates problem-specific information such as a natural basis, boundary conditions or topological structure as we will show in §4.3.2 and 4.3.3. This paves the way for designing inherently scalable neural network architectures capable of handling complicated geometries, for instance manifolds or CW-complexes. In the remainder of the section, we discuss and characterise the components of a U-Net.

Encoder and decoder subspaces. We begin with the spaces in V and W which a U-Net acts on. Current literature views U-Nets as learnable mappings between input and output tensors. In contrast, this work views U-Nets and their encoder and decoder as operators on spaces that must be chosen to suit our task. In order to perform computations in practice, we must restrict ourselves to subspaces V_i and W_i

[1]Implementation wise, this corresponds to concatenating the representations across the channel dimension.

of the potentially infinite-dimensional spaces V and W. For instance, if our U-Net is modelling a mapping between images, which can be viewed as bounded functions over a squared domain $\mathbb{X} = [0,1]^2$, it is convenient to choose V and W as subspaces of $L^2(\mathbb{X})$, the space of square-integrable functions [62, 66] (see Appendix C.3). Here, a data point w_i in the decoder space W_i is represented by the coefficients $c_{i,j}$ where $w_i = \sum_j c_{i,j} e_{i,j}$ and $\{e_{i,j}\}_j$ is a (potentially orthogonal) basis for W_i. We will consider precisely this case in §4.3.1. The projected images on the subspace V_i, W_i are still functions, but piece-wise constant ones, and we store the values these functions obtain as 'pixel' tensors in our computer [66] (see Figure C.14 in Appendix C.1).

Role of encoder, decoder, projection operators. In spite of their seemingly symmetric nature and in contrast to common understanding, the roles of the encoder and decoder in a U-Net are fundamentally different from each other. The decoder on resolution i learns the transition from W_{i-1} to W_i while incorporating information from the encoder on V_i via the skip connection. The encoder E_i can be viewed as a change of basis mapping on the input space V_i at resolution i and is not directly tied to the approximation the decoder makes on W_i. This learned change of basis facilitates the decoder's approximation on W_i. In §4.3.1, we will further extend our understanding of the encoder and discuss its implications for designing U-Nets. The projection operators serve to extract a compressed input. They are selected to suit task-specific needs, such as pooling operations (e.g. average pooling, max pooling) in the context of images, or orthogonal projections if V is a Hilbert space. Note that there is no embedding operator[2], the operator facilitating the transition to a higher-resolution space, explicitly defined as it is invariably the natural inclusion of W_{i-1} into W_i.

Self-similarity of U-Nets via preconditioning. The key design principle of a U-Net is preconditioning. The U-Net of resolution $i-1$, U_{i-1} makes an approximation on W_{i-1} which is input of and preconditions U_i. Preconditioning

[2]In practice, if we for instance learn a transposed convolution operation, it can be equivalently expressed as a standard convolution operation composed with the natural inclusion operation.

facilitates the transition from W_{i-1} to W_i for the decoder. In the encoder, the output of $P_{i-1}E_i$ is the input of U_{i-1}. When our underlying geometry is refinable (such as a square), we may use a refinable set of basis functions. In the standard case of a U-Net with average pooling on a square domain, our underlying set of basis functions are (Haar) wavelets (see §4.2.3) – refinable basis functions defined on a refinable geometry. This preconditioned design of U-Nets reveals a self-similarity structure (see Figure 4.3) inherent to the U-Net when the underlying space has a refinable geometry. This enables both an efficient multi-resolution approximation of U-Nets [66] and makes them modular, as a U-Net on resolution $i-1$ is a coarse approximation for a U-Net on resolution i. We formalise this notion of preconditioning in Proposition 4.1 in §4.2.3.

4.2.2 High-resolution scaling behavior of U-Nets

Given equation (4.1), it is clear that the expressiveness of a U-Net $\mathcal{U}$ is governed by the expressiveness of its decoder operators $\mathcal{D}$. If each D_i is a universal approximator [96], then the corresponding U-Nets U_i likewise have this property. Assuming we can represent any mapping $U_i : V_i \mapsto W_i$ as a U-Net, our goal now is to comprehend the role of increasing the resolution in the design of $\mathcal{U}$, and to discern whether any function $U : V \to W$ can be represented as a high-resolution limit $\lim_{i\to\infty} U_i$ of a U-Net. We will explore this question in the context of regression problems of increasing resolution.

To obtain a tractable answer, we focus on choosing W as a *Hilbert space*, that is W equipped with an inner product. This allows us to define $\mathcal{W}$ as an increasing sequence of *orthogonal* subspaces of W. Possible candidates for orthogonal bases include certain Fourier frequencies, wavelets (of a given order) or radial basis functions. The question of which basis is optimal depends on our problem and data at hand: some problems may be hard in one basis, but easy in another. In §4.4, Haar wavelets are a convenient choice. Let us define $\mathcal{S}$ as infinite resolution

data in $V \times W$ and $\mathcal{S}_i$ as the finite resolution projection in $V_i \times W_i$ comprising of $(v_i, w_i) = (P_i(v), Q_i(w))$ for each $(v, w) \in \mathcal{S}$. Here, P_i is the U-Net projection onto V_i, and $Q_i : W \mapsto W_i$ is the orthogonal projection onto W_i. Assume U_i^* and U^* are solutions to the finite and infinite resolution regression problems $\min_{U_i \in \mathcal{F}_i} \mathcal{L}_i^2(U_i|\mathcal{S}_i)$ and $\min_{U \in \mathcal{F}} \mathcal{L}^2(U|\mathcal{S})$ respectively, where $\mathcal{F}_i$ and $\mathcal{F}$ represent the sets of measurable functions mapping $V_i \mapsto W_i$ and $V \mapsto W$. Let $\mathcal{L}_i^2$ and $\mathcal{L}^2$ denote the L^2 losses:

$$\mathcal{L}_i^2(U_i|\mathcal{S}_i) \coloneqq \frac{1}{|\mathcal{S}_i|} \sum_{(w_i, v_i) \in \mathcal{S}_i} \|w_i - U_i(v_i)\|^2, \quad \mathcal{L}^2(U|\mathcal{S}) \coloneqq \frac{1}{|\mathcal{S}|} \sum_{(w,v) \in \mathcal{S}} \|w - U(v)\|^2.$$

The following result analyses the relationship between U_i^* and U^* as $i \to \infty$ where $\mathcal{L}_{i|j}^2$ and $\mathcal{L}_{|j}^2$ are the losses above conditioned on resolution j (see Appendix C.1).

Theorem 4.1. *Suppose U_i^* and U^* are solutions of the L^2 regression problems above. Then, $\mathcal{L}_{i|j}^2(U_i^*) \leq \mathcal{L}_{|j}^2(U^*)$ with equality as $i \to \infty$. Further, if $Q_i U^*$ is V_i-measurable, then $U_i^* = Q_i U^*$ minimises $\mathcal{L}_i^2$.*

Theorem 4.1 states that solutions of the finite resolution regression problem converge to solutions of the infinite resolution problem. It also informs us how to choose V_i relative to W_i. If we have a $\mathcal{V}$ where for the decoders on $\mathcal{W}$, the W_i component of U^*, $Q_i U^*$, relies solely on the input up to resolution i, then the prediction from the infinite-dimensional U-Net projected to W_i can be made by the U-Net of resolution i. The optimal choice of V_i must be expressive enough to encode the information necessary to learn the W_i component of U^*. This suggests that if V_i lacks expressiveness, we risk efficiency in learning the optimal value of U_i^*. However, if V_i is too expressive, no additional information is gained, and we waste computational effort. Therefore, we should choose V_i to encode the necessary information for learning information on resolution i. For example, when modelling images, if we are interested in low-resolution features on W_i, high-resolution information is extraneous, as we will further explore in §4.4 in the context of diffusion models.

4.2.3 U-Nets are conjugate to ResNets

Next, our goal is to understand why ResNets are a natural choice in U-Nets. We will uncover a conjugacy between U-Nets and ResNets. We begin by formalising ResNets in the context of U-Nets.

Definition 4.2. ResNet, Residual U-Nets. Given a measurable space X and a vector space Y, a mapping $R : X \to Y$ is defined as a *ResNet* preconditioned on $R^{\text{pre}} : X \to Y$ if $R(x) = R^{\text{pre}}(x) + R^{\text{res}}(x)$, where $R^{\text{res}}(x) = R(x) - R^{\text{pre}}(x)$ is the *residual* of R. A *Residual U-Net* is a U-Net $\mathcal{U}$ where $\mathcal{W}, \mathcal{V}$ are sequences of vector spaces, and the encoder and decoder operators $\mathcal{E}, \mathcal{D}$ are ResNets preconditioned on $E_i^{\text{pre}}(v_i) = v_i$ and $D_i^{\text{pre}}(w_{i-1}|v_i) = w_{i-1}$, respectively.

A preconditioner initialises a ResNet, then the ResNet learns the residual relative to it. The difficulty of training a ResNet scales with the deviation from the preconditioner, as we saw in our synthetic experiment in §4.1. In U-Nets, ResNets commonly serve as encoder and decoder operators. In encoders, preconditioning on the identity on V_i allow the residual encoder E_i^{res} to learn a change of basis for V_i, which we will discuss in more detail in §4.3.1. In decoders, preconditioning on the identity on the lower resolution subspace W_{i-1} allows the residual decoder D_i^{res} to learn from the lower resolution and the skip connection. Importantly, ResNets can compose to form new ResNets, which, combined with the recursion (4.1), implies that residual U-Nets are conjugate to ResNets.

Proposition 4.1. If $\mathcal{U}$ is a residual U-Net, then U_i is a ResNet preconditioned on $U_i^{\text{pre}}(v_i) = U_{i-1}(\tilde{v}_{i-1})$, where $\tilde{v}_{i-1} = P_{i-1}(E_i(v_i))$.

Proposition 4.1 states that a U-Net at resolution i is a ResNet preconditioned on a U-Net of lower resolution. This suggests that U_i^{res} learns the information arising from the resolution increase. We will discuss the specific case $E_i = \text{Id}_{V_i}$, and $U^{\text{pre}}(v_i) = U_{i-1}(P_{i-1}(v_i))$ in §4.3.1. Proposition 4.1 also enables us to interpret

$\lim_{i\to\infty} U_i$ from §4.2.2 as a ResNet's 'high-resolution' scaling limit. This is a new scaling regime for ResNets, different to time scaled Neural ODEs [32], and warrants further exploration in future work. Finally, we provide an example of the common *Residual U-Net*.

Example 1. Haar Wavelet Residual U-Net. A *Haar wavelet residual U-Net* $\mathcal{U}$ is a residual U-Net where: $V = W = L^2(\mathbb{X})$, $V_i = W_i$ are multi-resolution Haar wavelet spaces (see Appendix C.3.3), and $P_i = \text{proj}_{V_i}$ is the orthogonal projection.

We design Example 1 with images in mind, noting that similar data over a squared domain such as PDE (see §4.5) are also applicable to this architecture. We hence choose Haar wavelet [81] subspaces of $L^2(\mathbb{X})$, the space of square integrable functions and a Hilbert space, and use average pooling as the projection operation P_i [66]. Haar wavelets will in particular be useful to analyse why U-Nets are a good inductive bias in diffusion models (see §4.4). U-Net design and their connection to wavelets has also been studied in [144, 191, 248].

4.3 Generalised U-Net design

In this section, we provide examples of different problems for which our framework can define a natural U-Net architecture. Inspired by Galerkin subspace methods [193], our goal is to use our framework to generalise the design of U-Nets beyond images over a square. Our framework also enables us to encode problem-specific information into the U-Net, such as a natural basis or boundary conditions, which it no longer needs to learn from data, making the U-Net model more efficient.

4.3.1 Multi-ResNets

A closer look at the U-Net encoder. To characterise a U-Net in Definition 4.1, we must in particular choose the encoder subspaces $\mathcal{V}$. This choice depends

on our problem at hand: for instance, if the inputs are images, choosing Haar wavelet subspaces is most likely favourable, because we can represent and compress images in a Haar wavelet basis well, noting that more complex (orthogonal) wavelet bases are possible [153, 212]. What if we choose $\mathcal{V}$ unfavourably? This is where the encoder comes in. While the encoder subspaces $\mathcal{V}$ define an initial basis for our problem, the encoder learns a *change of basis* map to a new, implicit basis $\tilde{v}_i = E_i(v_i)$ which is more favourable. This immediately follows from Eq. (4.1) since U_i acts on V_i through $\tilde{v}_i$. The initial subspaces $\mathcal{V}$ can hence be viewed as a prior for the input compression task which the encoder performs.

Given our initial choice of the encoder subspaces $\mathcal{V}$, the question whether and how much work the encoders $\mathcal{E}$ have to do depends on how far away our choice is from the optimal choice $\tilde{\mathcal{V}}$ for our problem. This explains why the encoders E_i are commonly chosen to be ResNets preconditioned on the identity $E_i^{\text{pre}} = \text{Id}_{V_i}$, allowing the residual encoder E_i^{res} to learn a change of basis. If we had chosen the optimal sequence of encoder subspaces, the residual operator would not have to do any work; leaving the encoder equal to the precondition $E_i = \text{Id}_{V_i}$. It also explains why in practice, encoders are in some cases chosen significantly smaller than the decoder [34], as a ResNet encoder need not do much work given a good initial choice of $\mathcal{V}$. It is precisely this intuition which motivates our second example of a U-Net, the Multi-Res(olution) Res(idual) Network (*Multi-ResNet*).

Definition 4.3. Multi-ResNets. A Multi-ResNet is a residual U-Net with encoder $E_i = \text{Id}_{V_i}$.

Example 2. Haar Wavelet Multi-ResNets. A Haar Wavelet Multi-ResNet is a Haar Wavelet Residual U-Net with encoder $E_i = \text{Id}_{V_i}$.

We illustrate the Multi-ResNet, a novel U-Net architecture, in Figure 4.1 where we choose $E_i = \text{Id}_{V_i}$. Practically speaking, the Multi-ResNet simplifies its encoder to have no learnable parameters, and simply projects to V_i on resolution i. The latter can for the example of Haar wavelets be realised by computing a multi-level Discrete

Wavelet Transform (DWT) (or equivalently average pooling) over the input data [66]. Multi-ResNets allow us to save the parameters in the encoder, and instead direct them to bolster the decoder. In our experiments in §4.5.1, we compare Multi-ResNets to Residual U-Nets and find that for PDE surrogate modelling and image segmentation, Multi-ResNets yield superior performance to Residual U-Nets as Haar wavelets are apparently a good choice for $\mathcal{V}$, while for other problems, choosing Haar wavelets is suboptimal. Future work should hence investigate how to optimally choose $\mathcal{V}$ for a problem at hand. To this end, we will discuss natural bases for $\mathcal{V}$ and $\mathcal{W}$ for specific problems in the remainder of this section.

4.3.2 U-Nets which guarantee boundary conditions

Next, our main goal is to show how to design U-Nets which choose $\mathcal{W}$ in order to encode constraints on the output space directly into the U-Net architecture. This renders the U-Net more efficient as it no longer needs to learn the constraints from data. We consider an example from *PDE surrogate modelling*, approximating solutions to PDE using neural networks, a nascent research direction where U-Nets already play an important role [79], where our constraints are given boundary conditions and the solution space of our PDE. In the *elliptic boundary value problem* on $\mathbb{X} = [0, 1]$ [2], the task is to predict a weak (see Appendix C.1) PDE solution u from its forcing term f given by

$$\Delta u = f, \qquad u(0) = u(1) = 0, \tag{4.2}$$

where u is once weakly differentiable when the equation is viewed in its weak form, $f \in L^2(\mathbb{X})$ and Δu is the Laplacian of u. In contrast to Examples 1 and 2, we choose the decoder spaces as subspaces of $W = \mathcal{H}_0^1$, the space of one weakly differentiable functions with nullified boundary condition, a Hilbert space (see Appendix C.1), and choose $V = L^2(\mathbb{X})$, the space of square integrable functions. This choice ensures that input and output functions of our U-Net are in the correct function class for the prescribed problem. We now want to choose a basis to construct the subspaces

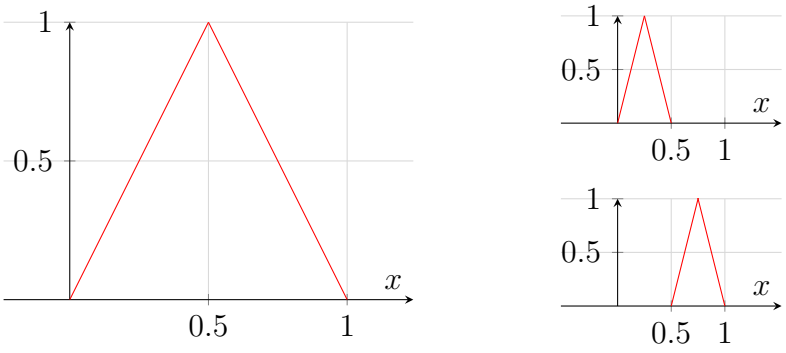


Figure 4.4: Refinement of an orthogonal basis for $\mathcal{H}_0^1 = \text{span}\{\phi_{0,0}, \phi_{1,0}, \phi_{1,1}\}$. We visualise the graphs of basis functions defined in (4.3): [Left] $\phi_{0,0} = \phi$, [Top Right] $\phi_{1,0}$, and [Bottom Right] $\phi_{1,1}$. When increasing resolution, steeper triangular-shaped basis functions are constructed.

$\mathcal{V}$ and $\mathcal{W}$ of V and W. For $\mathcal{V}$, just like in Multi-ResNets in Example 2, we choose V_j to be the Haar wavelet space of resolution j, an orthogonal basis. For $\mathcal{W}$, we also choose a refinable basis, but one which is natural to $\mathcal{H}_0^1$. In particular, we choose $W_i = \text{span}\{\phi_{k,j} : j \leq k, k = 1, \ldots, 2^{i-1}\}$ where

$$\phi_{k,j}(x) = \phi(2^k x + j/2^k), \qquad \phi(x) = 2x \cdot \mathbb{1}_{[0,1/2)}(x) + (2 - 2x) \cdot \mathbb{1}_{[1/2,1]}(x). \tag{4.3}$$

This constructs an orthogonal basis of $\mathcal{H}_0^1$, illustrated in Figure 4.4, which emulates our design choice in Section 4.3.1 where the orthogonal Haar wavelet basis was beneficial as W was L^2-valued. Each $\phi_{k,j}$ obeys the regularity and boundary conditions of our PDE, and consequently, an approximate solution from our U-Net obeys these functional constraints as well. These constraints are encoded into the U-Net architecture and hence need not be learned from data. This generalised U-Net design paves the way to broaden the application of U-Nets, analogous to the choice of bases for Finite Element [26] or Discontinuous Galerkin methods [38].

4.3.3 U-Nets for complicated geometries

Our framework further allows us to design U-Nets which encode the geometry of the input space right into the architecture. This no longer requires to learn the geometric structure from data and enables U-Nets for complicated geometries. In

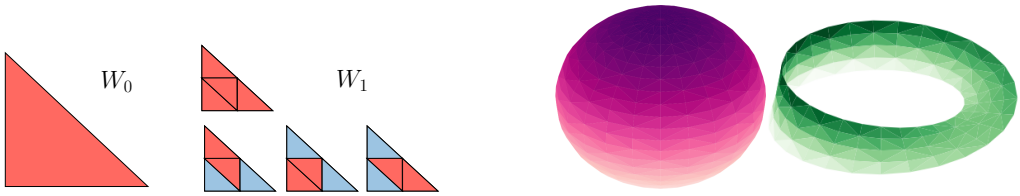


Figure 4.5: U-Nets encoding the topological structure of a problem. [Left] A refinable Haar wavelet basis with basis functions on a right triangle, $\phi_{i,j=0} = \mathbb{1}_{\text{red}} - \mathbb{1}_{\text{blue}}$. [Right] A sphere and a Möbius strip meshed with a Delaunay triangulation [54, 134]. Figures and code as modified from [7].

particular, we are motivated by *tessellations*, the partitioning of a surface into smaller shapes, which play a vital role in modelling complicated geometries across a wide range of engineering disciplines [22]. We here focus on U-Nets on a triangle due to the ubiquitous use of triangulations, for instance in CAD models or simulations, but note that our design principles can be applied to other shapes featuring a self-similarity property. We again are interested in finding a natural basis for this geometry, and characterise key components of our U-Net.

In this example, neither W nor V are selected to be $L^2(\mathbb{X})$ valued on the unit square (or rectangle) $\mathbb{X}$. Instead, in contrast to classical U-Net design in literature, $W = V = L^2(\triangle)$, where $\triangle$ is a right-triangular domain illustrated in Figure 4.5 [Left]. Note that this right triangle has a self-similarity structure in that it can be constructed from four smaller right triangles, continuing recursively. A refinable Haar wavelet basis for this space can be constructed by starting from $\phi_{i,0} = \mathbb{1}_{\text{red}} - \mathbb{1}_{\text{blue}}$ for $j = 0$ as illustrated in Figure 4.5 [Left]. This basis can be refined through its self-similarity structure to define each subspace from these basis functions via $W_i = V_i = \text{Span}\{\phi_{k,j} \,:\, j \leq k,\ k = 1, \ldots, 2^{i-1}\}$ (see Appendix C.1 for details). In §4.5.3, we investigate this U-Net design experimentally. In Appendix C.1 we sketch out how developing our U-Net on triangulated manifolds enables score-based diffusion models on a sphere [51] without any adjustments to the diffusion process itself. This approach can be extended to complicated geometries such as manifolds or CW-complexes as illustrated in Figure 4.5 [Right].

4.4 Why U-Nets are a useful inductive bias in diffusion models

U-Nets are a go-to neural architecture for diffusion models particularly on image data, as demonstrated in an abundance of previous work [52, 56, 92, 120, 172, 196, 199, 208, 209]. However, the reason why U-Nets are particularly effective in the context of diffusion models is understudied. Our U-Net framework enables us to analyse this question. We focus on U-Nets over nested Haar wavelet subspaces $\mathcal{V} = \mathcal{W}$ that increase to $V = W = L^2([0,1])$, with orthogonal projection $Q_j : W \mapsto W_i$ on to W_i corresponding to an average pooling operation Q_i [80] (see Appendix C.3). U-Nets with average pooling are a common choice for diffusion models in practice, for instance when modelling images [56, 92, 120, 208, 209]. We provide theoretical results which identify that high-frequencies in a forward diffusion process are dominated by noise exponentially faster, and how U-Nets with average pooling exploit this in their design.

Let $X \in W$ be an infinite resolution image. For each resolution i define the image $X_i = Q_i X \in W_i$ on 2^i pixels which can be described by $X_i = \sum_k X_i^{(k)} \phi_k$, where $\Phi = \{\phi_k : k = 1, \dots, 2^i\}$ is the standard (or 'pixel') basis. The image X_i is a projection of the infinite resolution image X to the finite resolution i. We consider the family of denoising processes $\{X_i(t)\}_{i=1}^{\infty}$, where for resolution i, the process $X_i(t) = \sum_k X_i^{(k)}(t)\phi_k \in W_i$ is initialised at X_i and evolves according to the denoising diffusion forward process (DDPM, [92]) at each pixel $X_i^{(k)}(t) := \sqrt{1-\alpha_t} X_i^{(k)} + \sqrt{\alpha_t}\varepsilon^{(k)}$ for standard Gaussian noise $\varepsilon^{(k)}$. We now provide our main result (see Appendix C.1 for technical details).

Theorem 4.2. *For time $t \geq 0$ and $j \geq i$, $Q_i X_j(t) \overset{d}{=} X_i(t)$. Furthermore if $X_i(t) = \sum_{j=0}^{i} \widehat{X}^{(j)}(t) \cdot \widehat{\boldsymbol{\phi}}_j$, be the decomposition of $X_i(t)$ in its Haar wavelet frequencies (see Appendix C.3). Each component $\widehat{X}^{(j)}(t)$ of the vector has variance 2^{j-1} relative to the variance of the base Haar wavelet frequency.*

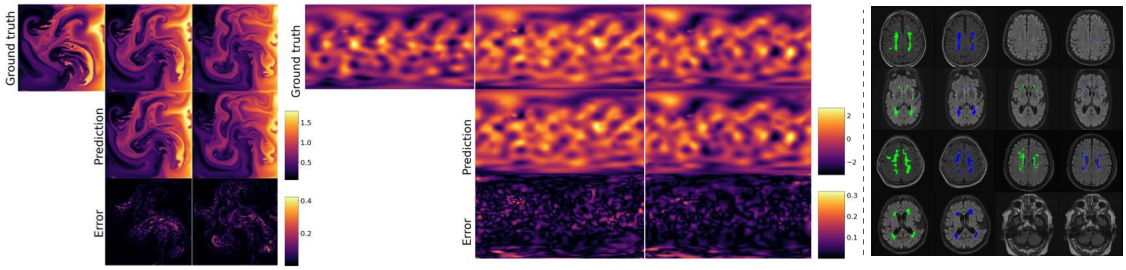


Figure 4.6: PDE modelling and image segmentation with a Multi-ResNet. [Left,Middle] Rolled out PDE trajectories (ground-truth, prediction, L^2-error) from the `Navier-Stokes` [Left], and the `Shallow Water` equation [Middle]. Figure and code as modified from [79, Figure 1]. [Right] MRI images from `WMH` with overlayed ground-truth (green) and prediction (blue) mask.

Theorem 4.2 analyses the noising effect of a forward diffusion process in a Haar wavelet basis. It states that the noise introduced by the forward diffusion process is more prominent in the higher-frequency wavelet coefficients (large j), whereas the lower-frequency coefficients (small j) preserves the signal. Optimal recovery of the signal in such a scenario has been investigated in [60], where soft thresholding of the wavelet coefficients provides a good L^2 estimator of the signal and separates this from the data. In other words, if we add i.i.d. noise to an image, we are noising the higher frequencies faster than the lower frequencies. In particular, high frequencies are dominated by noise *exponentially* faster. It also states that the noising effect of our diffusion on resolution i, compared to the variance of the base frequency $i = 0$, destroys the higher-frequency details as $i \to \infty$ for any positive diffusion time.

We postulate that U-Nets with average pooling exploit precisely this observation. Recall that the primary objective of a U-Net in denoising diffusion models is to separate the signal from the noise which allows reversing the noising process. In the 'predict the noise' or ϵ recovery regime, the network primarily distinguishes the noise from the signal in the input. Yet, our analysis remains relevant, as it fundamentally pertains to the signal-to-noise ratio. Through average pooling, the U-Net discards those higher-frequency subspaces which are dominated by noise, because average pooling is conjugate to projection in a Haar wavelet basis [66, Theorem 2]. This inductive bias enables the encoder and decoder networks to focus on the signal on a

low enough frequency which is not dominated by noise. As the subspaces are coupled via preconditioning, the U-Net can learn the signal which is no longer dominated by noise, added on each new subspace. This renders U-Nets a computationally efficient choice in diffusion models and explains their ubiquitous use in this field.

4.5 Experiments

We conduct three main experimental analyses: (A) Multi-ResNets which feature an encoder with no learnable parameters as an alternative to classical Residual U-Nets, (B) Multi-resolution training and sampling, (C) U-Nets encoding the topological structure of triangular data. We refer to Appendix C.2.4 for our Ablation Studies, where a key result is that U-Nets crucially benefit from the skip connections, hence the encoder is successful and important in compressing information. We also analyse the multi-resolution structure in U-Nets, and investigate different orthogonal wavelet bases. These analyses are supported by experiments on three tasks: (1) Generative modelling of images with diffusion models, (2) PDE Modelling, and (3) Image segmentation. We choose these tasks as U-Nets are a go-to and competitive architecture for them. We report the following performance metrics with mean and standard deviation over three random seeds on the test set: FID score [89] for (1), rollout mean-squared error (r-MSE) [79] for (2), and the Sørensen–Dice coefficient (Dice) [57, 210] for (3). As datasets, we use `MNIST` [133], a custom triangular version of MNIST (`MNIST-Triangular`) and `CIFAR10` [127] for (1), `Navier-stokes` and `Shallow water` equations [79] for (2), and the MICCAI 2017 White Matter Hyperintensity (`WMH`) segmentation challenge dataset [130, 137] for (3). We provide our PyTorch code base at https://github.com/FabianFalck/unet-design. We refer to Appendices C.2, and C.4 for details on experiments, further experimental results, the datasets, and computational resources used.

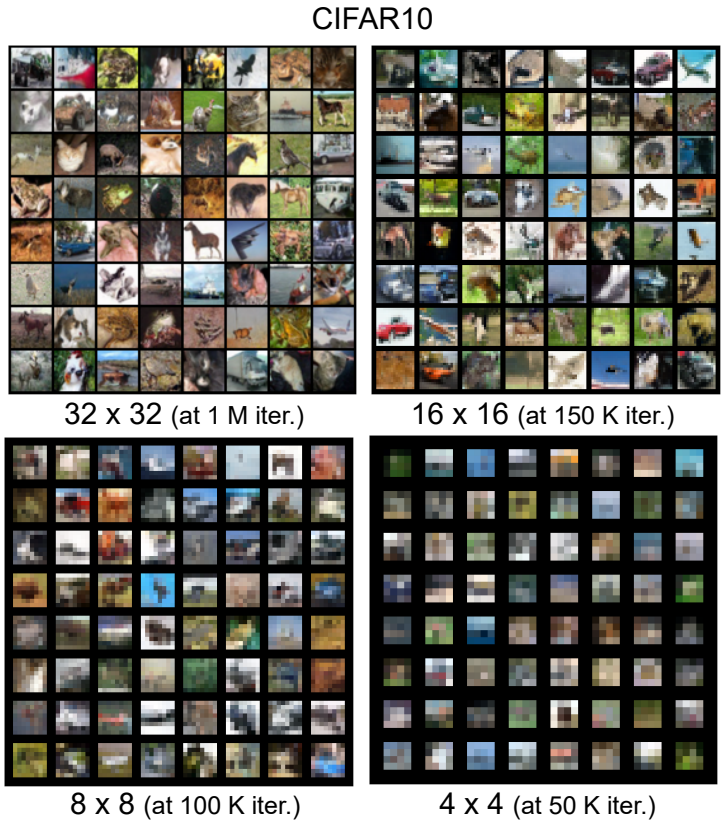


Figure 4.7: Preconditioning enables multi-resolution training and sampling of diffusion models.

4.5.1 The role of the encoder in a U-Net

In §4.3.1 we motivated Multi-ResNets, Residual U-Nets with identity operators as encoders over Haar wavelet subspaces $\mathcal{V} = \mathcal{W}$ of $V = W = L^2(\mathbb{X})$. We analysed the role of the encoder as learning a change of basis map and found that it does not need to do any work, if $\mathcal{V}$, the initial basis, is chosen optimally for the problem at hand. Here, we put exactly this hypothesis derived from our theory to a test. We compare classical (Haar wavelet) Residual U-Nets with a (Haar wavelet) Multi-ResNet. In Table 4.1, we present our results quantified on trajectories from the `Navier-stokes` and `Shallow water` PDE equations unrolled over several time steps and image segmentation as illustrated in Figure 4.6. Our results show that Multi-ResNets have competitive and sometimes superior performance when compared to a classical U-Net with roughly the same number of parameters. Multi-ResNets outperform classical U-Nets by 29.8%, 12.8% and 3.4% on average over three random seeds,

Table 4.1: Quantitative performance of the (Haar wavelet) Multi-ResNet compared to a classical (Haar wavelet) Residual U-Net on two PDE modelling and an image segmentation task.

Dataset	Neural architecture	# Par.	r-MSE ↓ / Dice ↑
Navier-stokes 128×128	Residual U-Net	34.5 M	$0.0057 \pm 2 \cdot 10^{-5}$
	Multi-ResNet, no par. added in dec. (*ours*)	15.7 M	$0.0107 \pm 9 \cdot 10^{-5}$
	Multi-ResNet, saved par. added in dec. (*ours*)	34.5 M	$\mathbf{0.0040 \pm 2 \cdot 10^{-5}}$
Shallow water 96×192	Residual U-Net	34.5 M	0.1712 ± 0.0005
	Multi-ResNet, no par. added in dec. (*ours*)	15.7 M	0.4899 ± 0.0156
	Multi-ResNet, saved par. added in dec. (*ours*)	34.5 M	$\mathbf{0.1493 \pm 0.0070}$
WMH 200×200	Residual U-Net	2.2 M	0.8069 ± 0.0234
	Multi-ResNet, no par. added in dec. (*ours*)	1.0 M	0.8190 ± 0.0047
	Multi-ResNet, saved par. added in dec. (*ours*)	2.2 M	$\mathbf{0.8346 \pm 0.0388}$

respectively. In Appendix C.2.1, we also show that U-Nets outperform FNO [141], another competitive architecture for PDE modelling, in this experimental setting.

For the practitioner, this is a rather surprising result. We can simplify classical U-Nets by replacing their parameterised encoder with a fixed, carefully chosen hierarchy of linear transformation as projection operators P_i, here a multi-level Discrete Wavelet Transform (DWT) using Haar wavelets, and identity operators for E_i. This 'DWT encoder' has no learnable parameters and comes at almost no computational cost. We then add the parameters we save into the decoder and achieve competitive and—on certain problems—strictly better performance when compared with a classical U-Net. However, as we show for generative modelling with diffusion models in Appendix C.2.1, Multi-ResNets are competitive with, yet inferior to Residual U-Nets, because the initial basis which $\mathcal{V}$ imposes is suboptimal and the encoder would benefit from learning a better basis. This demonstrates the strength of our framework in understanding the role of the encoder and when it is useful to parameterise it, which depends on our problem at hand. It is now obvious that future empirical work should explore how to choose $\mathcal{V}$ (and $\mathcal{W}$) optimally, possibly eliminating the need of a parameterised encoder, and also carefully explore how to optimally allocate and make use of the (saved) parameters in Multi-ResNets [104, 183].

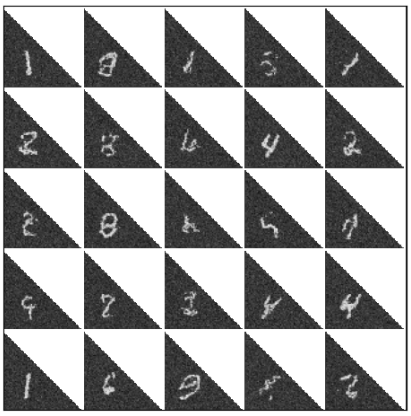


Figure 4.8: U-Nets encode the geometric structure of data.

4.5.2 Staged training enables multi-resolution training and inference

Having characterised the self-similarity structure of U-Nets in §4.2, a natural idea is to explicitly train the U-Net $\mathcal{U}$ on resolution $i-1$ first, then train the U-Net on resolution i while preconditioning on U_{i-1}, continuing recursively. Optionally, we can freeze the weights of $\mathcal{U}_{i-1}$ upon training on resolution $i-1$. We formalise this idea in Algorithm 1. Algorithm 1 enables training and inference of U-Nets on multiple resolutions, for instance when several datasets are available. It has two additional advantages. First, it makes the U-Net modular with respect to data. When data on a higher-resolution is available, we can reuse our U-Net pretrained on a lower-resolution in a principled way. Second, it enables to checkpoint the model during prototyping and experimenting with the model as we can see low-resolution outputs of our U-Net early.

In Figure 4.7 we illustrate samples from a DDPM diffusion model [92] with a Residual U-Net trained with Algorithm 1. We use images and noise targets on multiple resolutions (`CIFAR10`/`MNIST`: $\{4 \times 4, 8 \times 8, 16 \times 16, 32 \times 32\}$) as inputs during each training stage. We observe high-fidelity samples on multiple resolutions at the end of each training stage, demonstrating how a U-Net trained via Algorithm 1 can utilise the data available on four different resolutions. It is also worth noting

Algorithm 1 Multi-resolution training and sampling via preconditioning.

Require: Boolean `FREEZE`.
1: **for** $i \leftarrow \{1, \dots, J\}$ **do**
2: **if** $i > 1$ **then**
3: Precondition on U_{i-1}.
4: **end if**
5: Train U_i
6: **if** `FREEZE` is True **then**
7: Freeze E_i^θ and D_i^θ (fix parameters).
8: **end if**
9: **end for**

that training with Algorithm 1 as opposed to single-stage training (as is standard practice) does not substantially harm performance of the highest-resolution samples: (FID on `CIFAR10`: staged training: 8.33 ± 0.010; non-staged training: 7.858 ± 0.250). We present results on `MNIST`, `Navier-Stokes` and `Shallow water`, with Multi-ResNets, and with a strict version of Algorithm 1 where we freeze E_i^θ and D_i^θ after training on resolution i in Appendix C.2.2.

4.5.3 U-Nets encoding topological structure

In §4.3.3, we showed how to design U-Nets with a natural basis on a triangular domain, which encodes the topological structure of a problem into its architecture. Here, we provide proof-of-concept results for this U-Net design. In Figure 4.8, we illustrate samples from a DDPM diffusion model [92] with a U-Net where we choose $\mathcal{V}$ and $\mathcal{W}$ as Haar wavelet subspaces of $W = V = L^2(\triangle)$ (see §4.3.3), ResNet encoders and decoders and average pooling. The model was trained on `MNIST-Triangular`, a custom version of `MNIST` with the digit and support over a right-angled triangle. While we observe qualitatively correct samples from this dataset, we note that these are obtained with no hyperparameter tuning to improve their fidelity. This experiment has a potentially large scope as it paves the way to designing natural U-Net architectures on tessellations of complicated geometries such as spheres, manifolds, fractals, or CW-complexes.

4.6 Conclusion

We provided a framework for designing and analysing U-Nets. Our work has several limitations: We put particular emphasis on Hilbert spaces as the decoder spaces. We focus on orthogonal wavelet bases, in particular of $L^2(\mathbb{X})$ or $L^2(\triangle)$, while other bases could be explored (e.g. Fourier frequencies, radial basis functions). Our framework is motivated by subspace preconditioning, with requires the user to actively design and choose which subspaces they wish to precondition on. Our analysis of signal and noise concentration in Theorem 4.2 has been conducted for a particular, yet common choice of denoising diffusion model, with one channel, and with functions supported on one spatial dimension only, but can be straightforwardly extended with the use of a Kronecker product. Little to no tuning is performed how to allocate the saved parameters in Multi-ResNet in §4.5.1. We design and empirically demonstrate U-Nets on triangles, while one could choose a multitude of other topological structures. Lastly, future work should investigate optimal choices of $\mathcal{U}$ for domain-specific problems.

Acknowledgments and Disclosure of Funding

Christopher Williams acknowledges support from the Defence Science and Technology (DST) Group and from a ESPRC DTP Studentship. Fabian Falck acknowledges the receipt of studentship awards from the Health Data Research UK-The Alan Turing Institute Wellcome PhD Programme (Grant Ref: 218529/Z/19/Z), and the Enrichment Scheme of The Alan Turing Institute under the EPSRC Grant EP/N510129/1. Chris Holmes acknowledges support from the Medical Research Council Programme Leaders award MC_UP_A390_1107, The Alan Turing Institute, Health Data Research, U.K., and the U.K. Engineering and Physical Sciences Research Council through the Bayes4Health programme grant. Arnaud Doucet acknowledges support of the UK Defence Science and Technology Laboratory (Dstl)

and EPSRC grant EP/R013616/1. This is part of the collaboration between US DOD, UK MOD and UK EPSRC under the Multidisciplinary University Research Initiative. Saifuddin Syed and Arnaud Doucet also acknowledge support from the EPSRC grant EP/R034710/1.

The authors report no competing interests.

This research is supported by research compute from the Baskerville Tier 2 HPC service. Baskerville is funded by the EPSRC and UKRI through the World Class Labs scheme (EP/T022221/1) and the Digital Research Infrastructure programme (EP/W032244/1) and is operated by Advanced Research Computing at the University of Birmingham.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	A Unified Framework for U-Net Design and Analysis
Publication Status	x Published □ Accepted for Publication □ Submitted for Publication □ Unpublished and unsubmitted work written in a manuscript style
Publication Details	A Unified Framework for U-Net Design and Analysis. Christopher Williams*, Fabian Falck*, George Deligiannidis, Chris Holmes, Arnaud Doucet, Saifuddin Syed. Advances in Neural Information Processing Systems (NeurIPS) 2023. * indicates equal contribution.

Student Confirmation

Student Name:	Fabian Falck		
Contribution to the Paper	I follow the CRediT contributor role taxonomy (Brand et al., 2015). Christopher Williams: Conceptualization, Methodology, Software, Formal analysis, Investigation, Writing - Original Draft, Writing - Review & Editing, Visualization Fabian Falck: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing - Original Draft, Writing - Review & Editing, Visualization George Deligiannidis: Writing - Review & Editing, Supervision, Funding acquisition Chris Holmes: Methodology, Resources, Writing - Review & Editing, Supervision, Project administration, Funding acquisition Arnaud Doucet: Methodology, Writing - Review & Editing, Supervision, Project administration, Funding acquisition Saifuddin Syed: Conceptualization, Methodology, Investigation, Writing - Original Draft, Writing - Review & Editing, Supervision, Project administration		
Signature		Date	September 29, 2024

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof. Arnaud Doucet		
Supervisor comments I agree with the description of the contributions.		
 Signature	Date	25/11/2024

This completed form should be included in the thesis, at the end of the relevant chapter.

5

Is In-Context Learning in Large Language Models Bayesian? A Martingale Perspective

Abstract

In-context learning (ICL) has emerged as a particularly remarkable characteristic of Large Language Models (LLM): given a pretrained LLM and an observed dataset, LLMs can make predictions for new data points from the same distribution without fine-tuning. Numerous works have postulated ICL as approximately Bayesian inference, rendering this a natural hypothesis. In this work, we analyse this hypothesis from a new angle through the *martingale property*, a fundamental requirement of a Bayesian learning system for exchangeable data. We show that the martingale property is a necessary condition for unambiguous predictions in such scenarios, and enables a principled, decomposed notion of uncertainty vital in trustworthy, safety-critical systems. We derive actionable checks with corresponding theory and test statistics which must hold if the martingale property is satisfied. We also examine if uncertainty in LLMs decreases as expected in Bayesian learning when more data is observed. In three experiments, we provide evidence for violations of the martingale property, and deviations from a Bayesian scaling behaviour of uncertainty, falsifying the hypothesis that ICL is Bayesian.

Contents

5.1 Introduction

Large Language Models (LLMs) are autoregressive generative models trained on vast amounts of data, exhibiting extraordinary performance across a wide array of tasks [261]. A particularly remarkable characteristic of LLMs is so-called *in-context learning* (ICL) [27, 59]: Given a pretrained language model p_M and an observed dataset $D := \{(x_1, y_1), \ldots, (x_n, y_n)\} = z_{1:n}$ of samples, LLMs capture the distribution of the underlying random variables X and Y in this in-context dataset. This allows them produce a new sample (x_{n+1}, y_{n+1}) using the *predictive distribution* $p_M(X_{n+1}, Y_{n+1}|Z_{1:n} = z_{1:n})$, or if x_{n+1} is observed infer the predictive distribution $p_M(Y_{n+1}|X_{n+1} = x_{n+1}, Z_{1:n} = z_{1:n})$, without retraining or fine-tuning p_M.

Few-shot learning via ICL [27] has produced numerous breakthroughs in LLM research [59], such as in supervised learning [164] or chain-of-thought prompting [235]. In spite of the remarkable empirical success of ICL, we lack a unified understanding of the algorithm and the properties of conditioning LLMs on in-context data. In this work, we are interested in characterising the type of learning that occurs in ICL. Specifically, we aim to answer the question: **is in-context learning for LLMs on exchangeable data (approximately) Bayesian?**

In contrast to prior work, our analysis focuses on one fundamental property of Bayesian learning systems for exchangeable data: the *martingale property*. In a nutshell, the martingale property describes the invariance of a model's predictive distribution with respect to missing data from a population. We will formally define and extensively explain the martingale property in §5.2, but begin by

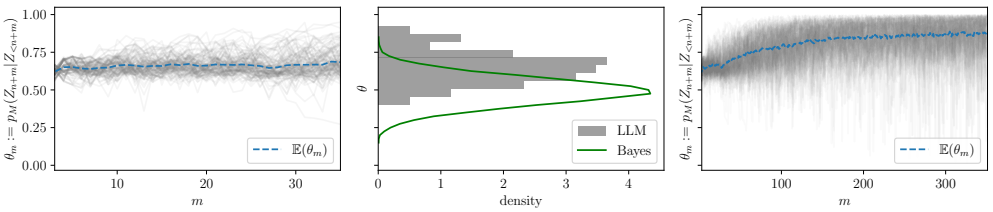


Figure 5.1: *In-context learning in Large Language Models is not Bayesian.* [Left] The *martingale property*, a necessary condition of Bayesian learning systems, is satisfied for short sample paths. [Centre] This allows us to approximate the *martingale posterior* (see §5.2.3) which, however, indicates deviation from a reference Bayesian model. [Right] For longer sample paths, we observe a drift which violates the martingale property, together rendering the ICL system non-Bayesian.

intuitively describing two important and desirable *consequences* of it with an example, highlighting its relevance. These consequences are: (i) the martingale property is a necessary condition for rendering *predictions unambiguous* in an *exchangeable* data setting, and (ii) it establishes a *principled notion* of the model's *uncertainty*.

Consider a drug company exploring the efficacy of a new medication for headaches. The company runs a two-arm Randomised Control Trial (RCT) with 100 patients, 50 in each arm, comparing the new treatment with the current standard of care (in this case ibuprofen), and records the outcome $Y \in \{0, 1\}$ whether patients are symptom-free four hours after treatment. It is important to note that in this setting, the distribution of outcomes is independent of the order in which the patients are observed, a property known as *exchangeability* (see §5.2 for a formal definition). Half-way into the trial, the company conducts an interim analysis. Define the interim observations $D = \{(x_1, y_1), \ldots, (x_{50}, y_{50})\}$ where y_k indicates outcome, and x_k the treatment arm and other patient covariates. Given these observations, the company wants to decide whether to stop the trial early. The company uses an LLM, which was trained on potentially useful background information from the internet (e.g. on clinical trials, or the efficacy of ibuprofen), to generate or impute the missing outcomes Y for the missing patients via ICL, and determines if the RCT is successful combining the observed and synthetic data. It repeats this imputation procedure J times, and decides to keep going with the trial if the fraction of symptom-free patients in the treatment over the control arm is above

a certain threshold on average over these J hypothetical trials. *Should we trust the LLM's prediction using ICL under this procedure?*

In preview of our experimental results in §5.4, the answer is *'No'*. Our experiments present evidence that state-of-the-art LLMs violate the martingale property in certain settings (see Fig. 5.1). The martingale property is a necessary condition for exchangeability, and in turn a fundamental property of Bayesian learning. If the martingale property is violated by an LLM performing ICL it implies that the model's predictions are not exchangeable, and hence that ICL with this LLM is not following any reasonable notion of probabilistic conditioning. This renders the LLM's predictive distribution incoherent: the model can make different predictions depending on the order in which the patients are imputed. This is problematic because by the design of an RCT, we know that there is no outcome dependence on the order of observations. It is incoherent and ambiguous to receive a different marginal predictions if we for example impute patient # 51 or patient # 100 first. Note that independent and identically distributed (i.i.d.) is a stricter condition implying exchangeability, and hence our work also applies to any i.i.d. data setting. This should caution the practitioner of the use of LLMs in exchangeable applications and data settings.

But there is a second reason why the martingale property is crucial: it enables a principled interpretation of the *uncertainty* of LLMs, allowing us to decompose inference into epistemic and aleatoric uncertainty (see §5.2 for a detailed introduction). Revisiting the RCT example above, if we acquire data from the 50 remaining patients, a costly decision, can this substantially decrease (epistemic) uncertainty? What is the effect of acquiring additional features for each patient, e.g. a genetic predisposition, on the (aleatoric) uncertainty? – Without satisfying the martingale property, we have no understanding of the effect on reducing uncertainty in applications where additional data acquisition is feasible, for instance active learning or reinforcement learning. We cannot study the question 'why is the point prediction of my LLM imprecise' in a principled way, and the uncertainty

of an LLM's predictive distribution remains opaque. This finding has important implications for safety-critical, high-stakes applications of LLMs where trustworthy systems with a principled uncertainty estimate are vital.

This work states the hypothesis that ICL in LLMs given exchangeable data is Bayesian. Numerous works have argued that ICL approximates some form of Bayesian inference [3, 82, 107, 243, 256] which we will review in App. D.4, rendering this hypothesis natural. Our work introduces a novel perspective which contradicts their conclusion: we show that the martingale property, a fundamental property of Bayesian learning systems, is violated for state-of-the-art LLMs such as Llama2, Mistral, GPT-3.5 and GPT-4. We on purpose focus our analysis on three synthetic experiments where the ground-truth data generating process is simple and known, and which provide a useful test bed without the convolution of unknown latent effects as is typical in natural language. Our goal is to provide a scientific and precise framework which measures and quantifies the degree to which ICL of an LLM is Bayesian.

More specifically, our *contributions* are: (a) We motivate the martingale property as a fundamental property of Bayesian learning, crucial for unambiguous predictions of an LLM in exchangeable settings, and a principled interpretation of uncertainty in LLMs (§5.2). (b) We derive actionable diagnostics with corresponding theory and test statistics of the martingale property for ICL. We also characterise the efficiency of ICL compared to standard Bayesian inference (§5.3). (c) We provide novel evidence for violations of the martingale property through LLMs in certain settings, and a deviation of the sample efficiency of ICL relative to Bayesian systems, falsifying our hypothesis that ICL in LLMs is Bayesian and cautioning against the use of LLMs in exchangeable and safety-critical applications (§5.4).

5.2 What Characterises a Bayesian Learning System? A Martingale Perspective

In this section we rigorously formalise properties of an ICL system that follows Bayesian principles. Theoretical details and technical proofs are presented in App. D.1.

5.2.1 The Martingale Property

We begin by defining the *martingale property*.

Definition 5.1. The predictive distributions for $\{Z_i\}$ satisfy the *martingale property* if for all integers $n, k > 0$ and realisations $\{z, z_{1:n}\}$ we have

$$p_M(Z_{n+1}{=}z|Z_{1:n}{=}z_{1:n}){=}p_M(Z_{n+k}{=}z|Z_{1:n}{=}z_{1:n}). \tag{5.1}$$

Eq. (5.1) states that $\{Z_i\} \sim p_M$ are *conditionally identically distributed* ([17]). As we will explain in §5.2.3, this renders distributions $\{p_M(Z_{n+1} = \cdot|Z_{1:n})\}$ to form a martingale, hence the name 'martingale property'.

It follows from Eq. (5.1) that predictive distributions of the form $p_M(Y_{n+k}|X_{n+k}, Z_{1:n})$ satisfy a similar identity:

$$\begin{aligned} &p_M(Y_{n+1} = y|X_{n+1} = x, Z_{1:n} = z_{1:n}) \\ &= p_M(Y_{n+k} = y|X_{n+k} = x, Z_{1:n} = z_{1:n}) \\ &= \mathbb{E}_{Z_{n+1:n+k-1}\sim p_M(\cdot|Z_{1:n}=z_{1:n})} p_M(Y_{n+k} = y|X_{n+k} = x, Z_{1:n+k-1}), \end{aligned} \tag{5.2}$$

for all integers $n, k > 0$, realisations $\{z_{1:n}, y\}$, and (almost every) realisation x measured by $p_M(X_{n+1}|Z_{1:n} = z_{1:n})$. In Eq. (5.2) the martingale property renders a model's predictions invariant to imputations of missing samples from the population (on average). Note that Eqs. (5.1) and (5.2) are equivalent in the unconditional case ($x_i = \varnothing$), which we consider in the majority of our experiments in §5.4.

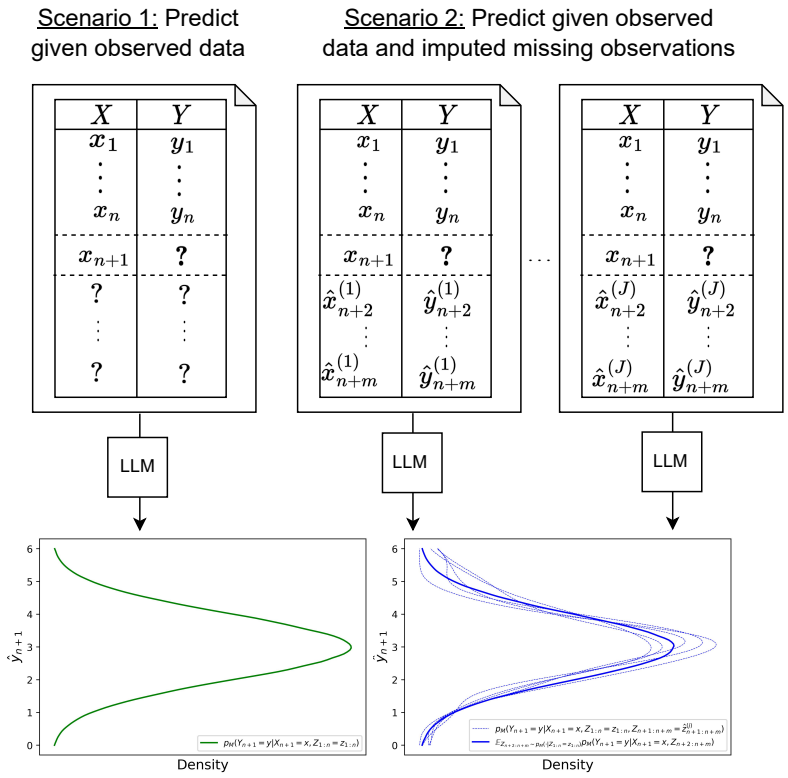


Figure 5.2: The *martingale property*, a fundamental requirement of a Bayesian learning system, requires *invariance with respect to missing samples from a population.*

5.2.2 The Martingale Property is Necessary for Unambiguous Predictions under Exchangeable Data

To understand the intuition behind the seemingly technical notion of the martingale property, consider two scenarios for ICL, illustrated in Fig. 5.2. In both scenarios, the LLM is given the observed data (D, x_{n+1}). In scenario 1, the LLM directly infers the *predictive distribution* $p_M(Y_{n+1}|Z_{1:n} = z_{1:n}, X_{n+1} = x_{n+1})$. In scenario 2, before making a prediction, the LLM generates (imputes) $m-1$ missing samples $\hat{z}_{n+2:n+m}$ from the population autoregressively. Given the observed data and the imputed samples as a prompt, we then sample from the LLM's predictive distribution $p_M(Y_{n+1}|Z_{1:n} = z_{1:n}, X_{n+1} = x_{n+1}, Z_{n+2:n+m} = \hat{z}_{n+2:n+m})$. We repeat this imputation procedure J times and average the obtained predictive distributions to receive a Monte Carlo estimate of the right-hand side of Eq. (5.2). Scenario 2 is of practical interest when estimating aggregated statistics of a population as illustrated in our RCT example in §5.1. – The martingale property then states that the predictive

distribution from scenario 1, $p_M(Y_{n+1}|Z_n = z_n, X_{n+1} = x_{n+1})$, and the predictive distribution from scenario 2, $p_M(Y_{n+1}|Z_n = z_n, x_{n+1}, Z_{n+2:n+m} = \hat{z}_{n+2:n+m})$, when averaged over all possible imputations of $\hat{z}_{n+2:n+m}$, are equivalent.

Why is the martingale property natural for any probabilistic system, and LLMs in particular? It is important to observe that all information about the distribution of X and Y presented to the model (in addition to its prior belief [251]) lies in the observed data (D, x_{n+1}). Imputing the samples $\hat{z}_{n+2:n+m}$ should hence not change the predictive distribution for y_{n+1} when averaged over all possible imputations. This is precisely the core idea of the martingale property. If the predictive distribution for y_{n+1} changes on average, the model is 'creating new knowledge' when there is none: it is 'hallucinating'. In preview of our experimental results in §5.4, we observe this violation of the martingale property in state-of-the-art LLM families. We call this phenomenon *introspective hallucinations*: by querying itself, the model changes its predictions (on average), which as we shall see in §5.2.4 violates how Bayesian systems learn.

There is another way in which predictions are rendered unambiguous: under *exchangeability* for which the martingale property is a necessary condition (see App. D.1) the model is invariant to the order of the observed and missing data. This requirement is vital if we know that the order of the underlying distributions is irrelevant, for instance because—as in the RCT example in §5.1—we have designed the experiment such that we can exclude a dependency on the order. Formally, this concept is known as exchangeability. A sequence of random variables $\{Z_i\} \sim p_M$ is *exchangeable* if for all $\ell \in \mathbb{N}$ and ℓ-permutations σ,

$$p_M(Z_1, \ldots, Z_\ell) = p_M(Z_{\sigma(1)}, \ldots, Z_{\sigma(\ell)}). \tag{5.3}$$

Exchangeability guarantees the invariance of predictions to the ordering of the observations $Z_{1:n}$, but also with respect to the order of future imputations $Z_{n+1,\ldots}|Z_{1:n}$. In the standard ICL setup, it is natural to assume that the sequence of example tuples

in the ICL dataset, which is part of the prompt, is i.i.d. and thus exchangeable, and many influential works make this assumption (often without stating it explicitly) [107, 231, 243]. To understand the importance of this assumption further, consider the RCT example in §5.1, where $\{Z_{1:100}\}$ are (by experimental design) exchangeable. A model p_M should hence satisfy

$$\begin{aligned} &p_M(Y_{n+k}|X_{n+k}=x, Z_{1:n}, X_{n+1:n+k-1}=\hat{x}_{n+1:n+k-1}) \\ =\, &p_M(Y_{n+k}|X_{n+k}=x, Z_{1:n}, X_{n+1:n+k-1}=\hat{x}_{\sigma(n+1:n+k-1)}), \end{aligned}$$

meaning that the prediction for $Y_{n+k}|D, X_{n+k}$ is independent of the order of the imputed inputs $\hat{x}_{n+1:n+k-1}$. If a model p_M violates the above equality, there may be ambiguities in the prediction of the next sample (Y_{n+k}, X_{n+k}) as it may depend on and vary with the ordering. Such ambiguities would substantially undermine the credibility of predictions, as well as the downstream decision-making based on such procedures. The martingale property is connected to the above notions of invariance as a necessary condition for exchangeability. Furthermore, it can even ensure exchangeability of imputed samples as the observed sample size n becomes large, because Eq. (5.1) implies asymptotic exchangeability of $Z_{n+1,\dots}|Z_{1:n}$ [17, Thm. 2.5].

5.2.3 The Martingale Property Enables a Principled Notion of Uncertainty

The second desirable and important consequence of the martingale property is that it establishes a principled notion of uncertainty in the model's predicitive distribution. More specifically, it allows us to decompose this uncertainty, enabling us to study and interpret the uncertainty of a model.

To simplify the exposition, suppose the variables Z_i are discrete and have $A < \infty$ realisations (both standard in LLMs) [1], so that any distribution $p_\theta(Z = \cdot)$ can be identified by a vector $\theta \in \mathbb{R}^A$. Let θ_n denote the random vector that indexes

[1] We refer to App. D.2.1 for a review of the more general case.

$p_M(Z_{n+1} \mid Z_{1:n})$. Then, the martingale property is equivalent to stating that $\{\theta_n\}$ form a martingale w.r.t. the filtration defined by $\{Z_n\}$. Under boundedness conditions always satisfied in the above case, Doob's theorem [61] states that θ_n converges almost surely to a random vector θ_∞, and we have $\theta_n = \mathbb{E}_{\theta_\infty|Z_{1:n}}\theta_\infty$, or equivalently,

$$p_M(Z_{n+1}=\cdot \mid Z_{1:n}) = \int p(\theta_\infty|Z_{1:n}) p_{\theta_\infty}(Z=\cdot) d\theta_\infty. \tag{5.4}$$

Note the similarity of Eq. (5.4) with Bayesian inference: the Bayesian posterior predictive distribution has the form

$$p_M(Z_{n+1} = \cdot \mid Z_{1:n}) = \int p(\theta|Z_{1:n}) p(Z = \cdot|\theta) d\theta. \tag{5.5}$$

The random vector θ_∞ plays the same role as the parameter θ in a Bayesian model, as both determine a predictive distribution ($p_{\theta_\infty}(Z)$ or $p(Z|\theta)$). They are thus interchangeable for prediction purposes. Moreover, if p_M is defined through Bayesian inference over θ, p_{θ_∞} will define the same distribution over Z as $p(\cdot|\theta)$ (see App. D.2.1). Therefore we refer to the distribution $\theta_\infty|Z_{1:n}$ as the *martingale posterior*.

Eq. (5.4) shows that the variation or *uncertainty* in the predictive distribution $p_M(Z_{n+1} = \cdot \mid Z_{1:n})$ has two sources:

1. **epistemic uncertainty**, which is about the latent θ_∞ and can be reduced if more data is available; and
2. **aleatoric uncertainty**, which is irreducible given a fixed set of features even if infinite samples are observed and all aspects of the data generating process, namely the latent θ_∞, are known.

The close connection between Eqs. (5.4) and (5.5) shows that this decomposition of uncertainty is established by the same foundations as in Bayesian inference. This is particularly relevant for LLMs which lack clearly stated, interpretable and verifiable assumptions (such as a prespecified statistical model), rendering their predictive distribution a 'black-box'.

Importantly, we can construct the martingale posterior solely using path samples from p_M: we can sample from $p(\theta_{n+k}|Z_{1:n})$ simply by sampling $Z_{n+1:n+k-1}|Z_{1:n}$ as $\lim_{k\to\infty}\theta_{n+k} = \theta_\infty$. Alternatively, we can also estimate parametric models on the path samples as proposed in [70] (see App. D.2.1 for further details). This construction is an appealing tool for interpreting black-box models such as LLMs.

The interpretable decomposition of uncertainty further provides actionable guidance on how the combined uncertainty can be reduced: we can collect more samples to reduce epistemic uncertainty in scenarios where this is possible such as active learning, reinforcement learning or healthcare; particularly in regions of the input space where the uncertainty is high. In §5.3.3 we propose diagnostics to check if epistemic uncertainty decreases w.r.t. training sample size. On the contrary, if the aleatoric uncertainty is high and ought to be reduced, we cannot do so without 'changing the problem', for instance by collecting more features for each data point. This principled notion of uncertainty in a model is crucial in safety-critical, high-stakes scenarios for building trustworthy systems.

We present the following example for further intuition:

Example 5.1. Suppose $Z_i \in \{0, 1\}$. Then $\theta_\infty = (\theta_{\infty,0}, \theta_{\infty,1}) \in \mathbb{R}^2$, and $p_{\theta_\infty} = \text{Bern}(\theta_{\infty,1})$. Thus, in both Eq. (5.4) and Eq. (5.5) the epistemic uncertainty is represented by a distribution over the Bernoulli parameter, revealing their inherent connection. The epistemic uncertainty is especially important in scenarios where we use a black-box model p_M to impute the missing samples $\{Z_{n+i}\}$ from a population —as in the RCT example in §5.1— and want to quantify a model's lack of knowledge about the population. Note this distribution is not identifiable if we only have samples from a single-step predictive distribution $p_M(Z_{n+1}|Z_{1:n})$, but becomes identifiable given *sample paths*.

5.2.4 On the Link between the Martingale Property and Bayesian Learning Systems

So far, we asserted that the martingale property is fundamental to a Bayesian ICL system. In this subsection, we want to further formalise this. We have already discussed the close connection between the martingale property, exchangeability (§5.2.2), and uncertainty (§5.2.3). We will now show that for ICL on i.i.d. data, exchangeability, for which the martingale property is a necessary condition, and Bayesian inference are closely connected, equivalent conditions.

ICL typically assumes i.i.d. observations $Z_{1:n}$, which is our primary focus in this work (see §5.2.2). Therefore, a correctly specified Bayesian model should produce marginal predictive distributions of the form

$$p_M(Z_{1:n} = z_{1:n}) = \int p_M(Z_1 = z_1, \ldots, Z_n = z_n|\theta)\pi(\theta)d\theta \tag{5.6}$$

$$= \int \left(\prod_{i=1}^{n} p_M(Z = z_i|\theta)\right)\pi(\theta)d\theta, \quad \forall n \in \mathbb{N}. \tag{5.7}$$

Here, θ denotes the parameter of a Bayesian model, π denotes the prior measure and $p_M(Z = \cdot \mid \theta)$ denotes the likelihood. From the factorisation over the data dimension n in (5.7), we can see that it is invariant with respect to permutations of $z_{1:n}$, and thus the left-hand side of the equation in (5.6) is invariant, too. It then follows that $\{Z_i\} \sim p_M$ satisfies Eq. (5.3), and thus $\{Z_i\}$ are exchangeable. The converse is also true by de Finetti's representation theorem [53]: Under mild regularity conditions any p_M that defines exchangeable $\{Z_i\}$ must have a representation in the form of Eq. (5.7). It then follows that the predictive distribution $p_M(Z_{n+1}|Z_{1:n})$ has the form of a Bayesian posterior predictive distribution,

$$p_M(Z_{n+1}|Z_{1:n}) = \int p_M(Z_{n+1}|\theta)\pi(\theta|Z_{1:n})d\theta,$$

and can thus be viewed as implicit Bayesian inference for the latent variable θ [100]. In conclusion, ICL on i.i.d. data corresponds to a Bayesian model that assumes (conditionally) i.i.d. observations *if and only if* it defines an exchangeable sample

sequence. Since the martingale property is a necessary condition for exchangeability, an ICL system not satisfying the martingale property cannot be Bayesian.

5.3 Probing Bayesian Learning Systems through Martingales

In this section we introduce practical diagnostics to probe if LLMs match the behaviour of Bayesian learning systems.

5.3.1 Are All Deviations from Bayes Bad? – Expected and Acceptable Deviations from Bayesian Reasoning

Numerous properties are implied if a learning system satisfies the martingale property, a distributional characteristic, and it is both infeasible and unnecessary as often practically irrelevant to check all of them in order to provide evidence for or against our hypothesis. For example, the martingale property implies that all conditional moments should be equivalent, i.e. $\mathbb{E}(Z^l_{n'+1}|Z_{1:n}) = \mathbb{E}(Z^l_{n'+k}|Z_{1:n})$ for all integers $n, n', k, l > 0$ and $n' > n$, yet higher-order moments are not vital in most applications and hence are acceptable deviations, if existent. Therefore, we will restrict our attention to two key implications of the martingale property which—if present—have important practical consequences.

Pretrained LLMs are general-purpose models and can at best approximate Bayesian learning via ICL. The martingale property is an invariance that is not hard-coded in their transformer-based architecture, and can only be approximately (rather than exactly) satisfied. Let us assume that an LLM internally maintains a 'hierarchy of states' [231], say a hierarchical Bayesian model, capturing different tasks (e.g. Bayesian ICL from i.i.d. data, or acting in a dialogue system), and at each sampling step first updates its belief about this state. Say there is a probability p that the LLM deviates from Bayesian ICL or simply fails to approximate. Even if p is small,

the probability of a deviation $1-(1-p)^m$ becomes substantial when accumulated over a long sampling path of length m. In early experiments, we observed frequent poor approximations for long sampling paths (see Fig. D.5 in the Appendix). This would trivially falsify the martingale property and our hypothesis.

In our experiments in §5.4, we hence restrict the sampling paths to a short, finite length where we check the martingale property. We also design our checks to be robust against such behaviour, for example by removing outliers before computing a test statistic. Furthermore, we are particularly interested in stark and unequivocal evidence of the model violating the martingale property beyond an expected error of any approximating model. We will analyse and quantify violations of the martingale property with diagnostics, which we introduce in §5.3.2, in order to check our hypothesis experimentally. In App. D.2.3 we derive the order of 'acceptable violations' for the test statistics we will introduce.

5.3.2 Diagnostics for the Martingale Property

As we showed in §5.2.4, the martingale property is fundamental to a Bayesian learning system. In this work, we probe the martingale property in LLMs via two properties *implied by* it. If these implied properties are strongly violated, so is the martingale property. More specifically, we will derive implications involving conditional expectations of the form $\mathbb{E}(f(Z_{n+1:n+m})|Z_{1:n})$, which can be estimated by generating sample paths $\{z^{(j)}_{n+1:n+m} \sim p_M(Z_{n+1:n+m}|Z_{1:n} = z_{1:n})\}_{j=1}^J$ autoregressively with an LLM, and use these samples to form Monte Carlo estimates of the conditional expectations. We begin with an equivalent characterisation of the (conditional) martingale property.

Proposition 5.1. A sequence $\{Z_{n+1:n+m}\} \sim p_M(\cdot|Z_{1:n})$ satisfies the martingale property if and only if the following holds: for all $n', k \in \mathbb{N}$ and integrable functions g, h:

$$\mathbb{E}((g(Z_{n'+k}) - g(Z_{n'+1}))h(Z_{n+1:n'})|Z_{1:n}) = 0. \tag{5.8}$$

We now state two implications of Proposition 5.1, our two diagnostics of the martingale property, which we will check experimentally in §5.4.

Corollary 5.1. Let $\{Z_i : i \in \mathbb{N}\}$ be a sequence of random variables satisfying the martingale property. Then for all integers $n, n', k > 0$ and $n' > n$ it holds that:

(i) $\mathbb{E}(g(Z_{n+1})|Z_{1:n}) = \mathbb{E}(g(Z_{n+k})|Z_{1:n})$ for all integrable functions g, and
(ii) $\mathbb{E}((Z_{n'+k+1} - Z_{n'+1})Z_{n'}^\top|Z_{1:n}) = 0$.

Properties (i) and (ii) are derived from Proposition 5.1 by making different choices of the functions (g, h). Property (i) follows by setting $h(Z_{n+1:n'}) \equiv 1$ and examines the marginal predictive distributions $p_M(Z_{n+k}|Z_{1:n})$. We instantiate (i) using (at most) two choices of g: In preview of §5.4, we will perform our checks on unconditional experiments where Z_i—or equivalently Y_i because of the unconditional setting—are Bernoulli or Gaussian distributed random variables. In the Bernoulli experiment it suffices to choose the identity function $g(z) = z$, as the mean $\mathbb{E}(Z_{n+k}|Z_{1:n})$ provides full information about the distribution $p_M(Z_{n+k}|Z_{1:n})$. In the Gaussian experiment, we will observe that choosing $g(z) = z$ and $g(z) = z^2$ is in most cases sufficient to reveal substantial violations from the martingale property.

Property (ii) is equivalent to requiring Eq. (5.8) to hold for all linear functions (g, h), which follows by linearity of the functions and the conditional expectation. We will again see in our experiments that this choice is usually sufficient to reveal deviations from the martingale property. Let us further consider our choices for h and g with an example.

Example 5.2. Suppose p_M is a Bayesian learning system over a latent parameter θ (see Eq. (5.7)), and the respective likelihood $p(Z|\theta)$ satisfies $\mathbb{E}_{Z\sim p(Z|\theta)}Z = \theta$. Then by Corollary 5.1, for all (k, n') we have

- $\mathbb{E}(Z_{n+k}|Z_{1:n}) = \mathbb{E}(\theta|Z_{1:n})$, and
- $\mathbb{E}(Z_{n'+k+1}Z_{n'+1}^\top|Z_{1:n}) = \mathbb{E}(\theta\theta^\top|Z_{1:n})$ [see e.g. 72, p. 454].

In this setting, condition (i) (with $g(z) = z$) and (ii) thus guarantee that the conditional mean and covariance equal the posterior mean and covariance, respectively, independent of the indices (n', k). These two important aspects of the posterior are hence consistently expressed by the model. The example is especially relevant as it covers Bernoulli ($p(Z|\theta) = \text{Bern}(\theta)$) and Gaussian data, which will be our main focus in the experiments.

In App. D.3 we present aggregated statistics $T_{1,g}$ and $T_{2,k}$ to compute and empirically measure properties (i) and (ii) from sample paths generated by an LLM. In our experiments, we check if these statistics lie within bootstrapped confidence intervals obtained by a reference Bayesian predictive model, which is readily available in synthetic settings, through the same sampling procedure. We will refer to these comparisons as 'checks' of the martingale property. If $T_{1,g}$ and $T_{2,k}$ lie outside the confidence interval, properties (i) and (ii) and hence the martingale property are violated.

5.3.3 Diagnostics for Epistemic Uncertainty

As discussed in §5.2.3, the martingale property allows us to identify epistemic uncertainty, which should decrease with more observed samples. Here, we derive a third diagnostic for Bayesian ICL systems which probes this. We begin by presenting a theoretical fact which provides important intuition on the role of epistemic uncertainty.

Fact 5.1. Let $\pi(\theta)$ and $p_M(Z|\theta)$ be the prior and likelihood of a Bayesian model, $\bar{\theta}_n := \mathbb{E}_{\theta\sim\pi(\theta|z_{1:n})}\theta$ the posterior mean given data $z_{1:n}$, and $\|\cdot\|$ be any vector norm. Then,

$$\mathbb{E}_{\theta_0\sim\pi, z_{1:n}\sim\pi(z|\theta_0)}\mathbb{E}_{\theta\sim\pi(\theta|z_{1:n})}\|\theta - \bar{\theta}_n\|^2 = \mathbb{E}_{\theta_0\sim\pi, z_{1:n}\sim\pi(z|\theta_0)}\|\theta_0 - \bar{\theta}_n\|^2. \tag{5.9}$$

The left-hand side in Eq. (5.9) is the trace of the posterior covariance (variance) and thus measures epistemic uncertainty. The right-hand side is the estimation error for the true parameter. Thus, Fact 5.1 states that *epistemic uncertainty provides a quantification for the average-case estimation error*. Note that Eq. (5.9) only applies to data from the prior predictive distribution, and thus not necessarily to the real observations. Nonetheless, a significant deviation of a model from the known scaling behaviour of the estimation error will indicate non-conformance with any reasonable Bayesian models. This is precisely our starting point to derive another diagnostic for Bayesian ICL systems.

As discussed in §5.2.3, we use sample paths generated by an LLM to approximate a martingale posterior and estimate its epistemic uncertainty. Here, we characterise epistemic uncertainty through the trace of the posterior covariance of the martingale posterior, the 'spread' of the distribution. Because the sample paths we use are finite (see §5.3.1) we cannot study the exact martingale posterior directly, which can only be recovered with infinite samples. Instead, we study the sampling distribution of the maximum likelihood estimate (MLE) on the first m samples: $\hat{\theta}_m := \arg\max_{\theta\in\Theta} \sum_{i=1}^{m} \log p_\theta(Z_{n+i})$, where p_θ is the known parametric likelihood. We measure the spread of this distribution using its *inter-quartile range*

$$T_3 = Q_{0.75}(\{\hat{\theta}_m^{(j)}\}_{j=1}^{J}) - Q_{0.25}(\{\hat{\theta}_m^{(j)}\}_{j=1}^{J}), \tag{5.10}$$

where $\hat{\theta}_m^{(j)}$ denotes the MLE using the j-th sample path $\{z_{n+i}^{(j)}\}_{i=1}^{m}$, and $Q_{0.25}$ and $Q_{0.75}$ are the 0.25- and 0.75-quantiles. In our experiments in §5.4 we consider scenarios where the true data distribution is defined by regular parametric models. In such cases the optimal (squared) estimation error for the true parameter scales $O(d/n)$ where n is the ICL dataset size and d is the dimension of the parameter, which is also the minimax lower bound [224, Ch. 8]. When choosing $m = \Theta(n)$, a reference Bayesian model will also have the $O(d/n)$ scaling behaviour following classical posterior contraction results in statistics; see App. D.2.2. Therefore, we can compare the asymptotic scaling of T_3 between an LLM and a reference Bayesian parametric model through the same sampling-based procedure. If the

scaling behaviour of T_3 from our LLM deviates from that of the reference Bayesian model, we can conclude that the LLM either exhibits a marked loss of estimation efficiency, or does not maintain a correct notion of epistemic uncertainty at all. Both characteristics contradict a Bayesian ICL system and are undesirable.

5.4 Experimental Analysis on LLMs

In this section, we experimentally probe whether ICL in state-of-the-art LLMs is Bayesian using the diagnostics discussed in §5.3 and corresponding test statistics $T_{1,g}, T_{2,k}, T_3$. We provide our code base on `https://github.com/meta-inf/bayes_icl`.

5.4.1 Experiment Setup

We consider three types of synthetic datasets $z_{1:n}$:

- **Bernoulli**: $Z_i \sim \text{Bern}(\theta)$, where $\theta \in \{0.3, 0.5, 0.7\}$;
- **Gaussian**: $Z_i \sim \mathcal{N}(\theta, 1)$, where $\theta \in \{-1, 0, 1\}$;
- A synthetic **natural language** experiment representing a prototypical clinical diagnostic task, where $Z_i = (X_i, Y_i)$ indicate the presence or absence of a symptom and disease as a text string for the i-th patient, respectively. Further, $X_i \sim \text{Bern}(0.5), Y_i|X_i \sim \text{Bern}(0.3 + 0.4X_i)$.

On purpose, we reduce our experimental setup to these minimum viable test beds where the ground-truth latent parameters are known, stripping away the convoluted latent complexity of in-the-wild NLP data. We use the following LLMs: `llama-2-7B` with 7B parameters ([218]), `mistral-7B` ([106]), `gpt-3` [27] with 2.7B and 170B parameters, `gpt-3.5`, and `gpt-4` [176][2].

[2]We only use `gpt-4` in a subset of experiments (Fig. 5.3, Fig. 5.5 in the text) due to API and resource limitations (App. D.3.1).

In all experiments we compute test statistics on LLM samples, and compare their behaviour with the same statistics evaluated on samples from a reference Bayesian model. More specifically, in §5.4.2 we compare the statistics obtained from LLMs with the bootstrap confidence intervals (CIs) derived from the reference Bayesian model. A deviation will thus indicate that the LLM is unlikely to be a good approximation of the reference Bayesian model. More importantly, when n becomes moderately large, the Bernsten von-Mises theorem [224] applies: the deviations then imply that the LLM is highly likely deviating from all *reasonable Bayesian models*, namely those satisfying the regularity conditions of the theorem. This is because the theorem guarantees that the test statistics derived from all such models have asymptotically[3] equivalent distributions.

We refer to App. D.3.1 for additional experimental details, such as the prompt format, tokenization, and computational requirements, as well as additional experimental results.

5.4.2 Checking the Martingale Property

We first check if state-of-the-art LLMs satisfy the martingale property. As we discussed in §5.2, this is a necessary condition for an exchangeable Bayesian ICL system.

Bernoulli experiment. Fig. 5.3 reports the results of the Bernoulli experiments with $n = 50$ observed samples, LLM sample paths of length $m \in \{n/2, 2n\}$, and datasets with ground-truth mean $\theta \in \{.3, .5, .7\}$. As discussed in §5.3.2 and §5.4.1 above, we compute the test statistics $T_{1,g}$ and $T_{2,k}$ on J sample paths generated by an LLM, and compare them with bootstrap CIs (of high confidence, see scale of y-axis) obtained from a reference Bayesian model. Here we define the reference model using a Bernoulli likelihood and a non-informative Beta$(1, 1)$ prior.

[3]We note that the asymptotic equivalence results are relevant in our setting. As a concrete example, in the setting of Fig. 5.3 (a), the CIs obtained by using Beta$(1, 11)$ and Beta$(1, 1)$ as the reference model are practically indistinguishable; the difference is on the order of 10^{-4}.

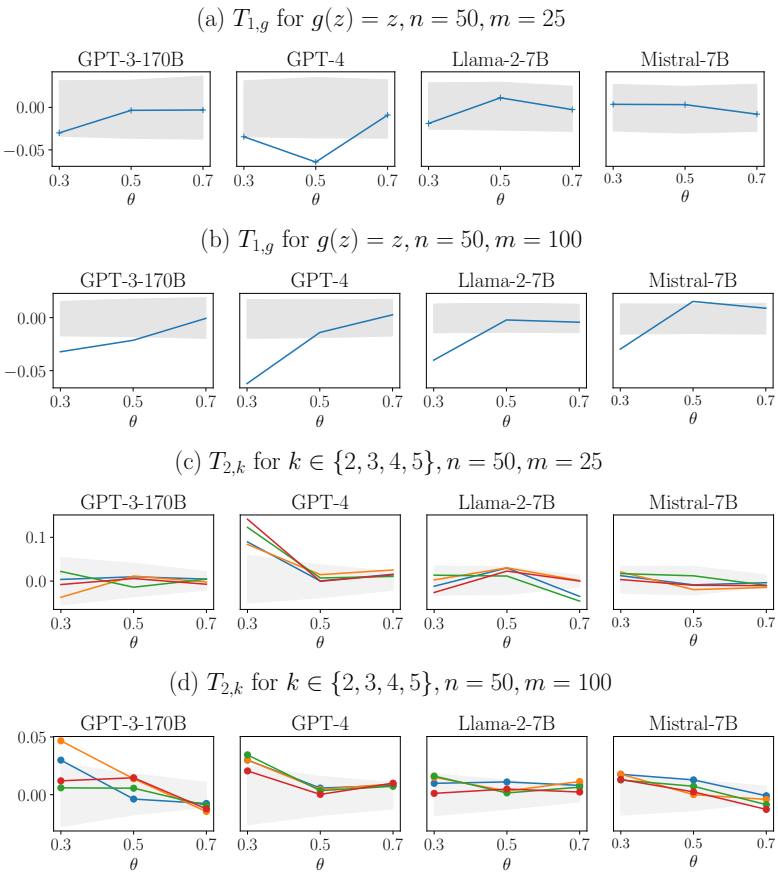


Figure 5.3: Checking the martingale property on Bernoulli experiments. Each data point represents a test statistic (y-axis) evaluated for an LLM, as derived in §5.3.2. Subplot and x-axis correspond to choices of Bernoulli probabilities and LLMs. Shade indicates the 95% confidence interval from a reference Bayesian model.

For short sample paths of length $m = n/2$ (subplots (a) and (b)), most LLMs lead to test statistics that are generally within the respective CIs, with the main exception being `gpt-4` ($\theta \in \{0.3, 0.5\}$), indicating a mostly adherence to the martingale property. However, for longer sample paths with $m = 2n$ (subplots (c) and (d)), more frequent deviations from the CIs are observed. For brevity, full results for other choices of n and LLMs are deferred to App. D.3.2. The findings are generally consistent across all choices of n. We also observe `gpt-3.5` to perform better than `gpt-4` but worse than `gpt-3-170b`. As we discuss in App. D.3.2 the latter observation may be explained by the fact that `gpt-3.5` and `gpt-4` have undergone instruction tuning [177]. In summary, in the Bernoulli experiments the LLMs generally adhere to the martingale property in short sampling horizons, but in

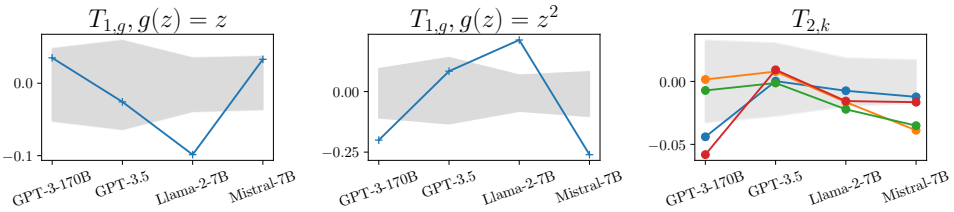
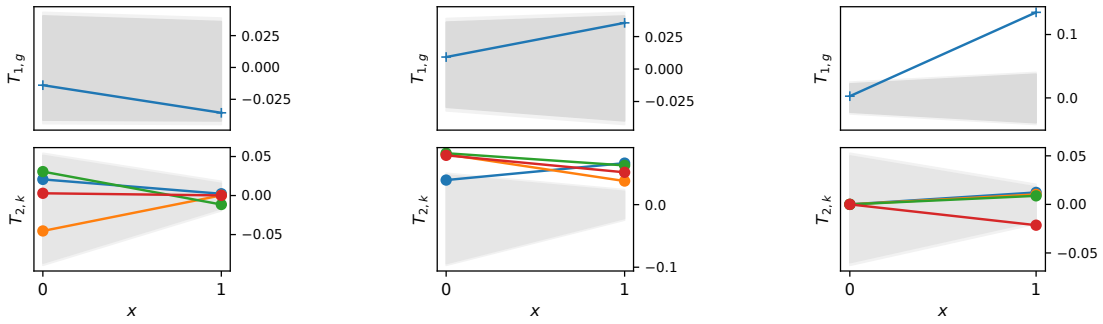


Figure 5.4: Checking the martingale property on Gaussian experiments. We present runs with $\theta = -1, n = 100, m = 50$ from different LLMs (x-axis) with test functions $g(z) = z$ and $g(z) = z^2$. See Fig. 5.3 for further details.

longer horizons demonstrate a significant deviation from the martingale property and hence the Bayesian principle.

Gaussian experiment. In Fig. 5.4 we present results on the Gaussian experiment with $\theta = -1, n = 100, m = n/2$, again performing both checks of the martingale property and using a reference Bayesian model with the non-informative prior $\mathcal{N}(0, 100)$. As we can see, all models except `gpt-3.5` demonstrate clear deviation from the martingale property. Additional results for `gpt-3.5` in App. D.3 present our diagnostics with other choices of (n, m, θ), demonstrating a deviation from the predictive distribution of the reference Bayesian posterior. In conclusion, the presented evidence on the Gaussian experiment falsifies our hypothesis of Bayesian behaviour with the tested LLMs.

Synthetic natural language experiment. In Fig. 5.5 we present our results for the natural language experiment with $n = 80, m = 40, g(z) = z$ using the GPT models. Here, we compute the test statistics on samples separated by the Bernoulli-distributed value of X_i (see App. D.3.1 for details). As we can see, both `gpt-3.5` and `gpt-4` demonstrate deviation from a reference Bayesian posterior. This provides further evidence of violations of the martingale property in settings where natural language (instead of numbers) is used.



(a) GPT-3-170B **(b)** GPT-3.5 **(c)** GPT-4

Figure 5.5: Checking the martingale property on the natural language experiment. We present both checks with test statistics computed separately for each value of X_i (x-axis). See Fig. 5.3 for further details.

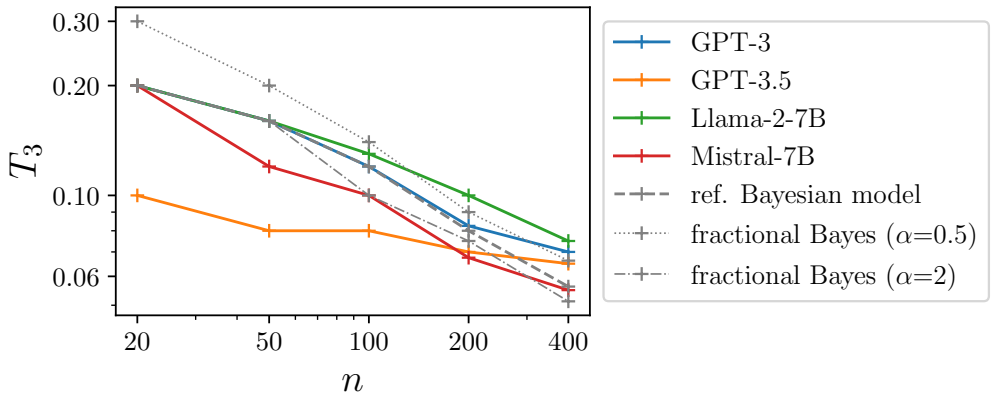


Figure 5.6: Scaling of epistemic uncertainty on the Bernoulli experiment: the test statistic T_3 (§5.3.3) computed on LLMs, compared with Bayesian and fractional Bayesian models.

5.4.3 Checking Epistemic Uncertainty of LLMs

In this subsection we analyse the scaling behaviour of an LLM's uncertainty. In Fig. 5.6 we measure T_3 (y-axis on a log-scale) and compare the approximate martingale posterior of an LLM with a reference Bayesian model when increasing the number of observed samples n (x-axis). We consider a Bernoulli experiment with $\theta = 0.5$ as it is the only experimental setting where, with a short sampling horizon of $m = n/2$, all LLMs approximately adhere to the martingale property. In addition to the standard reference Bayesian model, we also consider two α-fractional Bayesian posteriors [18], which are generalisations of the Bayesian posterior that

exhibit a $O(d/\alpha n)$ scaling for its epistemic uncertainty. They allow us to check the weaker hypothesis whether an LLM's epistemic uncertainty scales at least up to the correct order of magnitude.

We observe that the asymptotic rate of `llama-2-7b` and `gpt-3.5` is slower than that of a Bayesian model, which suggests inefficiency as discussed in §5.3.3. Furthermore, `gpt-3.5` demonstrates over-confidence in the small-sample regime. The scaling of `gpt-3-170b` and `mistral-7b` are closer to the Bayesian model, even though not exactly matching the latter. This finding is interesting as on the Bernoulli experiments, `gpt-3-170b` and `mistral-7b` also demonstrate the best adherence to the martingale property.

5.5 Conclusion

In this work we stated the martingale property as a fundamental requirement of a Bayesian learning system for exchangeable data, and discussed its desirable consequences if satisfied by an LLM. Based on this property we derived three different diagnostics that allowed us to check whether LLMs adhere to the Bayesian principle on synthetic in-context learning tasks. We presented stark evidence that state-of-the-art LLMs violate the martingale property, and hence falsified the hypthesis that ICL in LLMs is Bayesian.

Our investigation is particularly relevant to a recent line of work that investigates LLM-based ICL for tabular data modelling: for prediction on noisy tabular datasets [155, 245], the martingale property would enable us to diagnose the predictive uncertainty; and for synthetic data generation [23, 83, 227], it is vital to ensuring valid inference based on imputations of missing data (§5.2.2). It is thus of practical interest to develop models that better adhere to the martingale property.

The primary limitation of our work is the (intentional) restriction to small-scale, synthetic datasets, which are different from common NLP applications. We note that while our diagnostics are designed for synthetic problems, they reflect a broader principle: Bayesian epistemic uncertainty can be extracted from black-box models by examining the correlation structure in sequential predictions. This is clearly shown by the variance estimator in Example 5.2, and by the fact that MLE on sampled paths approximates the Bayesian posterior (§5.3.3). Future work could investigate generalisations of this approach.

More broadly, the RCT example in §5.1 can arguably be viewed as the simplest type of decision task involving multi-step reasoning, as the right decision (here based on an average treatment effect) is only naturally determined after imputing all missing samples. Thus, it would be interesting to investigate analogies to the hallucination behaviour we have identified for ICL in more complex reasoning tasks such as those involving chain-of-thought prompting [235]. Lastly, it may be worth to consider fine-tuning objectives to achieve an idealised Bayesian behaviour with a model after pretraining, but before deployment.

Acknowledgments and Disclosure of Funding

Fabian Falck acknowledges the receipt of studentship awards from the Health Data Research UK-The Alan Turing Institute Wellcome PhD Programme (Grant Ref: 218529/Z/19/Z). Ziyu Wang acknowledges support from Novo Nordisk. Chris Holmes acknowledges support from the Medical Research Council Programme Leaders award MC_UP_A390_1107, The Alan Turing Institute, Health Data Research, U.K., and the U.K. Engineering and Physical Sciences Research Council through the Bayes4Health programme grant.

This research is supported by research compute from the Baskerville Tier 2 HPC service. Baskerville is funded by the EPSRC and UKRI through the World Class

Labs scheme (EP/T022221/1) and the Digital Research Infrastructure programme (EP/W032244/1) and is operated by Advanced Research Computing at the University of Birmingham. We further acknowledge the receipt of OpenAI API credits through the OpenAI Researcher Access Program.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	Is In-Context Learning in Large Language Models Bayesian? A Martingale Perspective
Publication Status	x Published □ Accepted for Publication □ Submitted for Publication □ Unpublished and unsubmitted work written in a manuscript style
Publication Details	Is In-Context Learning in Large Language Models Bayesian? A Martingale Perspective. Fabian Falck*, Ziyu Wang*, Chris Holmes. International Conference on Machine Learning (ICML) 2024. Also presented at Secure and Trustworthy Large Language Models workshop at ICLR 2024 (oral). * indicates equal contribution.

Student Confirmation

Student Name:	Fabian Falck		
Contribution to the Paper	I follow the CRediT contributor role taxonomy (Brand et al., 2015). Fabian Falck: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing - Original Draft, Writing - Review & Editing, Visualization Ziyu Wang: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing - Original Draft, Writing - Review & Editing, Visualization Chris Holmes: Conceptualization, Methodology, Resources, Writing - Review & Editing, Supervision, Project administration, Funding acquisition		
Signature		Date	September 29, 2024

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof. Chris Holmes		
Supervisor comments		
Signature 	Date	November 27, 2024

This completed form should be included in the thesis, at the end of the relevant chapter.

6

Conclusion

Contents

This thesis analysed existing and proposed novel inductive biases for generative models. Covering several generative model classes, including VAEs, HVAEs, diffusion models and autoregressive models, we extended the research literature by theoretically characterising aspects which are crucial for making these models work well, and exploited this insight to derive ways to improve their applicability, stability, and performance for generative tasks.

6.1 Discussion

In Chapter 2 we presented MFCVAE, a novel VAE for the purpose of finding structure in high-dimensional data by uncovering multiple ways in which the data can be clustered. This model is useful in exploratory analysis of high-dimensional data where we—often guided by prior knowledge—assume the presence of multiple clusterings in a dataset, each containing distinct clusters. We achieved this with a VAE featuring a multiple MoG prior, whose generative model is the assumed generative process of the data. We stabilised its training through the combination of a VLAE neural architecture, in which each latent level's neural network has a different expressivity, and a progressive training algorithm, which learns the facets (i.e. clusterings) sequentially.

The method we proposed is applicable to any type of data, however, our experiments are limited to images. While this is a common limitation of many works in deep

clustering, specific choices such as the neural architecture and the progressive training algorithm, which both exploit the inherently hierarchical, multi-resolution nature of images, are not guaranteed to translate well to other modalities out of the box. Another limitation is MFCVAE's 'duplication' of clusters: sometimes multiple clusters represent the same abstract characteristic (e.g. the combination of 'cube' and 'blue floor hue' in 3DShapes). A technique which made the clusters unique may overcome these identifiability issues [122], and we leave this for future work.

Hyperparameter tuning for this method can require effort: in addition to the stabilising techniques mentioned above, finding the right training and loss parameters is crucial for the method to perform well, and sometimes small adjustments to the hyperparameters can make the method deteriorate between different random seeds (while it is stable between random seeds if the right hyperparameters have been found). In this case the method may collapse to a single facet, or a single cluster within one facet. Furthermore, by definition of the task, there are no ground-truth labels which can be used to optimise the method. This applies in particular to crucial parameters such as the number of clusterings and the number of clusters in each clustering which need to be defined a priori. We envision a more holistic algorithm which automatises hyperparameter search for multi-facet clustering during an exploratory data analysis. This could for instance be realised by simultaneously training multiple clustering models and selecting the best model based on their likelihood of the data, or problem-specific heuristics suited for exploring the data and gaining novel insight into it.

In Chapter 3 we analysed and expanded the understanding of the inductive bias of HVAEs. We showed a conjugacy between average pooling and projection to a lower resolution Haar wavelet subspace up to a change of basis, connecting the U-Net like architecture of HVAEs with Haar wavelet compression. Furthermore, by characterising HVAE cells as discretisations of a diffusion SDE, and their U-Net like architecture as discretisations of a multi-resolution diffusion process, we identified an important connection between HVAEs and diffusion models. This insight expands

our understanding of the relation between the two model classes, and allows future work to further analyse and explore the potential interpolation between them.

A well-known issue when training HVAEs is posterior collapse, the phenomenon where the posterior distribution exactly equals the prior distribution, nullifying the contribution of the respective KL-divergence in the loss. Posterior collapse typically results in deteriorating model performance. In HVAEs, the phenomenon is particularly distinct due to the large number of KL divergences in the loss, one for each latent variable in the hierarchy. This empirically manifests as all but one of the KL divergences collapsing at some point during training. While we tried to theoretically (e.g. in Theorem 5) and empirically (e.g. through gradient clipping) explain and overcome these instabilities, HVAEs keep deteriorating at times during training without a known root cause. Posterior collapse hence remains an open problem.

This project opens up several potential research directions for HVAEs. An obvious next step of this work is to design HVAE cells which implement other, potentially higher-order SDE discretisations. A second idea is to analyse the benefits of weight-sharing in HVAEs further. Even though we found weight-sharing to be beneficial for parameter efficiency, the number of latent layers that can share weights without diminishing performance was typically small (about 3 to 6). Implementing more modern neural network layers such as attention or making use of other advances which enable weight-sharing across all (time) steps in diffusion models might overcome this limitation.

More generally, when compared to diffusion models, HVAEs are currently a largely abandoned stream of research. There are mainly empirical reasons for this due to better performance and training stability of diffusion models, however, there is no fundamental reason why HVAEs, arguably the more general model class, could not outperform diffusion models. Future work should further explore their relation and provide further insight on how they compare [173].

In Chapter 4 we rigorously analysed and extended U-Nets. We provided a definition for U-Nets which allowed us to characterise preconditioning as the key design principle of U-Nets, the role of the encoder and decoder operators, their high-resolution scaling limits, and their conjugacy to ResNets. Based on this mathematical framework for U-Nets, we proposed a novel type of U-Net, a Multi-ResNet, which has a parameter-free encoder, and demonstrated the efficiency and performance of this model on PDE surrogate modelling and image segmentation. In the context of diffusion models, we identified how high-frequency components under the lens of a Haar wavelet basis are dominated by noise exponentially faster than low frequency components, and how U-Nets exploit this inductive bias of diffusion models.

This work paves the way to further analysis of U-Nets within our framework. It enables the design of U-Net architectures capturing problem-specific constraints, such as a natural basis for a PDE problem, a boundary constraint, or the topology of a problem (e.g. data on a sphere or triangle). It allows us to understand the inductive bias of U-Nets better and improve it for specific generative models, for instance diffusion models, in future work. Most importantly, this work is an attempt to ever so slightly open and understand the black box of the neural network used within generative models, which is their key ingredient, yet it is often poorly understood.

One important limitation is our focus on Haar wavelets as the function (sub-)spaces of choice. While this focus is due to the frequent use of average pooling as the downsampling operation in U-Nets on imaging data, which is conjugate to Haar wavelet projection, other function spaces natural for specific data or problems should be explored, particularly for Multi-ResNets. Further, this work has demonstrated how to satisfy functional constraints in a U-Net, such as boundary conditions of a PDE, but we have not explored its benefit experimentally. Lastly, the extension to apply U-Nets on non-square domains which this work enables is—with the exception of a proof-of-concept on a triangular domain—left for future work.

In Chapter 5 we analysed whether in-context learning in LLMs follows Bayesian principles. By investigating the martingale property, a necessary condition for a Bayesian learning system, in LLMs, we found that state-of-the-art LLMs consistently violate this property, and hence do not show Bayesian behaviour. This has important implications for the use of LLMs in safety-critical applications where principled uncertainty estimates are essential, and for the properties of LLMs for reasoning more generally.

Compared with industrial-scale LLM research, the experiments in this work were performed on relatively small and synthetic data. Furthermore, the prior and likelihood distributions were assumed to be known. Extending the methodology and experiments to large-scale, real-world data, which LLMs are normally applied on, would be a valuable extension of our work. Moreover, as LLMs are an active field of research with new model versions appearing regularly, it will be interesting to see how LLMs evolve with respect to the Bayesian properties we proposed, which could in future iterations serve as potential evaluation metrics. Lastly, analysing the effect of advanced inference-time techniques such as chain-of-thought prompting [235] and fine-tuning which encourage the model's adherence to Bayesian learning principles would be an interesting direction for future work.

6.2 Outlook

Generative models are a rapidly evolving area of research. Generative approaches for numerous applications are proposed every year. These generative approaches are typically a complex interplay of multiple components crucial for the model's success. A non-exhaustive list includes their training and sampling algorithm (including potential training stages, such as pretraining and fine-tuning), the hyperparameters, the loss function, the neural architecture, and safety tuning. Each of these components has specific instantiations in large-scale models. Their development is typically driven by empirical performance on evaluation metrics and

is highly tuned for a specific model instance. This possibly results in local minima whose insights are not always influencing other instantiations within the same model class, let alone models from another class. This thesis complements this important, highly influential stream of work at the frontier by contributing theoretical and analytical insight into the inductive biases of selected core components of generative models, with a view to translate these into empirical value.

Generative model classes and their inductive biases are often perceived as entirely disjunct from one another. This is reinforced by the observation that there is often exactly one model class which outperforms all others on certain applications or data modalities, such as diffusion models on images and LLMs on text. However, the utility and inductive bias of some of their components (such as the U-Net architecture) is transferable across generative models, as we have shown for HVAEs and diffusion models in Chapter 3. To give another example, Kingma et al. showed that many commonly used loss functions in diffusion models can be viewed as an ELBO with data augmentation, which relates this important component of VAEs, HVAEs and diffusion models. Furthermore, they also demonstrated that the loss of diffusion models with a certain weighting is equivalent to the loss of a specific, frequently used variant of flow matching models [143], a recent contender of diffusion models [117]. These results demonstrate that we can derive theoretical results which uncover close connections between model classes, a line of work which this thesis contributed to.

Future work should further explore the relations and interpolations of generative models and their inductive bias. We should work towards a unifying theory of generative models, which identifies the foundational design principles and inductive biases of generative models, and understands when and why they work. This theory shall provide the mathematical framework for designing novel inductive biases of future generative models.

Appendices

A

Appendix of *Multi-Facet Clustering Variational Autoencoders*

Contents

A.1 J Independent Mixture of Gaussians prior on z

Let $p(\boldsymbol{z}_j)$ be the marginal distribution of $\boldsymbol{z}_j$ as follows

$$p(\boldsymbol{z}_j) = \sum_{c_j=1}^{K_j} p(c_j, \boldsymbol{z}_j) \tag{A.1}$$

$$= \sum_{c_j=1}^{K_j} p(c_j) p(\boldsymbol{z}_j | c_j) \tag{A.2}$$

$$= \sum_{c_j=1}^{K_j} p(c_j) \mathcal{N}(\boldsymbol{z}_j | \boldsymbol{\mu}_{c_j}, \boldsymbol{\Sigma}_{c_j}) \tag{A.3}$$

where $p(c_j)$ is a categorical distribution. Thus, $p(\boldsymbol{z}_j)$ is a Mixture-of-Gaussians (MoG).

Let us now derive $p(\vec{\mathbf{z}})$, the marginal distribution of $\vec{\mathbf{z}}$, as follows

$$p(\vec{\mathbf{z}}) = p(\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_J) \tag{A.4}$$

$$= \prod_{j=1}^{J} p(\boldsymbol{z}_j) \tag{A.5}$$

$$= \prod_{j=1}^{J} \sum_{c_j=1}^{K_j} p(c_j) \mathcal{N}(\boldsymbol{z}_j | \boldsymbol{\mu}_{c_j}, \boldsymbol{\Sigma}_{c_j}) \tag{A.6}$$

where Eq. (A.5) follows from the indepdendence assumption of facets, and Eq. (A.6) uses Eq. (A.3). The resulting marginal of $\vec{\mathbf{z}}$ is our prior of J independent MoGs.

Linear (rather than exponential) complexity of number of clusters. Besides its representational advantages, the multi-facet prior structure features a computational advantage: Given multiple known partitions of a dataset, the total number of clusters over all facets required to represent these partitions scales *linearly* w.r.t. the number of such partitions. In comparison, the number of clusters required in a single-facet model suffers from combinatorial explosion and scales *exponentially*.

To understand this, let us consider a hypothetical multi-partition image dataset of (rather standardised) hotel rooms which features J facets $C_1, C_2, \ldots, C_J$ with

K_j possible discrete values for each characteristic, for example, the colour of the bed sheets, walls, interiors, whether a phone is present or not, the view of the room (beach, forest, city, ...). We now attempt to find reasonable clusters in this dataset. In principle, a single-partition model could learn all cross-combinations $C_1 \times C_2 \times \cdots \times C_J$. In general, this requires to learn "at least" $\mathcal{O}(\prod_{j=1}^{J} K_j)$ latent clusters [1]. Compare this with a multi-partition model such as MFCVAE. Here, we need to learn "at least" $\mathcal{O}(\sum_{j=1}^{J} K_j)$. If $K_j = K$ is equally large for all facets, the number of latent clusters to learn is $\mathcal{O}(K^J)$ for a single-partition model and $\mathcal{O}(K \cdot J)$ for a multi-partition model.

A.2 VaDE Trick Proofs

A.2.1 Single-Facet VaDE Trick

Proof. (Theorem 2.1: Single-Facet VaDE Trick)

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \mathrm{KL}\left[q_\phi(c|\mathbf{x})||p_\theta(c|\mathbf{z})\right] = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \mathbb{E}_{q_\phi(c|\mathbf{x})} \log \frac{q_\phi(c|\mathbf{x})}{p_\theta(c|\mathbf{z})} \tag{A.7}$$

$$= \mathbb{E}_{q_\phi(c|\mathbf{x})} \log \frac{q_\phi(c|\mathbf{x})}{\exp(\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(c|\mathbf{z}))} \tag{A.8}$$

$$= \mathrm{KL}\left[q_\phi(c|\mathbf{x})||\boldsymbol{\pi}(c|q_\phi(\mathbf{z}|\mathbf{x}))\right] - \log Z(q_\phi(\mathbf{z}|\mathbf{x})) \tag{A.9}$$

which is minimised w.r.t. $q_\phi(c|\mathbf{x})$ by setting the KL term to zero by $q_\phi(c|\mathbf{x}) = \boldsymbol{\pi}(c|q_\phi(\mathbf{z}|\mathbf{x}))$, where

$$\boldsymbol{\pi}(c|q_\phi(\mathbf{z}|\mathbf{x})) := \frac{\exp\left(\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(c|\mathbf{z})\right)}{Z(q_\phi(\mathbf{z}|\mathbf{x}))} \quad \text{for } c = 1, \ldots, K \tag{A.10}$$

$$Z(q_\phi(\mathbf{z}|\mathbf{x})) := \sum_{c=1}^{K} \exp\left(\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(c|\mathbf{z})\right) . \tag{A.11}$$

as required. Here, $Z(q_\phi(\mathbf{z}|\mathbf{x}))$ is the appropriate normalization constant for Eq. (A.10) to define a probability mass function. □

[1] This assumes that every cluster in the latent space corresponds to exactly one cross-combination of the data facets. Empirically, we find that for statistical reasons ("having more shots"), it can be desirable to have more latent clusters per facet than values possible for each facet.

Misapprehension in original statement

In the original paper, [108], they reach Eq. (2.5):

$$\mathcal{L}(\mathbf{x};\theta,\phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - \mathrm{KL}\left[q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})\right] - \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \mathrm{KL}\left[q_\phi(c|\mathbf{x})||p_\theta(c|\mathbf{z})\right]}_{\text{Ⓐ}}.$$

The claim made (appendix A of [108]) is that $q(c|\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}'|\mathbf{x})}\, p(c|\mathbf{z}')$ makes the final term, $\text{Ⓐ} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \mathrm{KL}\left[q_\phi(c|\mathbf{x})||p_\theta(c|\mathbf{z})\right]$, equal to zero. Substituting this form for $q_\phi(c|\mathbf{x})$ in Eq. (A.12), we get:

$$\begin{aligned}
\text{Ⓐ} &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \mathrm{KL}\left[q_\phi(c|\mathbf{x})||p_\theta(c|\mathbf{z})\right] \\
&= \int \mathrm{d}\mathbf{z}\, q_\phi(\mathbf{z}|\mathbf{x}) \sum_{c=1}^{K} q_\phi(c|\mathbf{x}) \log \frac{q_\phi(c|\mathbf{x})}{p_\theta(c|\mathbf{z})} \\
&= \int \mathrm{d}\mathbf{z}\, q_\phi(\mathbf{z}|\mathbf{x}) \sum_{c=1}^{K} \mathbb{E}_{q_\phi(\mathbf{z}'|\mathbf{x})}\, p(c|\mathbf{z}') \log \frac{\mathbb{E}_{q_\phi(\mathbf{z}''|\mathbf{x})}\, p(c|\mathbf{z}'')}{p_\theta(c|\mathbf{z})} \qquad \text{(A.12)} \\
&= \int \mathrm{d}\mathbf{z}\, q_\phi(\mathbf{z}|\mathbf{x}) \sum_{c=1}^{K} \left(\int \mathrm{d}\mathbf{z}'\, q_\phi(\mathbf{z}'|\mathbf{x}) p(c|\mathbf{z}')\right) \log \frac{\int \mathrm{d}\mathbf{z}''\, q_\phi(\mathbf{z}''|\mathbf{x}) p(c|\mathbf{z}'')}{p_\theta(c|\mathbf{z})} \\
&= \sum_{c=1}^{K} \underbrace{\left(\int \mathrm{d}\mathbf{z}'\, q_\phi(\mathbf{z}'|\mathbf{x}) p(c|\mathbf{z}')\right)}_{\neq 0} \underbrace{\left[\log \int \mathrm{d}\mathbf{z}''\, q_\phi(\mathbf{z}''|\mathbf{x}) p(c|\mathbf{z}'') - \int \mathrm{d}\mathbf{z}\, q_\phi(\mathbf{z}|\mathbf{x}) \log p_\theta(c|\mathbf{z})\right]}_{\overset{?}{=}0} \qquad \text{(A.13)} \\
&\overset{?}{=} 0
\end{aligned}$$

In the above derivations, we use $\mathbf{z}$, $\mathbf{z}'$ and $\mathbf{z}''$ to mark separate occurrences of the variable $\mathbf{z}$ in different integrals. The first term in Eq. (A.13) is strictly positive. To satisfy the claim, the second term in Eq. (A.13) would have to be equal to zero for all $c \in \{1, \ldots, K\}$, which in general does not hold. We note that in the original codebase for [108], training is not done using the form of the ELBO as above, Eq. (2.5) with $\text{Ⓐ} = 0$, instead, using the general form where all terms are calculated.

Monte Carlo Sampling of the VaDE-Trick objective: These results and analysis raise the natural question: how is it that this misapprehension has lasted? Perhaps this is because of the following lucky accident when performing MC sampling.

If one substitutes the optimal forms of Theorem 2.1 back into the initial $\mathcal{L}$ and then estimates the resulting objective using a *single* MC sample from $\mathbf{z}$, then the resulting estimator *looks* like is an estimator of Eq (2.5) with the final term set to zero, that is:

$$\mathcal{L}(\mathbf{x};\theta,\phi) \stackrel{?}{=} \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - \mathrm{KL}\left[q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})\right].$$

Equivalently, in reverse, taking a *single* MC sample for $\mathbf{z}$ and using the above misapprehension as the training objective results in the same estimator as one gets from taking one MC sample for the true objective.

Let us push through the former of these, constructing the objective and MC estimator for the correct optimal objective:

$$\mathcal{L}(\mathbf{x};\theta,\phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})q_\phi(c|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z},c)}{q_\phi(\mathbf{z}|\mathbf{x})q_\phi(c|\mathbf{x})}\right] \tag{A.14}$$

$$= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\,\mathbb{E}_{q_\phi(c|\mathbf{x})} \log \frac{q_\phi(c|\mathbf{x})}{p_\theta(\mathbf{z},c)} \tag{A.15}$$

$$= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})] - \mathbb{E}_{q_\phi(c|\mathbf{x})} \log \frac{q_\phi(c|\mathbf{x})}{\exp(\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{z},c))} \tag{A.16}$$

$$= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})]$$
$$- \mathrm{KL}\left[q_\phi(c|\mathbf{x})||\pi(c|q_\phi(\mathbf{z}|\mathbf{x}))\right] + \log \vec{Z}(q_\phi(\mathbf{z}|\mathbf{x})) \tag{A.17}$$

where

$$\boldsymbol{\pi}(c|q_\phi(\mathbf{z}|\mathbf{x})) := \frac{\exp(\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{z},c))}{\vec{Z}(q_\phi(\mathbf{z}|\mathbf{x}))} \quad \text{for } c \in \mathcal{C} \tag{A.18}$$

$$\vec{Z}(q_\phi(\mathbf{z}|\mathbf{x})) := \sum_{c\in\mathcal{C}} \exp(\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{z},c)) \,. \tag{A.19}$$

Setting $q_\phi(c|\mathbf{x}) = \boldsymbol{\pi}(c|q_\phi(\mathbf{z}|\mathbf{x}))$ and substituting $\mathbf{z}^{(l)}$ for $l = 1,\ldots,L$ Monte Carlo samples from $q_\phi(\mathbf{z}|\mathbf{x})$:

$$\mathcal{L}(\mathbf{x};\theta,\phi) \approx \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(\mathbf{x}|\mathbf{z}^{(l)}) - \log q_\phi(\mathbf{z}^{(l)}|\mathbf{x})$$
$$+ \log \sum_{c\in\mathcal{C}} \exp\left(\frac{1}{L}\sum_{l=1}^{L} \log p_\theta(\mathbf{z}^{(l)},c)\right) \tag{A.20}$$

which reduces for $L = 1$ to

$$\mathcal{L}(\mathbf{x};\theta,\phi) \approx \log p_\theta(\mathbf{x}|\mathbf{z}^{(1)}) - \log q_\phi(\mathbf{z}^{(1)}|\mathbf{x}) + \log p_\theta(\mathbf{z}^{(1)}). \tag{A.21}$$

This ***appears*** to be a MC estimator for

$$\mathcal{L}(\mathbf{x};\theta,\phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - \mathrm{KL}\left[q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})\right]. \tag{A.22}$$

This appearance is purely because the $\log \sum \exp \sum \log$ in Eq (A.20) luckily simplifies when $L = 1$. Because of this lucky coincidence, all empirical results in [108] are valid.

A.2.2 Multi-Facet VaDE Trick (factorised distribution)

Proof. (Theorem 2.2: Multi-Facet VaDE Trick for factorised distribution $q_\phi(\vec{\mathbf{z}}|\mathbf{x}) = \prod_j q_\phi(\mathbf{z}_j|\mathbf{x})$)

$$\mathbb{E}_{q_\phi(\vec{\mathbf{z}}|\mathbf{x})} \mathrm{KL}\left[q_\phi(\mathbf{c}|\mathbf{x})||p_\theta(\mathbf{c}|\vec{\mathbf{z}})\right] = \mathbb{E}_{q_\phi(\vec{\mathbf{z}}|\mathbf{x})} \mathbb{E}_{q_\phi(\mathbf{c}|\mathbf{x})} \log \frac{q_\phi(\mathbf{c}|\mathbf{x})}{p_\theta(\mathbf{c}|\vec{\mathbf{z}})} \tag{A.23}$$

$$= \mathbb{E}_{q_\phi(\mathbf{c}|\mathbf{x})} \log \frac{q_\phi(\mathbf{c}|\mathbf{x})}{\exp(\mathbb{E}_{q_\phi(\vec{\mathbf{z}}|\mathbf{x})} \log p_\theta(\mathbf{c}|\vec{\mathbf{z}}))} \tag{A.24}$$

$$= \mathbb{E}_{q_\phi(\mathbf{c}|\mathbf{x})} \log \frac{q_\phi(\mathbf{c}|\mathbf{x})}{\exp(\sum_j \mathbb{E}_{q_\phi(\mathbf{z}_j|\mathbf{x})} \log p_\theta(c_j|\mathbf{z}_j))} \tag{A.25}$$

$$= \mathbb{E}_{q_\phi(\mathbf{c}|\mathbf{x})} \log \frac{q_\phi(\mathbf{c}|\mathbf{x})}{\prod_j \exp(\mathbb{E}_{q_\phi(\mathbf{z}_j|\mathbf{x})} \log p_\theta(c_j|\mathbf{z}_j))} \tag{A.26}$$

$$= \mathrm{KL}\left[q_\phi(\mathbf{c}|\mathbf{x})|| \prod_j \boldsymbol{\pi}_j(c_j|q_\phi(\mathbf{z}_j|\mathbf{x}))\right] - \sum_j \log Z_j(q_\phi(\mathbf{z}_j|\mathbf{x})) \tag{A.27}$$

which is minimised w.r.t. $q_\phi(\mathbf{c}|\mathbf{x})$ by setting the KL term to zero by $q_\phi(\mathbf{c}|\mathbf{x}) = \prod_j \boldsymbol{\pi}_j(c_j|q_\phi(\mathbf{z}_j|\mathbf{x}))$, where

$$\boldsymbol{\pi}_j(c_j|q_\phi(\mathbf{z}_j|\mathbf{x})) := \frac{\exp(\mathbb{E}_{q_\phi(\mathbf{z}_j|\mathbf{x})} \log p_\theta(c_j|\mathbf{z}_j))}{Z_j(q_\phi(\mathbf{z}_j|\mathbf{x}))} \quad \text{for } c_j = 1, \ldots, K_j \tag{A.28}$$

$$Z_j(q_\phi(\mathbf{z}_j|\mathbf{x})) := \sum_{c_j=1}^{K_j} \exp(\mathbb{E}_{q_\phi(\mathbf{z}_j|\mathbf{x})} \log p_\theta(c_j|\mathbf{z}_j)) \ . \tag{A.29}$$

where $Z_j(q_\phi(\mathbf{z}_j|\mathbf{x}))$ is a normalisation constant for $\boldsymbol{\pi}_j(c_j|q_\phi(\mathbf{z}_j|\mathbf{x}))$, and we have used the relations

$$q_\phi(\vec{\mathbf{z}}|\mathbf{x}) = \prod_j q_\phi(\mathbf{z}_j|\mathbf{x}) \tag{A.30}$$

$$p_\theta(\mathbf{c}|\vec{\mathbf{z}}) = \prod_j p_\theta(c_j|\mathbf{z}_j) \tag{A.31}$$

as required. □

A.2.3 Multi-Facet VaDE Trick (general distribution)

While we use a posterior over $\vec{\mathbf{z}}$ that factorises between facets, $q_\phi(\vec{\mathbf{z}}|\mathbf{x}) = \prod_{j=1}^{J} q_\phi(\mathbf{z}_j|\mathbf{x})$, there is the question as to whether one can use a VaDE trick in the case where the posterior for $\vec{\mathbf{z}}$ has a general factorisation (e.g. an autoregressive factorisation over facets). An example would be $q_\phi(\vec{\mathbf{z}}|\mathbf{x}) = \prod_{j=1}^{J} q_\phi(\mathbf{z}_j|\mathbf{z}_{<j}, \mathbf{x})$, the posterior factorisation used in many hierarchical VAEs [34, 119, 221]. We answer this question in the affirmative:

Theorem A.1. *(Multi-Facet VaDE Trick for general $q_\phi(\vec{\mathbf{z}}|\mathbf{x})$, $p(\vec{\mathbf{z}}, \mathbf{c})$) For any probability distribution $q_\phi(\vec{\mathbf{z}}|\mathbf{x})$, the distribution $q_\phi(\mathbf{c}|\mathbf{x})$ that minimises $\mathbb{E}_{q_\phi(\vec{\mathbf{z}}|\mathbf{x})} \mathrm{KL}\left[q_\phi(\mathbf{c}|\mathbf{x})||p_\theta(\mathbf{c}|\vec{\mathbf{z}})\right]$ is*

$$\operatorname*{argmin}_{q_\phi(\mathbf{c}|\mathbf{x})} \mathbb{E}_{q_\phi(\vec{\mathbf{z}}|\mathbf{x})} \mathrm{KL}\left[q_\phi(\mathbf{c}|\mathbf{x})||p_\theta(\mathbf{c}|\vec{\mathbf{z}})\right] = \vec{\pi}(\mathbf{c}|q_\phi(\vec{\mathbf{z}}|\mathbf{x})) \tag{A.32}$$

where the minimum value is attained at

$$\min_{q_\phi(\mathbf{c}|\mathbf{x})} \mathbb{E}_{q_\phi(\vec{\mathbf{z}}|\mathbf{x})} \mathrm{KL}\left[q_\phi(\mathbf{c}|\mathbf{x})||p_\theta(\mathbf{c}|\vec{\mathbf{z}})\right] = -\log \vec{Z}(q_\phi(\vec{\mathbf{z}}|\mathbf{x})) \tag{A.33}$$

where

$$\vec{\pi}(\mathbf{c}|q_\phi(\vec{\mathbf{z}}|\mathbf{x})) := \frac{\exp(\mathbb{E}_{q_\phi(\vec{\mathbf{z}}|\mathbf{x})} \log p_\theta(\mathbf{c}|\vec{\mathbf{z}}))}{\vec{Z}(q_\phi(\vec{\mathbf{z}}|\mathbf{x}))} \quad \textit{for } \mathbf{c} \in \mathcal{C} \tag{A.34}$$

$$\vec{Z}(q_\phi(\vec{\mathbf{z}}|\mathbf{x})) := \sum_{\mathbf{c}\in\mathcal{C}} \exp(\mathbb{E}_{q_\phi(\vec{\mathbf{z}}|\mathbf{x})} \log p_\theta(\mathbf{c}|\vec{\mathbf{z}})) \ . \tag{A.35}$$

Proof.

$$\mathbb{E}_{q_\phi(\vec{\mathbf{z}}|\mathbf{x})} \mathrm{KL}\left[q_\phi(\mathbf{c}|\mathbf{x})||p_\theta(\mathbf{c}|\vec{\mathbf{z}})\right] = \mathbb{E}_{q_\phi(\vec{\mathbf{z}}|\mathbf{x})} \mathbb{E}_{q_\phi(\mathbf{c}|\mathbf{x})} \log \frac{q_\phi(\mathbf{c}|\mathbf{x})}{p_\theta(\mathbf{c}|\vec{\mathbf{z}})} \tag{A.36}$$

$$= \mathbb{E}_{q_\phi(\mathbf{c}|\mathbf{x})} \log \frac{q_\phi(\mathbf{c}|\mathbf{x})}{\exp(\mathbb{E}_{q_\phi(\vec{\mathbf{z}}|\mathbf{x})} \log p_\theta(\mathbf{c}|\vec{\mathbf{z}}))} \tag{A.37}$$

$$= \mathrm{KL}\left[q_\phi(\mathbf{c}|\mathbf{x})||\vec{\pi}(\mathbf{c}|q_\phi(\vec{\mathbf{z}}|\mathbf{x}))\right] - \log \vec{Z}(q_\phi(\vec{\mathbf{z}}|\mathbf{x})) \tag{A.38}$$

which is minimised by setting the KL term to zero as required. □

A.3 Monte Carlo estimator of Evidence Lower Bound

A.3.1 Primary form

We start the derivation from the ELBO in Eq. (2.4):

$$\mathcal{L}(\mathbf{x};\theta,\phi) = \mathbb{E}_{q_\phi(\vec{\mathbf{z}}|\mathbf{x})q_\phi(\mathbf{c}|\mathbf{x})}[\log \frac{p_\theta(\mathbf{x}|\vec{\mathbf{z}})p_\theta(\vec{\mathbf{z}},\mathbf{c})}{q_\phi(\vec{\mathbf{z}}|\mathbf{x})q_\phi(\mathbf{c}|\mathbf{x})}] \tag{A.39}$$

$$= \mathbb{E}_{q_\phi(\vec{\mathbf{z}}|\mathbf{x})q_\phi(\mathbf{c}|\mathbf{x})}[\log \frac{p_\theta(\mathbf{x}|\vec{\mathbf{z}})p_\theta(\vec{\mathbf{z}}|\mathbf{c})p_\theta(\mathbf{c})}{q_\phi(\vec{\mathbf{z}}|\mathbf{x})q_\phi(\mathbf{c}|\mathbf{x})}] \tag{A.40}$$

$$\begin{aligned} &= \mathbb{E}_{q_\phi(\vec{\mathbf{z}}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\vec{\mathbf{z}})\right] + \mathbb{E}_{q_\phi(\vec{\mathbf{z}}|\mathbf{x})q_\phi(\mathbf{c}|\mathbf{x})}\left[\log p_\theta(\vec{\mathbf{z}} \mid \mathbf{c})\right] \\ &\quad + \mathbb{E}_{q_\phi(\mathbf{c}|\mathbf{x})}\left[\log p_\theta(\mathbf{c})\right] - \mathbb{E}_{q_\phi(\vec{\mathbf{z}}|\mathbf{x})}\left[\log q_\phi(\vec{\mathbf{z}}|\mathbf{x})\right] - \mathbb{E}_{q_\phi(\mathbf{c}|\mathbf{x})}\left[\log q_\phi(\mathbf{c}|\mathbf{x})\right] \end{aligned} \tag{A.41}$$

Next, we note that Theorem 2 has the optimal value of $q(\mathbf{c}|\mathbf{x})$ taking the factorised form

$$q(\mathbf{c}|\mathbf{x}) = \prod_{j=1}^{J} q(c_j|\mathbf{x}). \tag{A.42}$$

Combining this with the factorised prior introduced in Eq. (2.2), the loss can then be simplified and approximated as

$$\begin{aligned} \mathcal{L}(\mathbf{x};\theta,\phi) \approx\ & \mathbb{E}_{q_\phi(\vec{\mathbf{z}}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\vec{\mathbf{z}})\right] + \sum_{j=1}^{J} \mathbb{E}_{q_\phi(\mathbf{z}_j|\mathbf{x})q_\phi(c_j|\mathbf{x})}\left[\log p_\theta(\mathbf{z}_j \mid c_j)\right] \\ &+ \sum_{j=1}^{J} \mathbb{E}_{q_\phi(c_j|\mathbf{x})}\left[\log p_\theta(c_j)\right] - \sum_{j=1}^{J} \mathbb{E}_{q_\phi(\mathbf{z}_j|\mathbf{x})}\left[\log q_\phi(\mathbf{z}_j|\mathbf{x})\right] - \sum_{j=1}^{J} \mathbb{E}_{q_\phi(c_j|\mathbf{x})}\left[\log q_\phi(c_j|\mathbf{x})\right] \end{aligned} \tag{A.43}$$

Then we approximate the ELBO using MC estimation by drawing samples from $q_\phi(\vec{\mathbf{z}}|\mathbf{x})$:

$$\begin{aligned} \tilde{\mathcal{L}}(\mathbf{x};\theta,\phi) =\ & \frac{1}{L}\sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\vec{\mathbf{z}}^{(l)}) + \frac{1}{L}\sum_{l=1}^{L}\sum_{j=1}^{J}\sum_{c_j=1}^{K_j} q_\phi(c_j|\mathbf{x}) \log p_\theta(\mathbf{z}_j^{(l)}|c_j) \\ &+ \sum_{j=1}^{J}\sum_{c_j=1}^{K_j} q_\phi(c_j|\mathbf{x}) \log p_\theta(c_j) - \frac{1}{L}\sum_{l=1}^{L}\sum_{j=1}^{J} \log q_\phi(\mathbf{z}_j^{(l)}|\mathbf{x}) - \sum_{j=1}^{J}\sum_{c_j=1}^{K_j} q_\phi(c_j|\mathbf{x}) \log q_\phi(c_j|\mathbf{x}) \end{aligned} \tag{A.44}$$

where the optimal value of $q_\phi(c_j|\mathbf{x})$ is obtained from Theorem 2.2:

$$q_\phi(c_j|\mathbf{x}) := \frac{\exp\left[\frac{1}{L}\sum_{l=1}^{L}\log p_\theta(c_j|\mathbf{z}_j^{(l)})\right]}{Z_j(q_\phi(\mathbf{z}_j|\mathbf{x}))} \quad \text{for } c_j = 1,\ldots,K_j \tag{A.45}$$

$$Z_j(q_\phi(\mathbf{z}_j|\mathbf{x})) := \sum_{c_j=1}^{K_j}\exp\left[\frac{1}{L}\sum_{l=1}^{L}\log p_\theta(c_j|\mathbf{z}_j^{(l)})\right] \ . \tag{A.46}$$

which reduces to

$$q_\phi(c_j|\mathbf{x}) = p_\theta(c_j|\mathbf{z}_j^{(1)}) \quad \text{for } j = 1,\ldots,J \tag{A.47}$$

when $L = 1$.

As a result, we obtain the loss for $L = 1$:

$$\begin{aligned}\tilde{\mathcal{L}}(\mathbf{x};\theta,\phi) = \log p_\theta(\mathbf{x}|\vec{\mathbf{z}}^{(1)}) + \sum_{j=1}^{J}\sum_{c_j=1}^{K_j} p_\theta(c_j|\mathbf{z}_j^{(1)})\Big(\log p_\theta(\mathbf{z}_j^{(1)}|c_j) + \log p_\theta(c_j)\Big) \\ - \sum_{j=1}^{J}\log q_\phi(\mathbf{z}_j^{(1)}|\mathbf{x}) - \sum_{j=1}^{J}\sum_{c_j=1}^{K_j} p_\theta(c_j|\mathbf{z}_j^{(1)})\log p_\theta(c_j|\mathbf{z}_j^{(1)})\end{aligned} \tag{A.48}$$

A.3.2 Alternate form

Again, we start the derivation of an MC estimator from Eq. (2.4):

$$\mathcal{L}(\mathbf{x};\theta,\phi) = \mathbb{E}_{q_\phi(\vec{\mathbf{z}}|\mathbf{x})q_\phi(\mathbf{c}|\mathbf{x})}[\log\frac{p_\theta(\mathbf{x}|\vec{\mathbf{z}})p_\theta(\vec{\mathbf{z}},\mathbf{c})}{q_\phi(\vec{\mathbf{z}}|\mathbf{x})q_\phi(\mathbf{c}|\mathbf{x})}] \tag{A.49}$$

$$= \mathbb{E}_{q_\phi(\vec{\mathbf{z}}|\mathbf{x})q_\phi(\mathbf{c}|\mathbf{x})}[\log\frac{p_\theta(\mathbf{x}|\vec{\mathbf{z}})p_\theta(\vec{\mathbf{z}})p_\theta(\mathbf{c}|\vec{\mathbf{z}})}{q_\phi(\vec{\mathbf{z}}|\mathbf{x})q_\phi(\mathbf{c}|\mathbf{x})}] \tag{A.50}$$

$$\begin{aligned}= \mathbb{E}_{q_\phi(\vec{\mathbf{z}}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\vec{\mathbf{z}}) - \log q_\phi(\vec{\mathbf{z}}|\mathbf{x}) + \log p_\theta(\vec{\mathbf{z}})\right] \\ + \mathbb{E}_{q_\phi(\mathbf{c}|\mathbf{x})}\Big[\mathbb{E}_{q_\phi(\vec{\mathbf{z}}|\mathbf{x})}\log p_\theta(\mathbf{c}\mid\vec{\mathbf{z}}) - \log q_\phi(\mathbf{c}|\mathbf{x})\Big]\end{aligned} \tag{A.51}$$

where an alternative factorisation of $p_\theta(\vec{\mathbf{z}},\mathbf{c})$ is used in Eq. (A.50), resulting in a different, but equivalent formulation of the ELBO in Eq. (A.41).

Next, we draw MC samples from $q_\phi(\vec{\mathbf{z}}|\mathbf{x})$:

$$\tilde{\mathcal{L}}(\mathbf{x};\theta,\phi) = \frac{1}{L}\sum_{l=1}^{L}\left[\log p_\theta(\mathbf{x}|\vec{\mathbf{z}}^{(l)}) - \log q_\phi(\vec{\mathbf{z}}^{(l)}|\mathbf{x}) + \log p_\theta(\vec{\mathbf{z}}^{(l)})\right] \tag{A.52}$$

$$+ \sum_{\boldsymbol{c}\in\mathcal{C}}\left\{q_\phi(\mathbf{c}|\mathbf{x})\left[\frac{1}{L}\sum_{l=1}^{L}\log p_\theta(\mathbf{c}\mid\vec{\mathbf{z}}^{(l)}) - \log q_\phi(\mathbf{c}|\mathbf{x})\right]\right\} \tag{A.53}$$

where the optimal value of $q_\phi(\mathbf{c}|\mathbf{x})$ when $L = 1$ is similarly obtained from Theorem 2.2:

$$q_\phi(\mathbf{c}|\mathbf{x}) = \prod_{j=1}^{J} q_\phi(c_j|\mathbf{x}) \tag{A.54}$$

$$q_\phi(c_j|\mathbf{x}) = p_\theta(c_j|\mathbf{z}_j^{(1)}) \quad \text{for } j = 1,\ldots,J \tag{A.55}$$

In this case, the term in (A.53) evaluates to zero, because from Theorem 2

$$\log p_\theta(\mathbf{c}\mid\vec{\mathbf{z}}^{(1)}) = \sum_{j=1}^{J}\log p_\theta(c_j\mid\mathbf{z}_j^{(1)}) = \sum_{j=1}^{J}\log q_\phi(c_j|\mathbf{x}) = \log q_\phi(\mathbf{c}|\mathbf{x}). \tag{A.56}$$

Consequently, we obtain the loss for $L = 1$:

$$\tilde{\mathcal{L}}(\mathbf{x};\theta,\phi) = \log p_\theta(\mathbf{x}|\vec{\mathbf{z}}^{(1)}) - \log q_\phi(\vec{\mathbf{z}}^{(1)}|\mathbf{x}) + \log p_\theta(\vec{\mathbf{z}}^{(1)}) \tag{A.57}$$

where

$$q_\phi(\vec{\mathbf{z}}^{(1)}|\mathbf{x}) = \prod_{j=1}^{J} q_\phi(\mathbf{z}_j^{(1)}|\mathbf{x}) \tag{A.58}$$

$$p_\theta(\vec{\mathbf{z}}^{(1)}) = \prod_{j=1}^{J} p_\theta(\mathbf{z}_j^{(1)}) = \prod_{j=1}^{J}\sum_{c_j=1}^{K_j} p_\theta(\mathbf{z}_j^{(1)}|c_j)p_\theta(c_j) \tag{A.59}$$

A.3.3 Empirical comparison of primary and alternate form

Here, we empirically compare the primary and alternate form with five and three terms, respectively. Each loss comes from a different factorization of $p_\theta(\vec{\mathbf{z}},\mathbf{c})$ as we show above, but are equivalent.

We verified in our implementation that both losses yield the exact same loss values on the same mini-batch, but gradients computed during optimisation are different

as both losses have non-overlapping terms and consequently convergence behavior may differ during training. We are interested in whether these differences are substantial. In particular, [108] used a 5-term loss function (similar to the primary loss in Eq. (A.48), even though a 3-term loss function (similar to the alternate loss) could also be obtained and is arguably more compact.

We investigate this question with the following experimental setup, which is close to the one in [108]: On MNIST, we conduct 10 training runs of our model with varying random seeds for both the primary and alternate form of the loss. We use the following hyperparameters with a shared architecture and refer to Appendix A.4 for a more detailed understanding on these configurations:

- Number of facets: $J = 1$
- Batch size: 512
- Learning rate: 0.002
- Dimension of $\mathbf{z}$: 10
- Number of c (number of clusters): 50
- Covariance structure of $p_\theta(\mathbf{z}|c)$: diagonal
- Output dimensions for layers in $g(\mathbf{x}; \phi)$: $[500, 500, 2000]$
- Output dimensions for layers in $f(\mathbf{z}; \theta)$: $[2000, 500, 500]$

In Fig. A.1, we show unsupervised clustering accuracy on the test set over training epochs for the 10 runs and the primary (left) and alternate (right) form of the loss. Our results indicate that there is no significant difference in performance between the two loss forms. We decide to use the primary form in all our experiments going forward as it is simpler for our implementation, e.g. when combined with progressive training, and is also more intuitively following the generative process of our model.

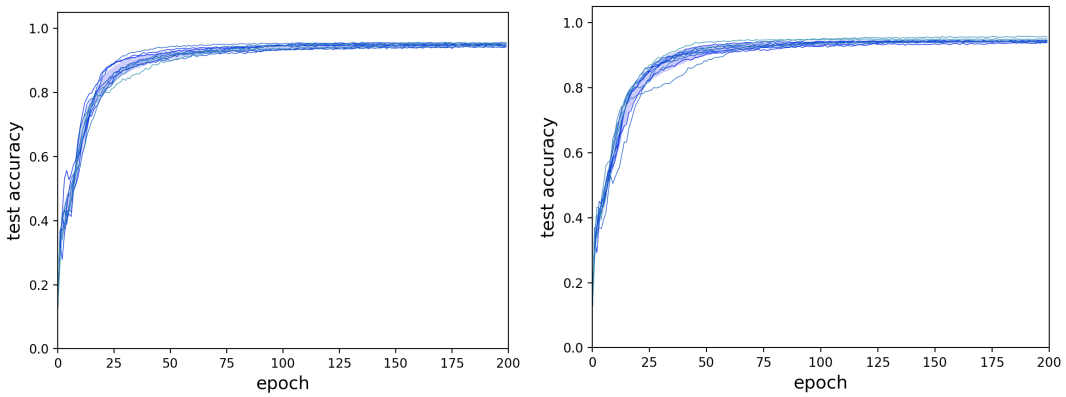


Figure A.1: Unsupervised clustering accuracy on the test set for 10 runs, comparing the primary (left) and alternate (right) loss form. Each run is illustrated by one curve. The blue shade is bounded by the mean accuracy plus and minus one standard deviation across the ten runs.

A.4 Experimental details

We provide our code implementing MFCVAE, using *PyTorch Distributions* [180], together with detailed instructions setup, training and evaluating our model, as well as reproducing the main results of this paper via shell scripts at `https://github.com/FabianFalck/mfcvae`.

A.4.1 Datasets and preprocessing

Throughout our experiments, we use three datasets: MNIST [133], 3DShapes [29], and SVHN [171]. In the following, we briefly introduce these datasets, particularly focusing on their abstract characteristics which might be separated out by a multi-facet clustering model, as well as ethical considerations with regards to their collection. For MNIST and SVHN, we use their implementations as PyTorch Dataset classes as part of the torchvision package to process [180]. For 3DShapes, we provide a custom PyTorch Dataset class which contains several preprocessing steps (detailed below) and the selection of arbitrary combinations of factors.

MNIST. The MNIST database [133] consists of grey-scale (almost binary)

handwritten digits from 10 classes ('0' to '9'). There are 60,000 training examples and 10,000 test examples. The handwritten digits are written by 500 writers (250 writers for training and test set, respectively), introducing a large variation in terms of style of these characters. The most prominent characteristics of MNIST are 1) the digit class, given as a supervised label 2) stroke width (e.g. 'bold', 'thin', ...) 3) the slant of the digits (e.g. 'right-tilted', 'left-tilted', 'upright', ...). During preprocessing, we transform the images to a 0 to 1 scale using min-max scaling.

To the best of our knowledge, the dataset is highly curated and cropped to individual digits, so that we can exclude offensive content or important personally identifiable information in these images. However, we note that as the images are handwritten, there is a possibility that they can be linked to these individuals.

MNIST "was constructed from NIST's Special Database 3 [SD-3] and Special Database 1 [SD-1]" [133]. To the best of our knowledge, SD-3 and SD-1 are no longer available for download (see `https://www.nist.gov/srd/shop/special-database-catalog`), as opposed to other Special Databases. We thus cannot comment on whether and if so in what form consent was obtained from subjects providing the handwritten digits.

3DShapes. The 3DShapes dataset [29] consists of images of three-dimensional shapes in front of a background, generated from six independent ground truth latent factors. These latent factors are floor colour (10 values), wall colour (10 values), object colour (10 values), scale (8 values), shape (4 values), and orientation (15 values). Since all ground truth latent factors are discrete, the nature of this dataset makes it particularly suited for a multi-facet clustering task.

The dataset is preprocessed as follows: We transform each factor's values to a scale of integers between 0 and the number of values of that factor minus one. Then, to be consistent with the SVHN dataset, we resize the original 64×64 images to the size 32×32 using bilinear interpolation. Lastly, we transform the images to a 0 to 1 scale using min-max scaling.

From this 3DShapes dataset, we extract the following 2 configurations which are used during our experiments (note that other configurations can be easily created using our provided Dataset class):

- *Configuration 1* (4,800 images): 10 values for floor colour, 1 value for wall colour, 1 value for object colour, 8 values for scale, 4 values for shape, 15 values for orientation

- *Configuration 2* (4,800 images): 1 value for floor colour, 10 values for wall colour, 1 value for object colour, 8 values for scale, 4 values for shape, 15 values for orientation

3DShapes is a simulated dataset. The dataset was generated using the QUery Networks Mujocu environment [63].

SVHN. SVHN [171] is a real-world image dataset of cropped digits obtained of house numbers in Google Street View images. We focus on the 73,257 training digits and the 26,032 test digits available in the `torchvision` package in PyTorch. SVHN is similar to MNIST in the sense that it is a labelled digit dataset, however, was collected with the aim of being significantly more complex and diverse: As the images were extracted from random Google Street View images in various countries, for example they have varying backgrounds, different number of digits per image (the central digit is used as the label), varying resolutions, and different digit styles, rendering them a challenging dataset for supervised and unsupervised learning tasks, and an interesting test bed for multi-facet clustering, as for some of these characteristics, it might be possible to separate them out. During preprocessing, we transform the images to a 0 to 1 scale using min-max scaling.

A.4.2 Neural architectures and Variational Ladder Autoencoder

In the following, we define two architectures which we implemented in our experiments:

- A Variational Ladder Autoencoder (VLAE) architecture, as defined in [259] and illustrated in Fig. A.2.
- A shared encoder and decoder architecture (we refer to it as "shared architecture" in the following), illustrated in Fig. A.3.

VLAE architecture. The VLAE architecture consists of an encoder (recognition model) and a decoder (generative model) which are symmetric to each other. Both encoder and decoder have a set of backbone layers b_j^{enc} and b_j^{dec}, respectively, which share parameters across the layers of latent variables, and thus naturally build a hierarchy of abstractions. From and into these backbones, a set of rung layers r_j^{enc} and r_j^{dec} emerge which parameterise the latent variables $\vec{\mathbf{z}}$ (encoder) and process their samples towards reconstructions (decoder).

Formally, following the notation in [259], we define the recognition model as

$$\hat{\mathbf{h}}_j = \mathbf{b}_j^{\text{enc}}(\hat{\mathbf{h}}_{j-1}) \tag{A.60}$$

$$\mathbf{z}_j \sim \mathcal{N}\left(\mathbf{z}_j; r_{j,\mu}^{\text{enc}}(\hat{\mathbf{h}}_j), r_{j,\sigma^2}^{\text{enc}}(\hat{\mathbf{h}}_j)\right) \tag{A.61}$$

where $j = 1, \ldots, J$; b_j^{enc} and r_j^{enc} are neural networks, $r_{j,\mu}^{\text{enc}}$ and $r_{j,\sigma^2}^{\text{enc}}$ refer to the elements of the output vector of r_j corresponding to the mean and variance of the parameterised Gaussian distribution $q(\mathbf{z}_j \mid \mathbf{x})$ with diagonal covariance matrix, and $h_0 \equiv \mathbf{x}$.

For each j, $q(c_j \mid \mathbf{x})$ is not directly parameterised by neural networks, but instead computed by Theorem 2.2 as described in Section 2.3.1.

We define the generative model as

$$c_j \sim p(c_j), \text{ for } j = 1, \ldots, J \tag{A.62}$$

$$\mathbf{z}_j \sim p(\mathbf{z}_j|c_j), \text{ for } j = 1, \ldots, J \tag{A.63}$$

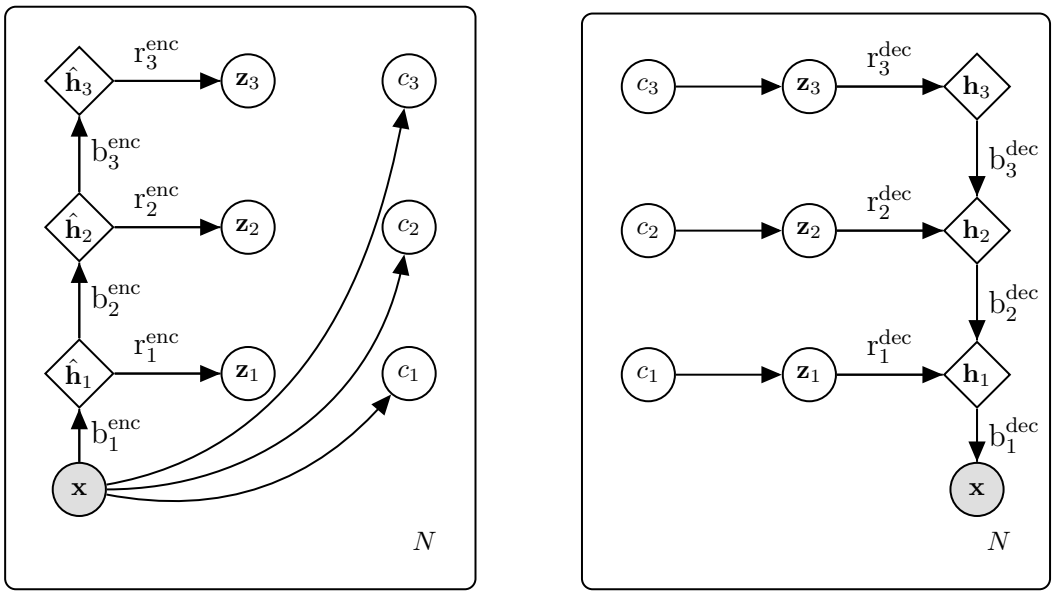
$$\tilde{\mathbf{z}}_J = b_J^{\text{dec}} \circ r_J^{\text{dec}}(\mathbf{z}_J) \tag{A.64}$$

$$\tilde{\mathbf{z}}_j = b_j^{\text{dec}} \left(\left[\tilde{\mathbf{z}}_{j+1}, r_j^{\text{dec}}(\mathbf{z}_j) \right] \right), \text{ for } j = 1, \ldots, J-1 \tag{A.65}$$

$$\mathbf{x} \sim u(\mathbf{x}; \tilde{\mathbf{z}}_1) \tag{A.66}$$

where b_j^{enc} and r_j^{enc} are neural networks, $[\cdot, \cdot]$ denotes concatenation of two vectors, and $u(\mathbf{x})$ is the likelihood model of $\mathbf{x}$.

We refer to Appendix A.4.4 for the exact implementation of all neural networks b_j^{enc}, r_j^{enc}, b_j^{dec}, and r_j^{dec} and the likelihood model $u(\cdot)$ for each of the three datasets.



(a) Recognition model **(b)** Generative Model

Figure A.2: Ladder-MFCVAE architecture with $J = 3$ as an example. (a) The recognition model and (b) generative model. Each *labelled* arrow corresponds to a neural network. The posterior for each c_j is defined using the multi-facet VaDE trick.

Shared architecture. We use the shared architecture as a simple comparison to test our hypothesis that a VLAE helps stabilise training. In the shared architecture, each facet has an equal depth of neural networks and shares the parameters. The encoder and decoder are both fully shared, except for the last hidden layers in both.

More precisely, the recognition model is defined as

$$\hat{\mathbf{h}} = s^{\text{enc}}(\mathbf{x}) \tag{A.67}$$

$$\mathbf{z}_j \sim \mathcal{N}\left(\mathbf{z}_j; t^{\text{enc}}_{j,\mu}(\hat{\mathbf{h}}), t^{\text{enc}}_{j,\sigma^2}(\hat{\mathbf{h}})\right) \tag{A.68}$$

where s^{enc} and each t^{enc}_j are neural networks, and $t^{\text{dec}}_{j,\mu}$ and $t^{\text{dec}}_{j,\sigma^2}$ again refer to those elements of the output vector of t_j corresponding to the mean and variance of the parameterised Gaussian distribution $q(\mathbf{z}_j \mid \mathbf{x})$ with diagonal covariance matrix. $q(c_j \mid \mathbf{x})$ is computed by Theorem 2.2 as described in Section 2.3.1.

The generative model is defined as

$$c_j \sim p(c_j), \text{ for } j = 1, \ldots, J \tag{A.69}$$

$$\mathbf{z}_j \sim p(\mathbf{z}_j|c_j), \text{ for } j = 1, \ldots, J \tag{A.70}$$

$$\mathbf{h} = t^{\text{dec}}([\mathbf{z}_1, \ldots, \mathbf{z}_J]) \tag{A.71}$$

$$\tilde{\mathbf{x}} = s^{\text{dec}}(\mathbf{h}) \tag{A.72}$$

$$\mathbf{x} \sim u(\tilde{\mathbf{x}}) \tag{A.73}$$

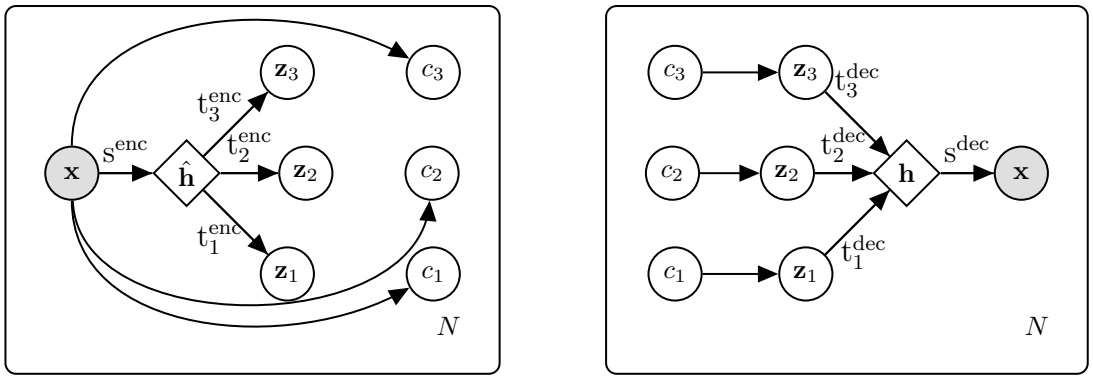
where t^{dec} and s^{dec} are neural networks, $[\cdot, \ldots, \cdot]$ refers to vector concatenation, and $u(\cdot)$ is the likelihood model of $\mathbf{x}$.

Again, we refer to Appendix A.4.4 for the exact implementation of all neural networks s^{enc}, t^{enc}_j, s^{dec}, and t^{dec}, as well as for the likelihood model $u(\cdot)$ for each of the three datasets.

A.4.3 Progressive training algorithm

We use a progressive training algorithm to train our VLAE architectures. We strongly base its implementation on [139] and refer to this source for a more complete introduction, but will point out differences to this formulation below.

The idea of progressive training is to start with training a single facet (typically the one of highest depth in the VLAE architecture), and then progressively loop



(a) Recognition model **(b)** Generative Model

Figure A.3: Shared encoder and decoder MFCVAE architecture with $J = 3$ as an example. (a) The recognition model and (b) generative model. Each labelled arrow corresponds to a neural network. The posterior for each c_j is defined using the multi-facet VaDE trick.

in the other facets one after the other in a smooth manner. To formalise this, we define a progressive step $s = 1, 2, \ldots, J-1$, where in step s, facets $J-s+1$ to J (both including) are contributing to the network (and might be currently looped in), and α_j, the fade-in coefficient of layer j. α_j linearly increases from 0.0 to 1.0 during the first 15,000 (for MNIST and SVHN) or 2,000 (for 3DShapes) batches of a progressive step (except for $s = 1$), is 0.0 if the facet has not yet been looped in, and is 1.0, otherwise. [139] used 5,000, but we increased this number for MNIST and SVHN to have a smoother loop-in of facets.

In contrast to the formulation in [139] which excludes from the model latent facets that are not looped in yet in a certain progressive step, in our formulation, all latent facets are part of the model throughout all progressive steps, yet do not contribute to the KL-divergences or the reconstruction term in Eq. (2.14). We achieve this by applying the fade-in coefficient only to the decoder rungs, not the encoder rungs (compare Eq. (9) in [139], where both the encoder and decoder rungs are faded in); and to weigh the KL-divergences in $\mathbf{z}_j$ and c_j, as before. In other words, this is similar to the implementation of progressive training in [139] with $\alpha_j = 1.0$ for the encoder rungs throughout all progressive steps, and the regular, smoothly increasing α_j value for the decoder rungs. Precisely, to implement the

progressive training algorithm, we amend Eq. (A.65) as follows:

$$\tilde{\mathbf{z}}_j = b_j^{\text{dec}}\left(\left[\tilde{\mathbf{z}}_{j+1}, \alpha_j r_j^{\text{dec}}(\mathbf{z}_j)\right]\right) \quad \text{, for } \; j = 1, \ldots, J, \tag{A.74}$$

$$\begin{aligned}\mathcal{L}^{\text{MFCVAE}}(\mathcal{D}; \theta, \phi) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\Bigg[& \mathbb{E}_{q_\phi(\vec{\mathbf{z}}|\mathbf{x})} \log p_\theta(\mathbf{x}|\vec{\mathbf{z}}) \\ & - \sum_{j=1}^{J} \alpha_j \left[\mathbb{E}_{q_\phi(c_j|\mathbf{x})} \operatorname{KL}(q_\phi(\mathbf{z}_j|x)||p_\theta(\mathbf{z}_j|c_j)) + \operatorname{KL}(q_\phi(c_j|\mathbf{x})||p(c_j))\right]\Bigg]\end{aligned} \tag{A.75}$$

where

$$\alpha_j = 1.0, \quad \text{for } \; j = (J - s + 1), \ldots, J \tag{A.76}$$

$$\alpha_{J-s} \in [0, 1] \;\; \text{(looped in)} \tag{A.77}$$

$$\alpha_j = 0.0, \quad \text{for all } \; j = 1, \ldots, (J - s - 1) \tag{A.78}$$

and all other equations of the VLAE remain unchanged. Thus, when $\alpha_j = 0.0$, the gradient w.r.t. any parameters in $b_j^{\text{enc}}, r_j^{\text{enc}}, b_j^{\text{dec}}, r_j^{\text{dec}}$, as well as the parameters of the priors $p(c_j)$ and $p(\mathbf{z}_j)$ are 0, and we achieve the same effect as if those components would not be part of the model.

Lastly, while we have tested "pretraining" the latent facets which are not looped in yet through a KL-regularisation terms in $\mathbf{z}_j$ and c_j (see Eq. (10) in [139]), we could not see a beneficial effect on stability of model training in our model. As this would add complexity to the training algorithm, we do not pursue this type of pretraining here.

A.4.4 Implementation details and hyperparameters

New assets. We publish the following new assets accompanying this paper:

- *Code*: We provide our source code with detailed instructions on setup, reproducing our results via shell scripts, training, evaluation in a README file at `https://github.com/FabianFalck/mfcvae`. We follow the code templates and NeurIPS guidelines for code submissions. Our code is provided under MIT

license. The initial implementation of our model are inspired by the codebase of VaDE in `https://github.com/eelxpeng/UnsupervisedDeepLearning-Pytorch/blob/master/udlp/clustering/vade.py`, as well as the official VLAE implementation ([259] and `https://github.com/ermongroup/Variational-Ladder-Autoencoder`), and the official ProVLAE implementation ([139] and `https://github.com/Zhiyuan1991/proVLAE/blob/master/model_ladder_progress.py`). Our convolutional VLAE architecture is largely based on the neural architecture for the CelebA dataset in the ProVLAE codebase mentioned above.

- *Pretrained models*: We further provide pretrained models with the hyperparameters reported in this section as part of the folder `pretrained_models/`.

Existing assets used. Our work uses the following Python software packages with accompanying licenses (if known): PyTorch [180] (in particular the PyTorch Distributions and Torchvision packages; custom license), Numpy [85] (BSD 3-Clause License), Weights&Biases [19] (MIT License), Matplotlib [99] (PSF License), Seaborn [232] (BSD 3-Clause License), Pickle [225] (N/A), H5Py [39] (BSD 3-Clause License), OpenCV 2 [25] (Apache License), Scikit-learn [181] (BSD 3-Clause License), boilr (`https://github.com/addtt/boiler-pytorch`) (MIT License). Regarding data assets used, we refer to Section A.4.1.

Data splits. For all three datasets, we split data into training and test dataset (no validation dataset used). For MNIST and SVHN, we use the standard data splits as provided with these datasets and in the TorchVision PyTorch package. For 3DShapes, in both configurations, we use 80% for training and the remaining 20% for testing. Here, we sample the images uniformly at random and without replacement. We also refer to Appendix A.4.1 for a more detailed discussion on preprocessing of these datasets.

Likelihood models. For MNIST data, we define its likelihood $p(\mathbf{x}|\vec{\mathbf{z}})$ as a product of independent Bernoulli likelihoods, where each dimension is Bernoulli-distributed with respect to some learnt parameter and independent of other dimensions. Bernoulli likelihood is a reasonable assumption, because most pixels in MNIST images have values close or equal to 0 and 1.

For 3DShapes and SVHN data, we define their likelihood $p(\mathbf{x}|\vec{\mathbf{z}})$ to be a product of independent Gaussian likelihoods, where each dimension is Gaussian-distributed with its mean learnt as a parameter and its variance fixed as a hyperparameter.

Other design choices. Apart from the neural architectures, hyperparameters and likelihood models of MFCVAE, there were other design choices which were made on a per-dataset basis:

- *Covariance structure of the Gaussian $p(\mathbf{z}_j|c_j)$ for each j and c_j*: The covariance matrices can be set to either diagonal or full. In this paper, diagonal covariance is found to be sufficient for MNIST. Full covariance is chosen for 3DShapes and SVHN as it results in a stronger disentanglement of facets.

- *Whether to fix $\boldsymbol{\pi}_j$ or train them as parameters*: In order to encourage clusters to have similar sizes in each facet, one option is to fix $\boldsymbol{\pi}_j$ to be $1/K_j$ componentwise for each facet j. We fix $\boldsymbol{\pi}_j$ in models trained on 3DShapes and SVHN.

- *Activation functions:* To avoid vanishing gradients and encourage a more stable training, we tested three activation functions, in particular, ReLU, leaky ReLU and ELU. For MNIST, we found ReLU to be sufficient. For 3DShapes and SVHN, where convolutional neural networks are involved, we sometimes observed vanishing gradients in training runs, which is why we used the ELU activation function where we no longer observed this problem (leaky ReLU likewise worked, but we chose ELU for consistency with previous VLAE implementations mentioned above).

Hyperparameter tuning. We performed several large exploratory hyperparameter sweeps over wide grids of possible hyperparameter values, looking at the qualitative improvement in facet disentanglement and, where available, the training accuracy of supervised labels (often only one label of the two facets of interest available). In these exploratory hyperparameter sweeps, we observed the following hyperparameter patterns that generalise across all datasets:

We noticed that results are stable w.r.t. to a large set of hyperparameters and ranges of possible values. In particular, this applies to batch size, learning rate, the number of batches used during fade-in, and to some degree the number of clusters in both facets. However, we noticed that some hyperparameters must be set rather carefully to achieve strong disentanglement between facets. In particular, we find that the style/colour facet's latent dimension must be rather precisely set to a narrow range of values yielding strong disentanglement of facets: For MNIST, $\dim(\mathbf{z}_1)$ has to be around 5. For SVHN, $\dim(\mathbf{z}_2)$ has to be around 5. For 3DShapes, $\dim(\mathbf{z}_2)$ has to be around 2.

Hyperparameters of the reported results. In the following, we report the hyperparameters for training our models reported and presented in Section 2.4 (note that the hyperparameters for the models trained on the two different 3DShapes configurations are the same). In Table A.1, we report chosen values of scalar hyperparameters. For full details on each of these hyperparameters, we refer to our code and in particular the help message of the respective command line arguments in the training script.

Neural architectures. Following up Appendix A.4.2, we here provide the detailed initialisation of hidden layers of our neural architectures.

We first discuss the fully-connected ladder architecture which we use to train MFCVAE on MNIST, with results presented in Section 2.4 and Appendix A.5. We initialise the VLAE architecture as detailed in Table A.2.

Table A.1: Scalar hyperparameters and design choices for our three model configurations on MNIST, 3DShapes and SVHN, with results presented in Section 2.4 and Appendix A.5.

	MNIST	3DShapes	SVHN
Batch size	512	150	150
Learning rate	0.0005	0.0003	0.0005
Latent dimension of the first facet, $\dim(\mathbf{z}_1)$	5	20	22
Number of clusters in the first facet, $\dim(c_1)$	25	60	200
Latent dimension of the second facet, $\dim(\mathbf{z}_2)$	5	2	7
Number of clusters in the second facet, $\dim(c_2)$	25	20	50
Number of training batches for each fade-in	15000	2000	15000
Likelihood model for $p(\mathbf{x}\|\vec{\mathbf{z}})$	Bernoulli	Gaussian	Gaussian
Standard deviation of $p(\mathbf{x}\|\vec{\mathbf{z}})$ componentwise (if Gaussian)	N/A	0.6	0.3
Data dependent initialisation for $g(\mathbf{x};\phi)$ and $f(\vec{\mathbf{z}};\theta)$	No	Yes	Yes
Covariance structure of $p(\mathbf{z}_j\|c_j)$	diagonal	full	full
Fix $\boldsymbol{\pi}_j$	No	Yes	Yes
Diagonal entries during initialisation for covariance of $p(\mathbf{z}_j\|c_j)$	0.01	0.01	0.01
Activation function	ReLU	ELU	ELU

Table A.2: Details of the fully-connected ladder architecture for our model trained on MNIST and reported in Section 2.4 and Appendix A.5.

Recognition Network	Generative Network
b_1^{enc}: $\dim(\mathbf{x}) \times 500$ linear layer	r_2^{enc}: $\dim(\mathbf{z}_2) \times 2000$ linear layer
ReLU activation	ReLU activation
r_1^{enc}: $500 \times (2 \cdot \dim(\mathbf{z}_1))$ linear layer	b_2^{enc}: 2000×500 linear layer
	ReLU activation
b_2^{enc}: 500×2000 linear layer	r_1^{enc}: $\dim(\mathbf{z}_1) \times 500$ linear layer
ReLU activation	ReLU activation
r_2^{enc}: $2000 \times (2 \cdot \dim(\mathbf{z}_2))$ linear layer	b_1^{enc}: $500 \times \dim(\mathbf{x})$ linear layer
	Sigmoid activation

Next, we describe the convolutional ladder architecture which we use to train MFCVAE on 3DShapes and SVHN, with results presented in Section 2.4 and Appendix A.5. We initialise the VLAE architecture as detailed in Table A.3.

Lastly, in Table A.4, we provide details on the shared architecture for MNIST training, with its results presented in Appendix A.5.1.

Initialisation. In MFCVAE, all parameters in the deep neural networks $g(\mathbf{x};\phi)$ and $f(\vec{\mathbf{z}};\theta)$ are initialised using either Glorot normal initialisation [73] for MNIST,

Table A.3: Details of the convolutional ladder architecture trained on 3DShapes and SVHN. Conv2d is the 2D convolutional operation, and ConvTranspose2d is the 2D transposed convolutional operation. We implement both operations using the `torch.nn` package in PyTorch. For both operations, the four numbers represent output channels, input channels, kernel size (height) and kernel size (width) respectively (C_{out}, C_{in}, H, W). Convolutional operations marked with (*) have stride 1 to ensure valid dimensions. All remaining convolutional operations have stride 2. For experimental results of models from this architecture, see Section 2.4 and Appendix A.5.

Recognition Network	Generative Network
b_1^{enc}: $64 \times \dim(\mathbf{x}) \times 4 \times 4$ Conv2d	r_2^{enc}: $\dim(\mathbf{z}_2) \times 16384$ linear layer
ELU activation, batch norm	ELU activation, batch norm
r_1^{enc}: $64 \times 64 \times 4 \times 4$ Conv2d	b_2^{enc}: $128 \times 256 \times 4 \times 4$ ConvTranspose2d
ELU activation, batch norm	ELU activation, batch norm
$64 \times 64 \times 4 \times 4$ Conv2d (*)	$64 \times 128 \times 4 \times 4$ ConvTranspose2d (*)
ELU activation, batch norm	ELU activation, batch norm
$1024 \times (2 \cdot \dim(\mathbf{z}_1))$ linear layer	r_1^{enc}: $\dim(\mathbf{z}_1) \times 16384$ linear layer
b_2^{enc}: $128 \times 64 \times 4 \times 4$ Conv2d	ELU activation, batch norm
ELU activation, batch norm	b_1^{enc}: $\dim(\mathbf{x}) \times 128 \times 4 \times 4$ ConvTranspose2d
r_2^{enc}: $128 \times 128 \times 4 \times 4$ Conv2d	ELU activation, batch norm
ELU activation, batch norm	Sigmoid activation (only for SVHN)
$256 \times 128 \times 4 \times 4$ Conv2d	
ELU activation, batch norm	
$3136 \times (2 \cdot \dim(\mathbf{z}_2))$ linear layer	

Table A.4: Details of the shared architecture trained on MNIST. For its results, see Appendix A.5.1.

Recognition Network	Generative Network
s^{enc}: $\dim(\mathbf{x}) \times 500$ linear layer	t_1^{dec}: $\dim(\mathbf{z}_1) \times 2000$ linear layer
ReLU activation	ReLU activation
500×2000 linear layer	t_2^{dec}: $\dim(\mathbf{z}_2) \times 2000$ linear layer
ReLU activation	ReLU activation
t_1^{enc}: $2000 \times (2 \cdot \dim(\mathbf{z}_1))$ linear layer	s^{dec}: 2000×500 linear layer
	ReLU activation
t_2^{enc}: $2000 \times (2 \cdot \dim(\mathbf{z}_2))$ linear layer	$500 \times \dim(\mathbf{x})$ linear layer
	Sigmoid activation

and using a data-dependent initialisation method [126] for 3DShapes and SVHN. The idea of the data-dependent initialisation is to set the parameters in the deep neural network such that all layers in the network are encouraged to train at roughly the same rate, with the aim of avoiding vanishing or exploding gradients. Data-

dependent initialisation is particularly useful for convolutional neural networks. Therefore, we use it as a starting point for model training of 3DShapes and SVHN datasets, where convolutional neural networks are used.

For the parameters of the MoGs, we initialise them facet-wise as follows:

- Mixing weights $\boldsymbol{\pi}_j$ are initialised to be $1/K_j$ component-wise.
- For each $k_j \in \{1, ..., K_j\}$, means $\boldsymbol{\mu}_{j,k_j}$ of the Gaussians are initialised with the means on an MoG (implemented with the package `sklearn.mixture.GaussianMixture`) fitted on a dataset consisting of latent observations $\mathbf{z}_j$ sampled from $q(\mathbf{z}_j|\mathbf{x})$, where $\mathbf{x}$ are all batches from the corresponding training dataset of MFCVAE. Note that encoder parameterising $q(\mathbf{z}_j|\mathbf{x})$ is not trained at this point. The aim is to encourage a smoother and faster learning of the multiple MoG prior by starting from an MoG fitted to the initial state instead of one initialised at random.
- Covariance matrices Σ_{j,k_j} are not initialised by the outputs from the fitted MoG above. Instead, a fixed value is assigned to all diagonal entries of the covariance matrices. The fixed value is the same across all clusters in all facets, which is a hyperparameter set to be much larger than the output variances from the trained fitted MoGs. This initialisation is favoured because at the start of the training, the MoGs do not contain much useful information as they were fitted on latent observations obtained from a randomly initialised model. An overly small variance at the start of the training could result in the model being stuck in a local optimum prematurely.

We note that we found these prior initialisations to be reasonable choices, but have not extensively explored alternatives.

Potential negative societal impacts. Our work is mainly of theoretical and methodological nature, thus, we do not have a direct application of our model on

which it could cause immediate negative societal impacts. Since we provide a general clustering algorithm, MFCVAE can be used in malicious or potentially unethical ways for any clustering task at hand, suited particularly for high-dimensional data. Our model does not account for fairness of clusters, which should be taken particular care of when dealing with data from human subjects. As our model is a generative model by nature, we mention the possibility to abuse our model for the generation of deepfakes for disinformation. Further, we have not investigated the vulnerability of our model to aversarial attacks, which might cause a significant security problem when applied in front-end applications and tasks.

Compute resources. We had access to two GPU clusters: One internal cluster with 12 Nvidia GeForce GTX 1080 graphic cards each with 8GB of memory that was shared with many other users (access for 5 months ongoing), and one Microsoft Azure cluster with initially two, later four Nvidia Tesla M60 each with 8GB of memory that was used only by the authors (access for approximately 4 months).

To train one model on each of the 4 dataset (configurations) on the Azure cluster detailed above, it takes approximately 31 min for MNIST, 36 min for 3DShapes (in both configurations), and 5h 54 min for SVHN. Since we performed a seed sweep over 10 runs on each of the 4 dataset (configurations), the total computational time to reproduce the main results in this paper is (31 min + $2 \cdot$ 36 min + 354 min) $\cdot$ 10 = 4570 min $\approx$ 76 hours of GPU time.

A.4.5 Differences between VaDE and ($J = 1$) MFCVAE

In the following, we describe the (pre-)training algorithm of VaDE [108] and compare it with the training of MFCVAE with one facet ($J = 1$). Throughout, VaDE uses a symmetric, shared encoder and decoder architecture (see Appendix A.4.2). VaDE has the following two differences compared to MFCVAE ($J = 1$):

- *Stacked Denoising Autoencoder (SAE)* [228] pretraining: VaDE uses a two-stage SAE deterministic pretraining algorithm which is in detail described in [242] to find good initialisations for the parameters θ of the decoder and ϕ of the encoder. During the first stage, denoising autoencoders, which are two-layer neural networks of symmetric shapes, are trained using a least-squares reconstruction loss (i.e. deterministically as a plain autoencoder). In every iteration of this first stage of the pretraining routine, the outermost layers (at the front of the encoder and the back of the decoder), which have been trained in previous iterations, are frozen, and the next denoising autoencoder towards the centre of the architecture is trained. Then, in the second stage of training, the entire architecture, which has been trained in this sequential fashion, is fine-tuned, again using a deterministic reconstruction loss. For a detailed description of this pretraining algorithm, we refer to [242].
  Once the pretraining routine is complete, VaDE uses the weights θ and ϕ obtained as the initialisation of regular VaDE training, maximizing the ELBO with Monte Carlo sampling.— We note that MFCVAE does not require any SAE pretraining. Instead, we either initialise these weights randomly (MNIST) or using data-dependent initialization (3DShapes and SVHN; details see Appendix A.4.4).

- VaDE restricts the covariance matrices Σ_c of the conditional Gaussian distributions $p(\mathbf{z} \mid c) = \mathcal{N}(\mu_c, \Sigma_c)$ to be *diagonal*, i.e. each $p(\mathbf{z} \mid c)$ is a product of $\dim(\mathbf{z})$ independent univariate Gaussian distributions. In contrast, MFCVAE allows Σ_c to be *full* and enables MFCVAE to express more complex (facet-wise) dependencies in the prior.

The SAE pretraining routine adds significant complexity to the training algorithm which MFCVAE does not require in order to obtain comparable performance. Once the encoder and decoder are initialised, both VaDE and MFCVAE use a Gaussian-Mixture model to initialise the prior $p(\mathbf{z})$ and its parameters π, μ_c and Σ_c, fitted with an EM-algorithm. We note that in the single-facet case, training MFCVAE

simplifies to only one progressive step, i.e. the main training stage of VaDE is equivalent to that of MFCVAE (but with different initialisation).

A.5 Additional experimental results

A.5.1 On the stability of training

In this appendix, we analyse the stability of MFCVAE with respect to different neural architectures and discuss the stability of deep clustering models in this context.

A natural starting point for a neural architecture of MFCVAE is a shared encoder and decoder architecture (as detailed in Appendix A.4.2), which was previously used in VaDE [108] and other deep clustering models. When using this architecture, we observe a high variation between runs which only vary in their (partly) random initialisation (determined by the random seed; see Fig. A.4 [Top left]). However, when being lucky, drawing the right lottery ticket, this architecture can yield excellent disentanglement of facets, just like our progressively trained VLAE architecture can do (but in a stable manner). We visualise input examples assigned to clusters from such a lucky run (which is not part of Fig. A.4 [Top left]) in Fig. A.5. We point out that this run is cherry-picked from over 100 runs with different random initialisation. We could not produce stable results with a shared encoder and decoder architecture, neither for $J = 1$ nor $J > 1$. Given these stability issues of deep clustering models, it is not only crucial to address them (which we do next), but this even more highlights the importance of providing error bars, and that picking the best run of many (as has been common practice among several deep clustering papers) is particularly here not acceptable.

To overcome these stability issues, we used a combination of a progressive training algorithm and a VLAE architecture. We found that only using a VLAE architecture (Fig. A.4 [Top right]) significantly improves the performance of disentangling facets

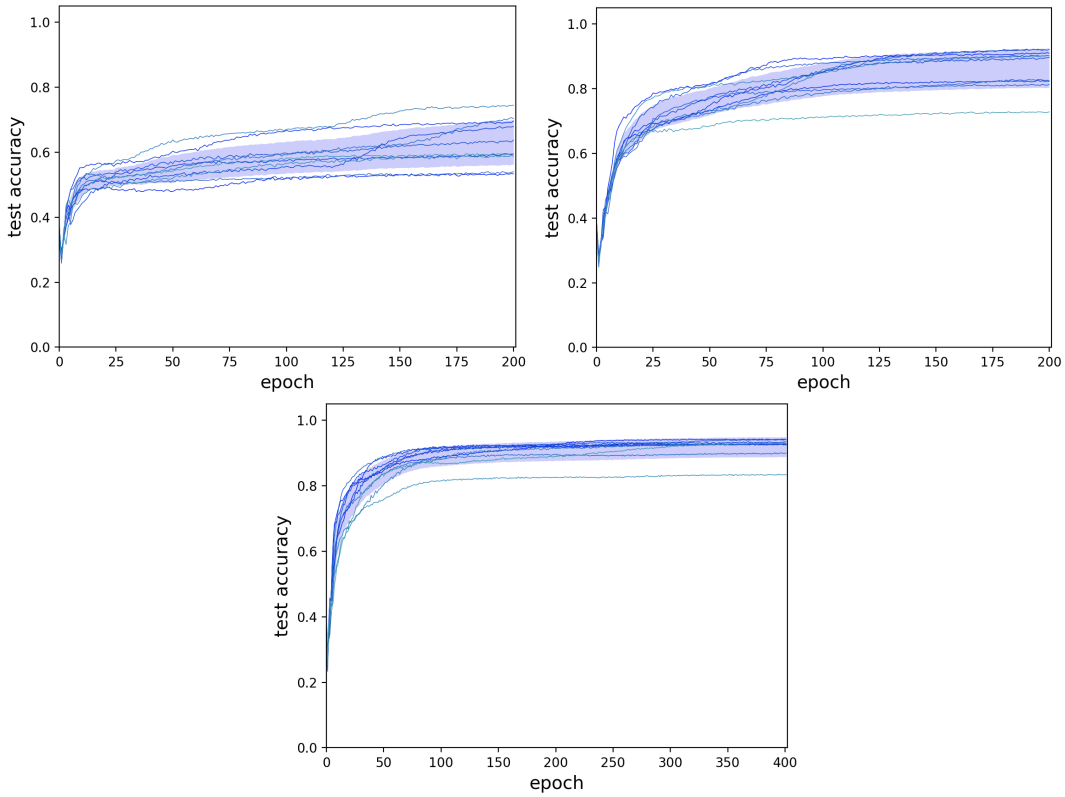


Figure A.4: Test accuracy over training epochs for models trained on MNIST for three different architectures. Ten runs are performed for each architecture. Each run is illustrated by one curve. The blue shade is bounded by the mean accuracy plus and minus one standard deviation across the ten runs. [Top left] Shared architecture [Top right] VLAE architecture *without* progressive training schedule [Bottom] VLAE architecture *with* progressive training schedule

(here only measured in terms of accuracy w.r.t. the supervised label), but is not sufficient to fully stabilise the runs over different random seeds. Only by additionally using a progressive training schedule (Fig. A.4 [Bottom]), we achieve very good disentanglement of facets and at the same time stable performance. While we here report this for one configuration of hyperparameters only and on MNIST, we made this observation throughout all datasets and in diverse hyperparameter settings.

A.5.2 Generalisation between training and test set

An important question in an exploratory setting is to what degree clustering results generalise from a training to a test set. In supervised machine learning, it is common

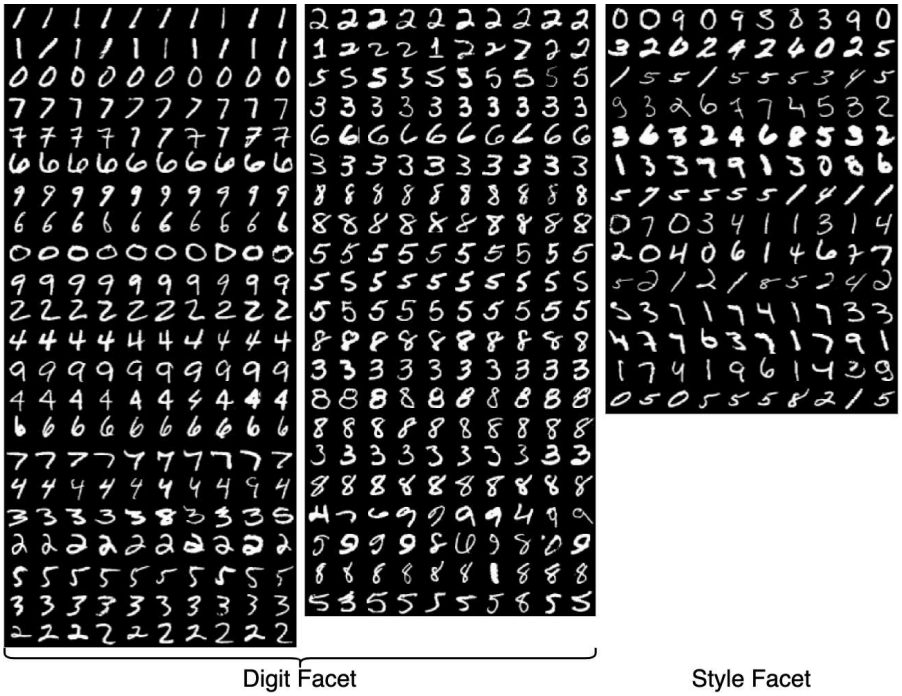


Figure A.5: Input examples of a cherry-picked MFCVAE model with a shared architecture, with a lucky lottery ticket drawn as the random initialisation, and two-facets ($J = 2$), trained on MNIST. Sorting is performed in the same way as in Fig. 2.4.

to see a generalisation gap: Performance of a model is generally better on the training set than on the test set, often because the model overfits on the training set, and the goal is to minimise this gap, while actually being interested in test set performance. Perhaps surprisingly, we observe that MFCVAE has a negligible generalisation gap, i.e. typically performs almost equally well on the training and test set.

To analyse this, we use the exact experimental setup of our main results as detailed in Appendix A.4.4 with $J = 2$ facets, training on MNIST. Fig. A.6 shows the unsupervised clustering accuracy over training epochs, evaluated both on the training set (left) and the test set (right). When evaluating unsupervised clustering accuracy after the model is fully trained, a mean training accuracy of $91.85\% \pm 3.09\%$ is achieved across ten runs, which is slightly lower than the mean test accuracy of $92.02\% \pm 3.02\%$ presented in Section 2.4.3. Thus, while we observe small differences between performance on training and test set, also when considering individual runs, these are not significant. In summary, MFCVAE generalises well

between training and test set.

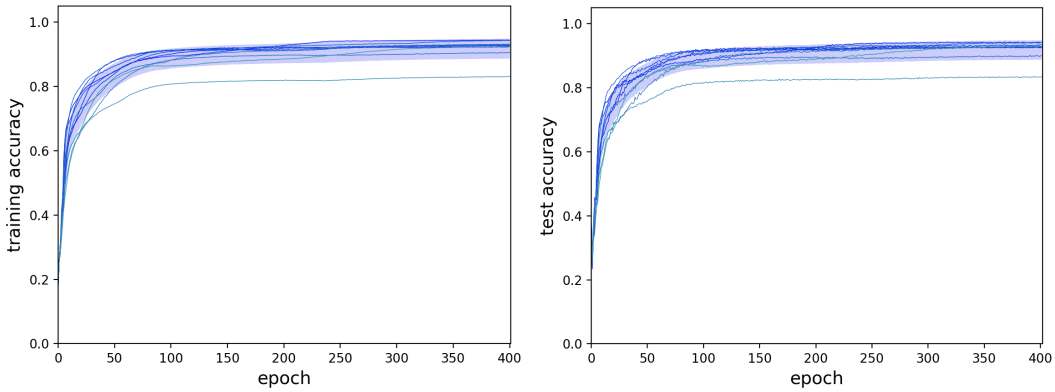


Figure A.6: Unsupervised clustering accuracy over training epochs for MFCVAE trained on (the training set of) MNIST as detailed in Appendix A.4.4, and evaluated on the training set [Left] and the test set [Right], respectively. Ten runs are performed, with each run being illustrated by one curve on the left and right, respectively. The blue shade is bounded by the mean accuracy plus and minus one standard deviation across the ten runs.

A.5.3 Discovering a multi-facet structure

This appendix provides the complete results of Section 2.4.1. As before, we visualise input examples from the test set for clusters of MFCVAE with two-facets ($J = 2$) trained on MNIST, 3DShapes and SVHN. Here, we show all clusters of our results in Fig. 2.4, visualised in Figs. A.7 to A.10. In all figures, inputs (columns) are sorted in decreasing order by their assignment probability $\max_{c_j} \boldsymbol{\pi}_j(c_j|q_\phi(\mathbf{z}_j|\mathbf{x}))$.

In particular, in Fig. A.7, we directly compare our model to the results shown in LTVAE [138], the model closest to ours in its attempt to learn a clustered latent space with multiple disentangled facets. As can be seen, LTVAE struggles to separate data characteristics into separate facets (see Fig. A.7 (b)). In particular, LTVAE learns digit class in both facets, i.e. this characteristic is not properly disentangled between facets. In comparison, MFCVAE better isolates the two characteristics, and does not learn digit class in the style facet.

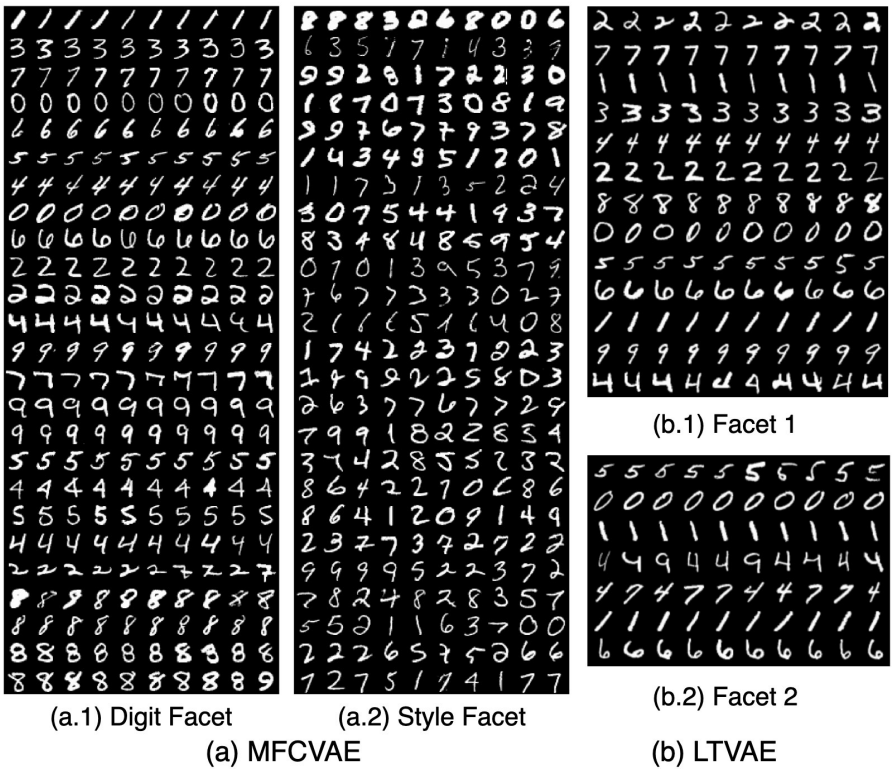


Figure A.7: (a) Input examples for clusters of MFCVAE with two-facets ($J = 2$) trained on MNIST. Rows and columns are sorted as in Fig. 2.4. (b) Input examples for clusters of LTVAE with two-facets, likewise trained on MNIST. Plot is taken as reported in [138], Fig. 5.

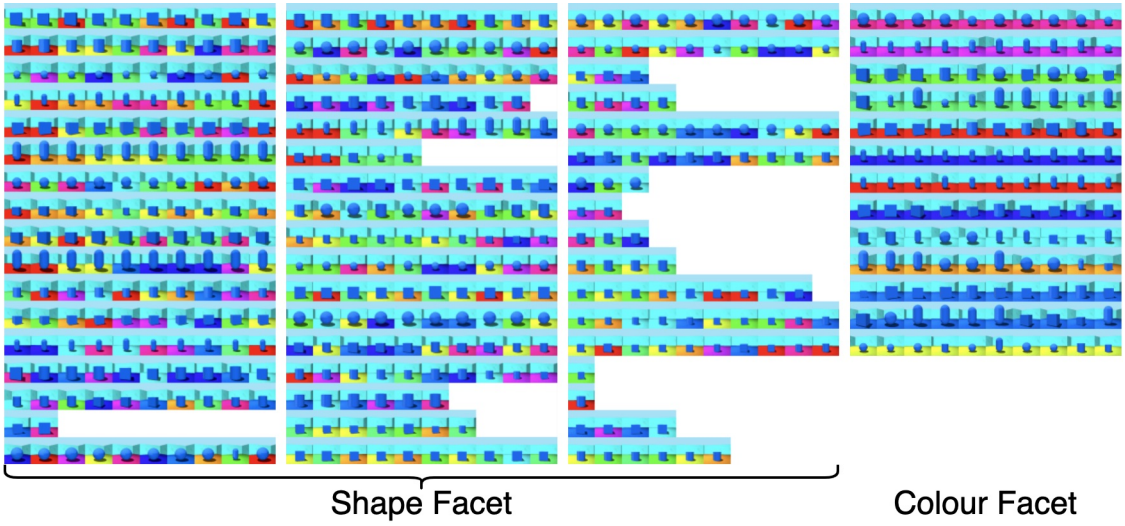


Figure A.8: Input examples for clusters of MFCVAE with two-facets ($J = 2$) trained on 3DShapes (configuration 1). Rows and columns are sorted as in Fig. 2.4.

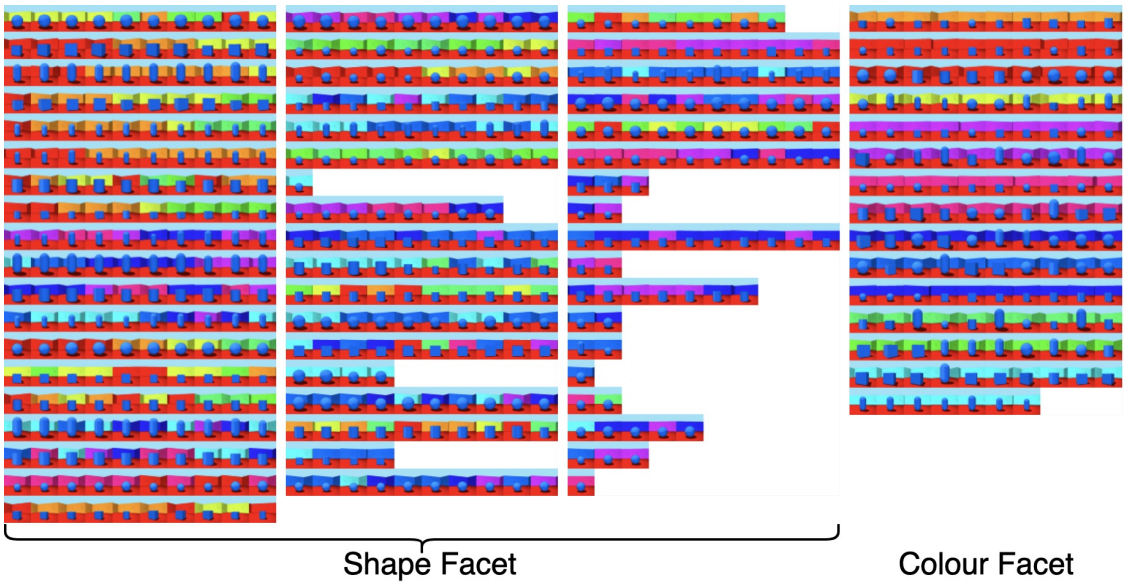


Figure A.9: Input examples for clusters of MFCVAE with two-facets ($J = 2$) trained on 3DShapes (configuration 2). Rows and columns are sorted as in Fig. 2.4.

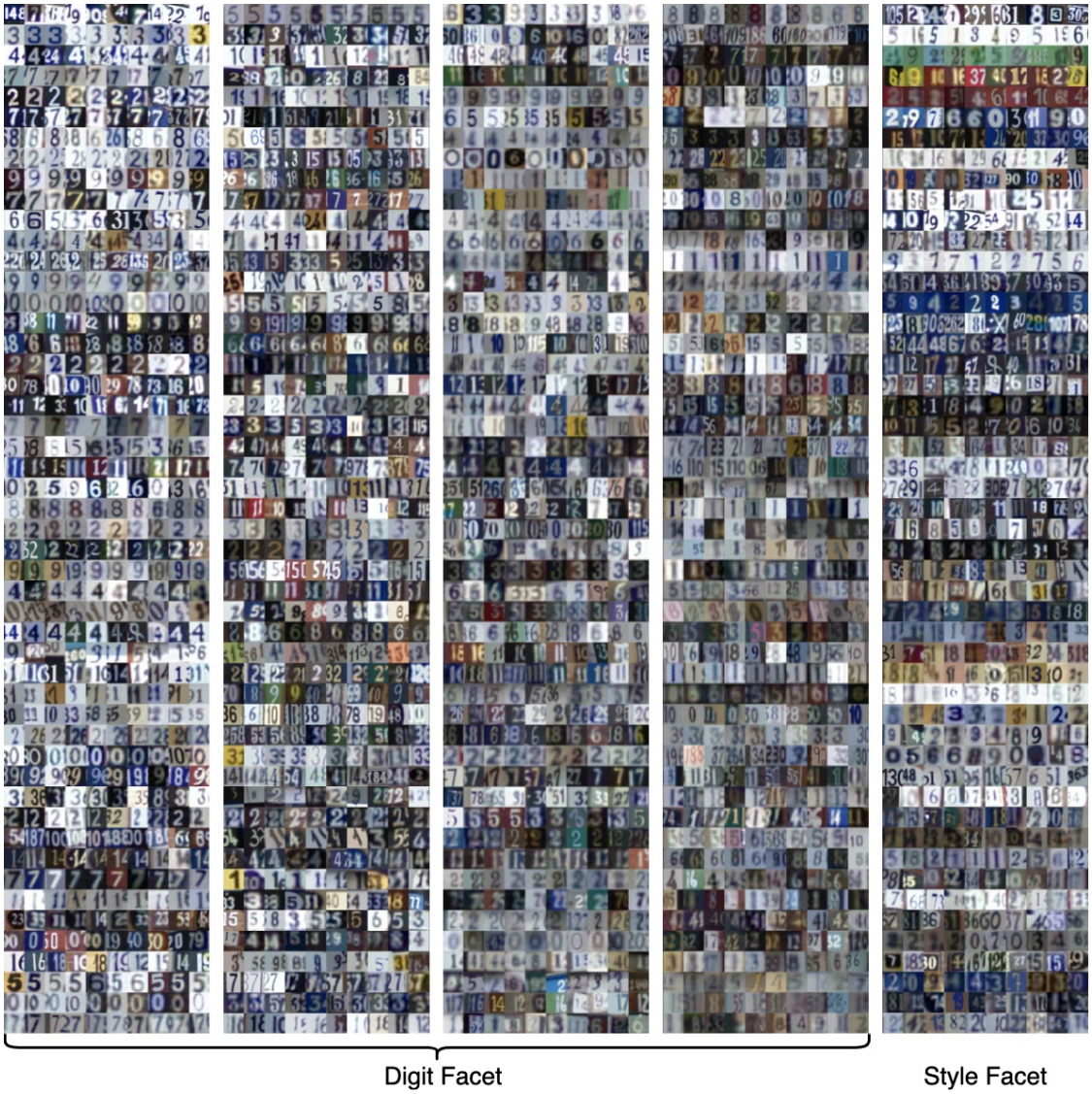


Figure A.10: Input examples for clusters of MFCVAE with two-facets ($J = 2$) trained on SVHN. Rows and columns are sorted as in Fig. 2.4.

A.5.4 Compositionality of facets

In this appendix, we provide further combinations of clusters of which the style facet is swapped and give a more rigorous explanation of the swapping procedure applied.

Let us have two input examples $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ assigned to two different style clusters according to Eq. (2.12) (and typically two different digit clusters), i.e. $c_j^{(i)} = \text{argmax}_{c_j} \boldsymbol{\pi}_j(c_j|q_\phi(\mathbf{z}_j|\mathbf{x}^{(i)}))$ and $c_j^{(1)} \neq c_j^{(2)}$, where $j = 1$ for MNIST and $j = 2$ for 3DShapes and SVHN. We can obtain the latent representations $\tilde{\mathbf{z}}_j^{(i)}$ of input examples for both facets, taking the mode of $q_\phi(\mathbf{z}_j|\mathbf{x}^{(i)})$, respectively, which is parameterised via a forward pass. Now, we swap the style/colour facet's latent representation ($\tilde{\mathbf{z}}_1^{(i)}$ for MNIST, and $\tilde{\mathbf{z}}_2^{(i)}$ for 3DShapes and SVHN) between the two inputs, while fixing the digit/shape facet's latent representation ($\tilde{\mathbf{z}}_2^{(i)}$ for MNIST, and $\tilde{\mathbf{z}}_1^{(i)}$ for 3DShapes and SVHN). Once the swapping is complete, we pass these latent representations through the decoder of our model to obtain reconstructions of our model from "swapped style" latent representations. Note that this swapping operation is symmetric in the two-facet case, in the sense that the resulting two reconstructions are the same regardless of whether we swap $\tilde{\mathbf{z}}_1$ or $\tilde{\mathbf{z}}_2$. Formally, we obtain reconstructions $\hat{\mathbf{x}}^{(1)} = f(\{\tilde{\mathbf{z}}_1^{(1)}, \tilde{\mathbf{z}}_2^{(2)}\}; \theta)$ and $\hat{\mathbf{x}}^{(2)} = f(\{\tilde{\mathbf{z}}_1^{(2)}, \tilde{\mathbf{z}}_2^{(1)}\}; \theta)$. It is such 4-tuples of inputs $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ and reconstructions $\hat{\mathbf{x}}^{(1)}$ and $\hat{\mathbf{x}}^{(2)}$ that we visualise.

In Fig. A.11, we visualise further 4-tuples from all datasets (and configurations), in addition to the results presented in Section 2.4.2. For each dataset, the first 4-tuple consists of the first element of both input rows (left and right) and both reconstruction rows (left and right). Note that we limit ourselves here to few digit/shape and colour/style cluster combinations. However, we note that for MNIST and 3DShapes, many other combinations could be found where a similar "style swapping effect" can be observed. For the much richer dataset SVHN on which our model is less well fitted, while we can find cluster combinations where compositionality of facets works somewhat well (see Fig. A.11 (d)), we can also find failure cases (see e.g. Fig. A.11 (e)). Here, the style, represented as a white

background, shall be composed with a white digit. We hypothesise that as this combination is particularly rare in the real world, the model struggles to reconstruct these latent combinations and as a consequence introduces undesired artefacts into the reconstructions.

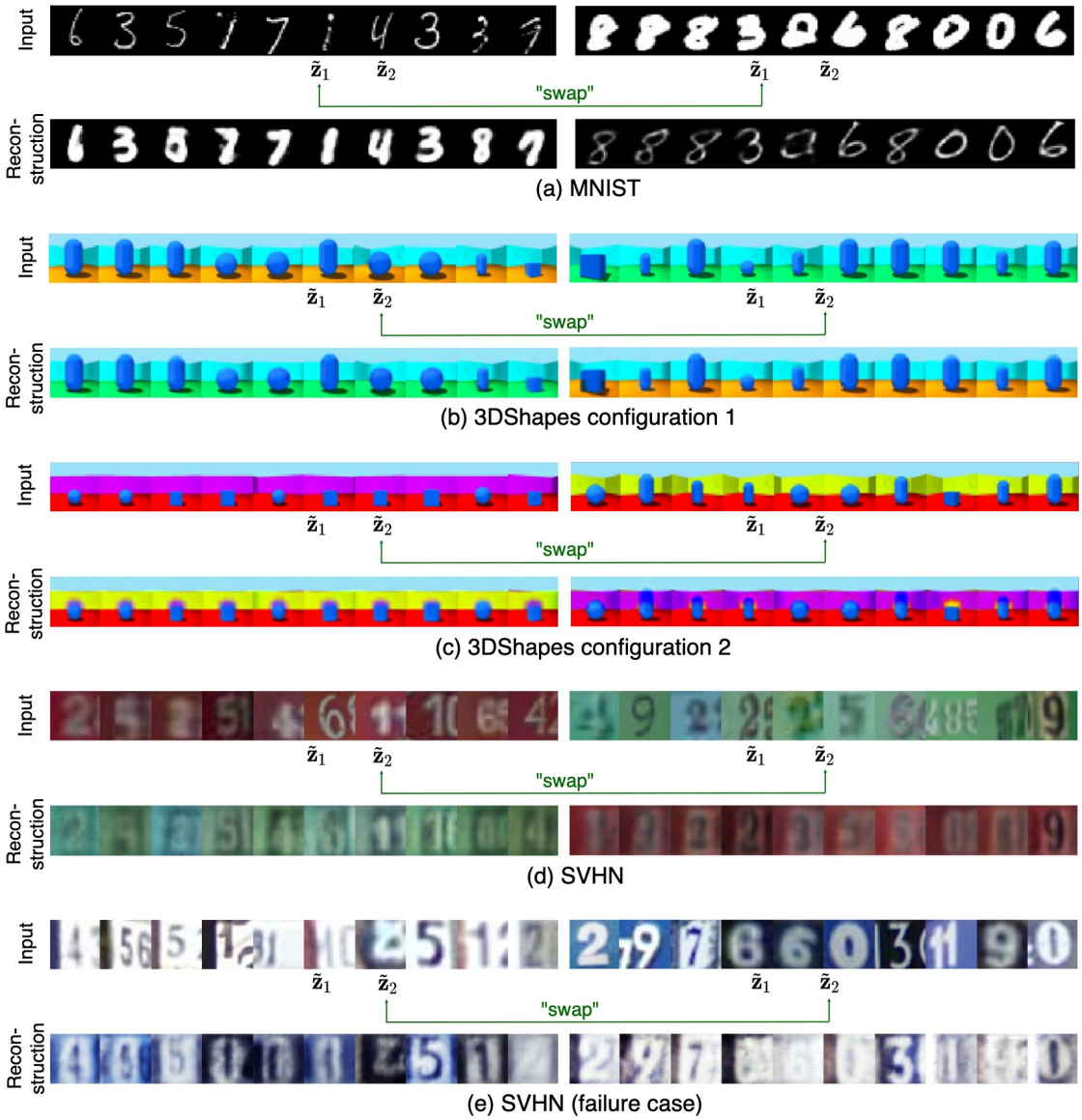


Figure A.11: Input examples and reconstructions when swapping their style/colour facet's latent representation.

A.5.5 Generative, unsupervised classification

To compare against assumed ground-truth clusterings imposed by the supervised class structure in our datasets, we report generative classification performance in terms of *unsupervised clustering accuracy* on the test set. When the number of clusters in a facet is equal to the number of ground-truth classes compared against, one can use the Hungarian algorithm to find the optimal 1-to-1 mapping between clusters and classes [108, 129]. When the number of clusters in a facet is greater than the number of ground-truth classes compared against, as is common, one can simply assign each cluster in a facet to the most frequent ground-truth class found within that cluster [236].

In Figs. A.12 to A.14, we plot generative classification performance on the test set measured in terms of unsupervised clustering accuracy over the training epochs. The blue shade is bounded by the mean accuracy plus and minus one standard deviation over the ten runs. The sudden jumps of accuracy after ≈ 100 epochs are caused by the progressive training algorithm which loops in a new facet at that point.

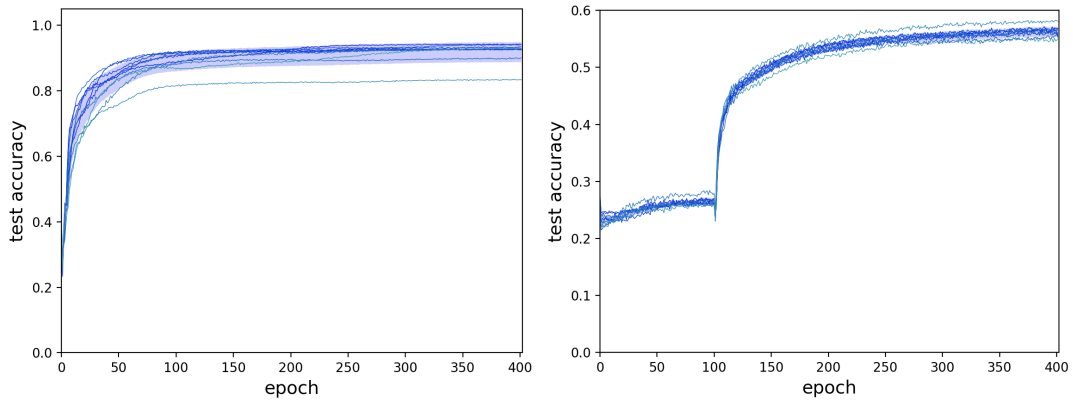


Figure A.12: Unsupervised clustering accuracy on the test set w.r.t. the supervised label, when trained on [Left] MNIST, and [Right] SVHN.

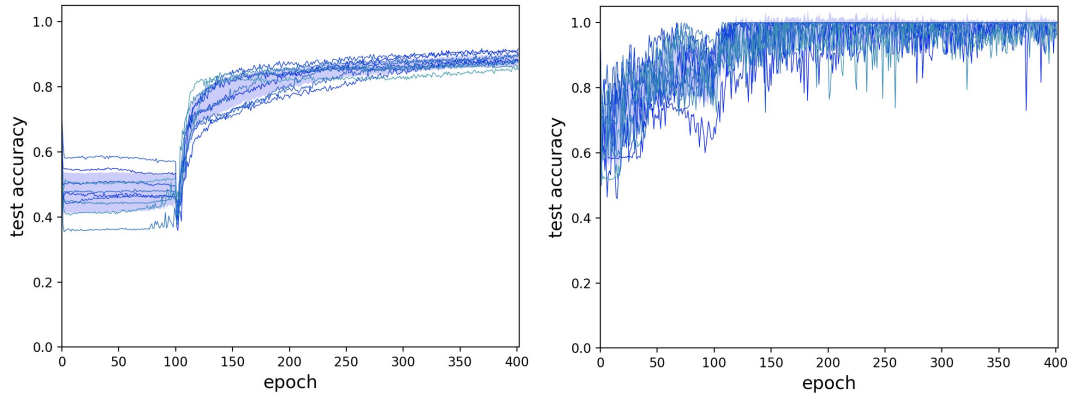


Figure A.13: Unsupervised clustering accuracy on the test set for [Left] object shape and [Right] floor colour, when trained on 3DShapes (config. 1).

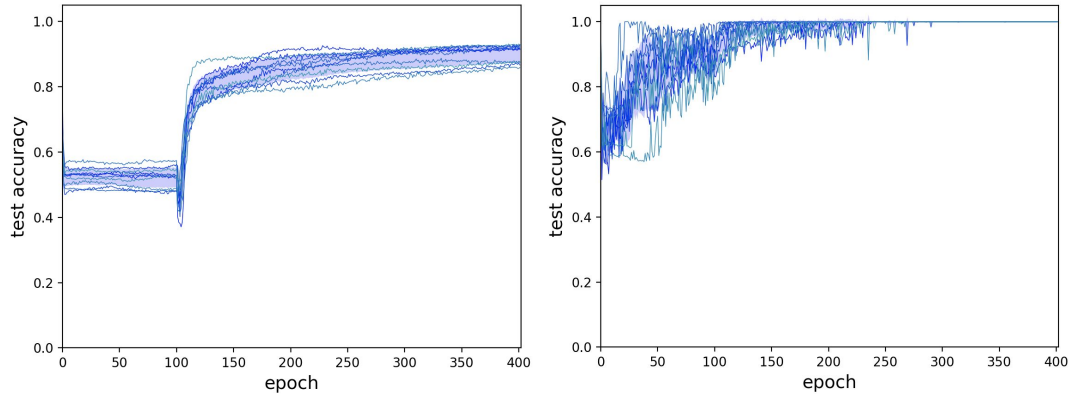


Figure A.14: Unsupervised clustering accuracy on the test set for [Left] object shape and [Right] wall colour, when trained on 3DShapes (config. 2).

A.5.6 Diversity of generated samples

First, we compare sample generation performance on MNIST between MFCVAE (a) and LTVAE (b) in Fig. A.15. We sample MFCVAE as discussed in Section 2.4.4. For LTVAE, while not fully clear, LTVAE samples from one Mixture-of-Gaussian, i.e. first samples from a categorical, then from the chosen component, to obtain $\mathbf{z}$ ([138], Section 4.5). This sampling procedure resembles MFCVAE's reconstructions that illustrate its digit facet (e.g. Fig. A.15 (a.1)). When comparing (a.1) and (b), it can be observed that generation performance is comparable between the two models, with each row representing a certain digit identity cluster. However, as shown in (a.2) and not demonstrated by LTVAE, our model additionally allows sample generation conditional on style clusters. This demonstrates the advantage of MFCVAE in its intervention capability for each facet during sample generation, allowing a rich set of options for potential downstream tasks.

To quantitatively compare the sampling diversity of MFCVAE against VaDE, we compute the Learned Perceptual Image Patch Similarity (LPIPS) [255] of 60,000 samples generated from models trained on MNIST and SVHN. We also report LPIPS computed on on real (i.e. non-synthetic) images, where we use the data from both training set and test set. For LPIPS, higher is better, while it must be ensured that the true data distribution is modelled (and not just a distribution that artificially maximises the metric). As shown in Table A.5, our method generates samples with similar diversity to VaDE on both datasets. Further, MFCVAE is close to the "real image diversity" for MNIST, yet is somewhat lower for SVHN.

Table A.5: LPIPS of real images and 60,000 samples generated from VaDE and MFCVAE for MNIST and SVHN.

	MNIST	SVHN
Real Images	0.112	0.227
VaDE	0.111	0.187
MFCVAE (ours)	0.116	0.182

In addition, Figs. A.16 to A.18 show the complete plots of synthetic samples generated from MFCVAE where all clusters are visualised compared to the main text. Again, we refer to Section 2.4.4 for an explanation on the procedure of how these samples are generated, but provide here additional details: During sample generation, the variance of the distributions over latent variables $\mathbf{z}_1$ and $\mathbf{z}_2$ are scaled by a "temperature" factor $\tau > 0$. This is a common technique in likelihood-based deep generative models to improve the quality of generated samples [118, 179]. To formalise this, at sampling time, the covariance matrix $\tau\Sigma_{c_j}$ is used for $p(\mathbf{z}_j|c_j)$, instead of Σ_{c_j}. In this set of experiments, temperature scaling is used for 3DShapes and SVHN, where we choose $\tau = 0.3$. For MNIST, we do not use temperature scaling, i.e. $\tau = 1.0$.

For MNIST and 3DShapes, it can be observed that for clusters with a lower average assignment probability, synthetic samples remain homogeneous w.r.t. their characteristic value in each facet. For SVHN, the sample reconstruction quality drops for clusters with lower average assignment probabilities which we can attribute to a smaller separation of facets on this dataset as observed in Section 2.4.3.

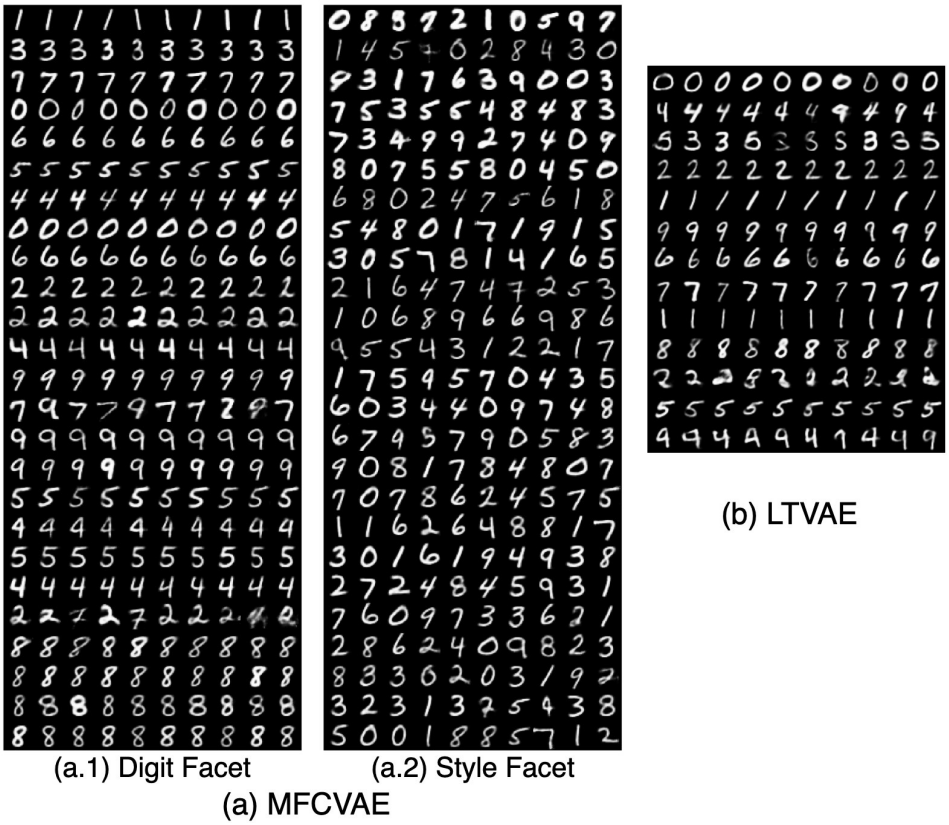


Figure A.15: (a) Synthetic samples generated from MFCVAE with two facets ($J = 2$) trained on MNIST, with all clusters visualised. Rows are sorted as in Fig. 2.6. (b) Synthetic samples generated from LTVAE trained on MNIST. Plot is taken as reported in [138], Fig. 7.

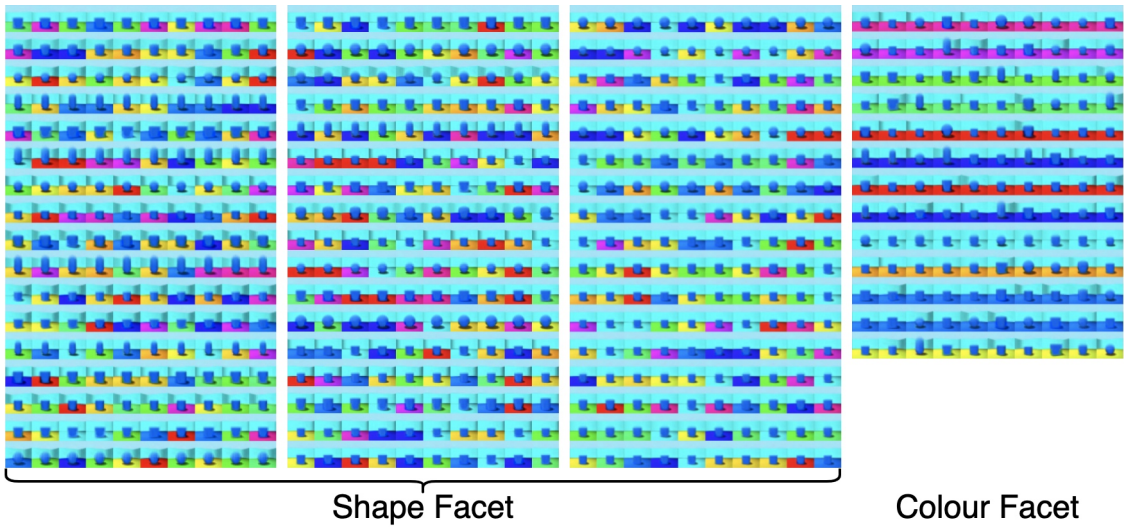


Figure A.16: Synthetic samples generated from MFCVAE with two facets ($J = 2$) trained on 3DShapes (configuration 1), with all clusters visualised. Rows are sorted as in Fig. 2.6.

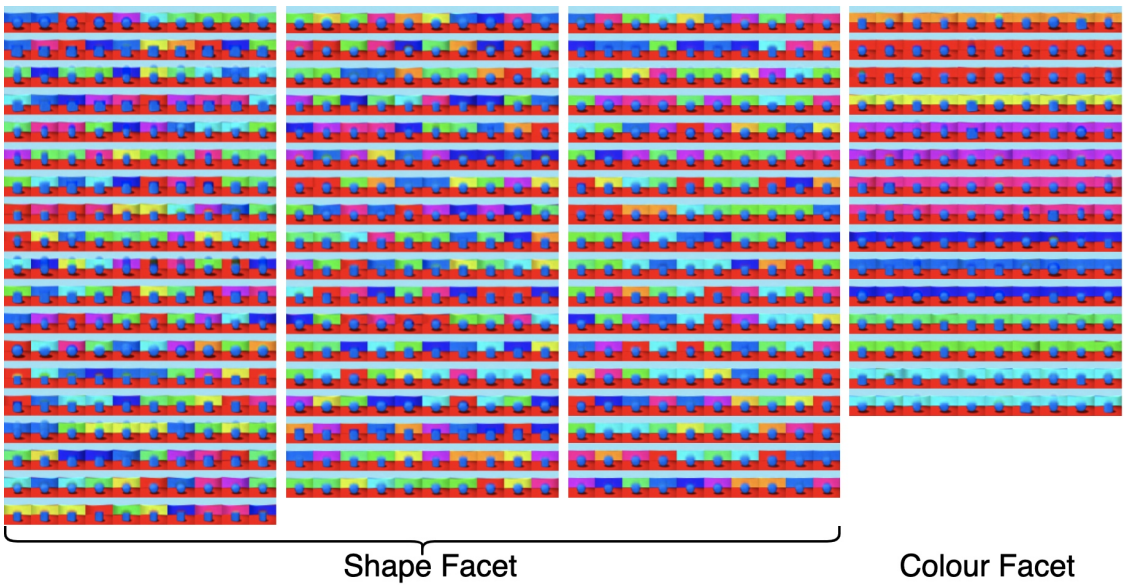


Figure A.17: Synthetic samples generated from MFCVAE with two facets ($J = 2$) trained on 3DShapes (configuration 2), with all clusters visualised. Rows are sorted as in Fig. 2.6.

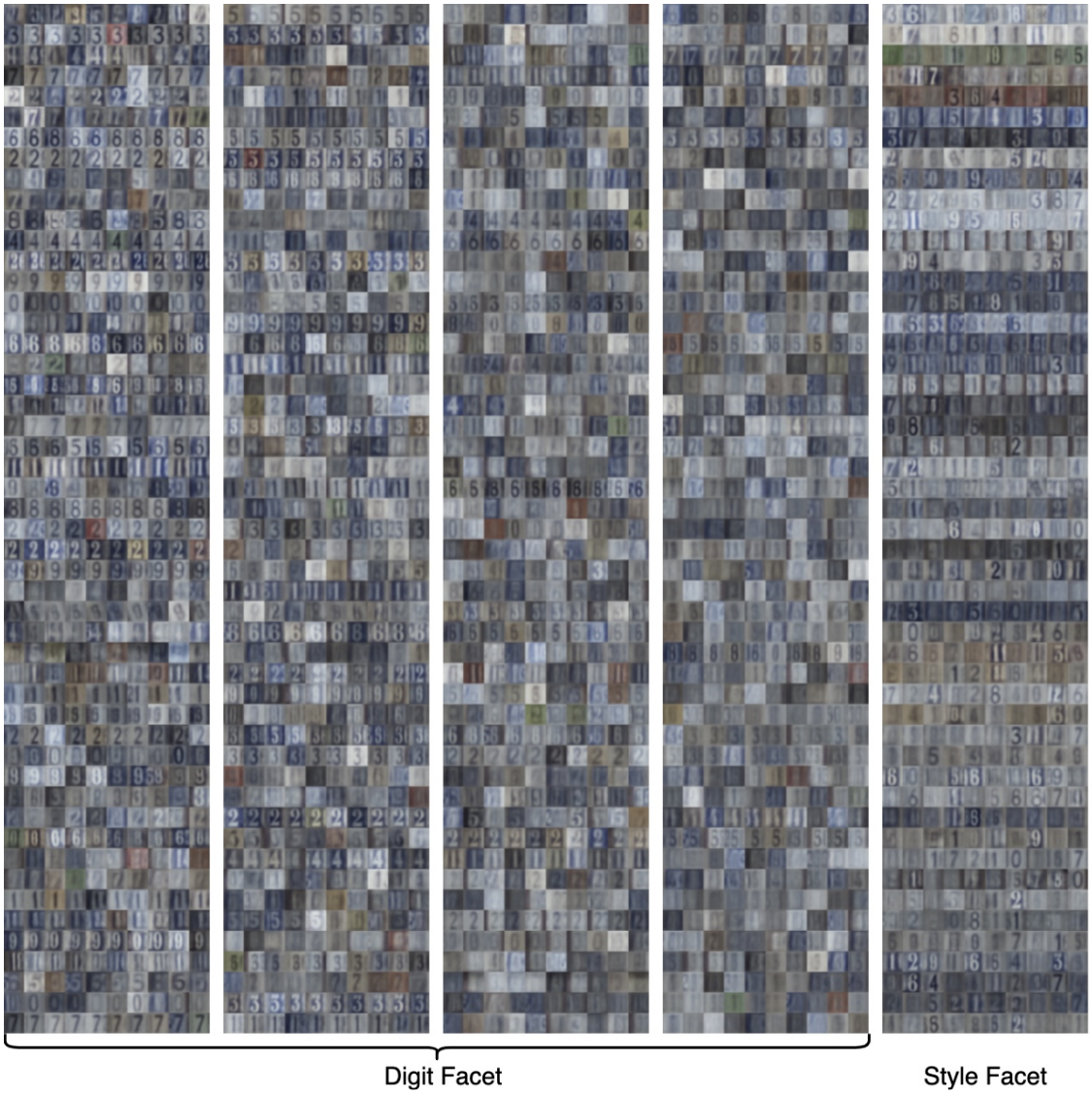


Figure A.18: Synthetic samples generated from MFCVAE with two facets ($J = 2$) trained on SVHN, with all clusters visualised. Rows are sorted as in Fig. 2.6.

B

Appendix of *A Multi-Resolution Framework for U-Nets with Applications to Hierarchical VAEs*

Contents

B.1 Framework Details and Technical Proofs

Here we provide proofs for the theorems in the main paper and additional theoretical results supporting these.

B.1.1 Definitions and Notations

The following provides an index of commonly used notation throughout this manuscript for reference.

The *function space* of interest in this work is $L^2(\mathbb{X})$, the space of square integrable functions, where $\mathbb{X}$ is a compact subset of $\mathbb{R}^m$ for some integer m, for instance, $\mathbb{X} = [0, 1]$. This set of functions is defined as

$$L^2(\mathbb{X}) = \{f : \mathbb{X} \to \mathbb{R} \,|\, \|f\|_2 < \infty,\ f \text{ Borel measurable}\}. \tag{B.1}$$

$L^2(\mathbb{X})$ forms a vector space with the standard operations.

We denote $V_{-j} \subset L^2(\mathbb{X})$ as a finite-dimensional approximation space. With the nesting property, $V_{-j+1} \subset V_{-j}$, the space U_{-j+1} is the orthogonal complement of V_{-j+1} within V_{-j}, i.e. $V_{-j} = U_{-j+1} \oplus V_{-j+1}$.

The *integration* shorthand notations used are as follows. For an integrable function $t \mapsto f(t)$, we use

$$f(t)dt := \int_0^t f(s)ds. \tag{B.2}$$

The function f may be multi-dimensional in which case we mean the multi-dimensional integral in whichever basis is being used. For *stochastic* integrals, we only analyse dynamics within the truncation V_{-J} of $L^2(\mathbb{X})$. In this case, W_t refers to a Brownian motion on the same amount of dimensions as V_{-J} in the *standard*, or 'pixel', basis of V_{-J}. The shorthand

$$g(W_t)dW_t := \int_0^t g(W_s)dW_s, \tag{B.3}$$

is used for the standard Itô integral. Last, for a stochastic process X_t on V_{-J} we mean the standard convention

$$\int_0^t dX_s = X_t - X_0. \tag{B.4}$$

For *measures*, we use $\mathbb{D}$ to prefix a set for which we consider the space of probability measures over: for instance, $\mathbb{D}(\mathbb{X})$ denotes the space of probability measures over $\mathbb{X}$. We often refer to measures over functions (i.e. images): recall that V_{-J} is an L^2-function space and we take $\mathbb{D}(V_{-J})$ to be probability measures over this space.

When referenced in Definition 3.2, the distance metric between two measures ν_1 and ν_2 which yields the topology of *weak continuity* is the *Monge–Kantorovich* metric [131, 188]

$$d_{\mathbb{P}}(\nu_1, \nu_2) = \sup_{f \in \mathrm{Lip}_1(\mathbb{X})} \int f d(\nu_1 - \nu_2), \tag{B.5}$$

where

$$\mathrm{Lip}_1(\mathbb{X}) = \{f : \mathbb{X} \to \mathbb{R} \Big| \, |f(x) - f(y)| \leq d(x, y), \forall x, y \in \mathbb{X}\}. \tag{B.6}$$

Further, we use the Wasserstein-2 metric which in comparison to the weak convergence above has additional moment assumptions. It is given by

$$\mathcal{W}_2(\nu_1, \nu_2) = \left(\inf_{\gamma \in \Gamma(\nu_1, \nu_2)} \mathbb{E} \|X_1 - X_2\|_2^2 \right)^{1/2}, \tag{B.7}$$

where $(X_1, X_2) \sim \gamma$ and $\Gamma(\nu_1, \nu_2)$ is the space of measures on $\mathbb{D}(V_{-J} \times V_{-J})$ with marginals ν_1 and ν_2.

B.1.2 Dimension Reduction Conjugacy

Assume momentarily the one dimensional case where $\mathbb{X} = [0, 1]$. Let V_{-j} be an *approximation space* contained in $L^2(\mathbb{X})$ (see Definition 3.1) pertaining to image pixel values

$$V_{-j} = \{f \in L^2([0,1]) \mid f_{[2^{-j} \cdot k, 2^{-j} \cdot (k+1))} = c_k, \, k \in \{0, \ldots, 2^j - 1\}, \, c_k \in \mathbb{R}\}. \tag{B.8}$$

For a function $f \in V_{-j}$, there are several ways to express f in different bases. Consider the *standard (or 'pixel') basis* for a fixed V_{-j} given via

$$e_{j,k} = \mathbb{1}_{[2^{-j}\cdot k, 2^{-j}\cdot(k+1)]}. \tag{B.9}$$

Clearly, the family $\mathbf{E}_j \coloneqq \{e_{j,k}\}_{k=0}^{2^j-1}$ is an orthogonal basis of V_{-j}, hence full rank with dimension 2^j. Functions in V_{-j} may be expressed as

$$f = \sum_{k=0}^{2^j-1} c_k \cdot e_{j,k}, \tag{B.10}$$

for $c_k \in \mathbb{R}$.

First, let us recall the average *pooling* operation in these bases $\mathbf{E}_j$ and $\mathbf{E}_{j-1}$ of V_{-j} and V_{-j+1}, where $\text{pool}_{-j,-j+1} : V_{-j} \to V_{-j+1}$. Its operation is given by

$$\text{pool}_{-j,-j+1}(f) = \text{pool}_{-j,-j+1}\left(\sum_{k=0}^{2^j-1} c_k \cdot e_{j,k}\right) = \sum_{i=0}^{2^{j-1}-1} \tilde{c}_i \cdot e_{j-1,i}, \tag{B.11}$$

where for $i \in \{0, \dots, 2^{j-1}-1\}$ we have the coefficient relation

$$\tilde{c}_i = \frac{c_{2i} + c_{2i+1}}{2} = \frac{1}{2^{-j}} \int_{[2^{-j}\cdot(2i), 2^{-j}\cdot(2i+1))} f(x)dx. \tag{B.12}$$

Average pooling and its imposed basis representation are commonly used in U-Net architectures [197], for instance in state-of-the-art diffusion models [92] and HVAEs [34].

Note that across approximation spaces of two resolutions V_{-j} and V_{-j+1}, the standard bases $\mathbf{E}_j$ and $\mathbf{E}_{j-1}$ share no basis elements. As basis elements change at each resolution, it is difficult to analyse V_{-j} embedded in V_{-j+1}. What we seek is a basis for all V_{-j} such that any basis element in this set at resolution j is also a basis element in V_{-J}, the approximation space of highest resolution J we consider. This is where wavelets serve their purpose: We consider a *multi-resolution (or 'wavelet') basis* of V_{-J} [154]. For the purpose of our theoretical results below, we are here focusing on a *Haar wavelet* basis [81] which we introduce in the following, but note that our framework straightforwardly generalises to other wavelet bases. Begin with

$\phi_1 = \mathbb{1}_{[0,1)}$ as L^2-basis element for V_{-1}, the space of constant functions on $[0,1)$. For V_{-2} we have the space of L^2 functions which are constant on $[0,1/2)$ and $[1/2,1)$, which we receive by adding the basis element $\psi = \sqrt{2}(\mathbb{1}_{[0,1/2)} - \mathbb{1}_{[1/2,1)})$. Here ϕ_1 is known as the father wavelet, and ψ as the mother wavelet. To make a basis for general V_{-j} we localise the mother wavelet with scaling and translation, i.e

$$\psi_{i,k} = 2^{-i/2} \cdot \psi(2^i(\,\cdot\, - k)) \quad \text{where } i \in \{0, j\}, k \in \{0, 2^{-i+1}\}. \tag{B.13}$$

It is straight-forward to check that $\mathbf{\Psi}_j \coloneqq \{\psi_{i,k}\}_{i=0,k=0}^{j,2^{i-1}}$ is an orthonormal basis of V_{-j} on $[0,1]$. Further, the truncated basis $\mathbf{\Psi}_{j-1}$, which is a basis for V_{-j+1}, is contained in the basis $\mathbf{\Psi}_j$. This is in contrast to $\mathbf{E}_{j-1}$ which has basis elements distinct from the elements in the basis $\mathbf{E}_j$ on a higher resolution.

The collections $\mathbf{E}_j$ and $\mathbf{\Psi}_j$ both constitute full-rank bases for V_{-j}. They further have the same dimension and so there is a linear isomorphism $\pi_j : V_{-j} \to V_{-j}$ for change of basis, i.e.

$$\pi_j(e_{j,i}) = \psi_{j,i}. \tag{B.14}$$

This can be normalised to be an isometry. We now analyse the pooling operation in our basis $\mathbf{\Psi}_j$, restating Theorem 3.2 from the main text and providing a proof.

Theorem 3.2. Given V_{-j} as in Definition 3.1, let $x \in V_{-j}$ be represented in the standard basis $\mathbf{E}_j$ and Haar basis $\mathbf{\Psi}_j$. Let $\pi_j : \mathbf{E}_j \mapsto \mathbf{\Psi}_j$ be the change of basis map illustrated in Fig. 3.3, then we have the conjugacy $\pi_{j-1} \circ \text{pool}_{-j,-j+1} = \text{proj}_{V_{-j+1}} \circ \pi_j$.

Proof. Define the conjugate pooling map in the wavelet basis, $\text{pool}^*_{j,j+1} : V_{-j} \to V_{-j+1}$ computed on the bases $\mathbf{\Psi}_j$ and $\mathbf{\Psi}_{j-1}$,

$$\text{pool}^*_{-j,-j+1} \coloneqq \pi_{j-1} \circ \text{pool}_{-j,-j+1} \circ \pi_j^{-1}. \tag{B.15}$$

Due to the scaling and translation construction in Eq. (B.13) and because the pooling operation is local, we need only consider the case for $\text{pool}_{-2,-1}$. This is

$$\begin{array}{ccc} (V_{-j}, \mathbf{E}_j) & \xrightarrow{\text{pool}_{-j,-j+1}} & (V_{-j+1}, \mathbf{E}_{j-1}) \\ \uparrow \pi_j^{-1} & & \pi_{j-1} \downarrow \\ (V_{-j}, \mathbf{\Psi}_j) & \xrightarrow[\text{pool}^*_{-j,-j+1}]{} & (V_{-j+1}, \mathbf{\Psi}_{j-1}) \end{array}$$

because one can view pooling between the higher-resolution spaces as multiple localised pooling operations between V_{-2} and V_{-1}. Now note that $\text{pool}_{-2,-1}$ maps V_{-2} to V_{-1}. Further,

$$\int_{\mathbb{X}} \psi(x)dx = 0, \tag{B.16}$$

where $\psi = \sqrt{2}(\mathbb{1}_{[0,1/2)} - \mathbb{1}_{[1/2,1)})$ is the mother wavelet. For $v \in V_{-2}$ let v have the wavelet representation $v = \tilde{c}_2\psi + \tilde{c}_1\phi_1$, where $\phi_1 = \mathbb{1}_{[0,1)}$ is the father wavelet. To pool we compute the average of the two coefficients ('pixel values')

$$\text{pool}^*_{-2,-1}(v) = \int_{\mathbb{X}} v(x)dx = \int_{\mathbb{X}} \tilde{c}_2\psi(x) + \tilde{c}_1\phi_1(x)dx = \tilde{c}_1. \tag{B.17}$$

Thus average pooling here corresponds to truncation of the wavelet basis for V_{-2} to the wavelet basis for V_{-1}. As this basis is orthonormal over $L^2(\mathbb{X})$, truncation corresponds to L^2 projection, i.e. $\text{pool}^*_{-j,-j+1} = \text{proj}_{V_{-j+1}}$, as claimed. □

Theorem 3.2 shows that the pooling operation is conjugate to projection in the Haar wavelet approximation space, and computed by truncation in the Haar wavelet basis. The only quantity we needed for our basis over the V_{-j} was the vanishing moment quantity

$$\int_{\mathbb{X}} \psi(x)dx = 0. \tag{B.18}$$

To extend this property to higher dimensions, such as the two dimensions of gray-scale images, we use the tensor product of $[0, 1]$, and further, the tensor product of basis functions. This property is preserved, and hence the associated average pooling operation is preserved on the tensor product wavelet basis, too. To further extend it to color images, one may consider the cartesian product of several L2 spaces.

B.1.3 Average pooling Truncation Error

In this section we prove Theorem 3.3, which quantifies the regularisation imposed by an average pooling bottleneck trained by minimising the reconstruction error. The proof structure is as follows: First we give an intuition for autoencoders with an average pooling bottleneck, then derive the relevant assumptions for Theorem 3.3. We next prove our result under strong assumptions. Last, we weaken our assumptions so that our theorem is relevant to HVAE architectures.

Suppose we train an autoencoder on V_{-j} without dimension reduction, calling the parameterised forward (or encoder/bottom-up) and backward (or decoder/top-down) passes $F_{j,\phi}$, $B_{j,\theta} : V_{-j} \mapsto V_{-j}$ respectively. We can optimise $F_{j,\phi}$ and $B_{j,\theta}$ w.r.t. ϕ and θ to find a perfect reconstruction, i.e. $x = B_{j,\theta}F_{j,\phi}x$ for all x in our data as there is no bottleneck (no dimensionality reduction): $B_{j,\theta}$ need only be a left inverse of $F_{j,\phi}$, as in

$$B_{j,\theta}F_{j,\phi} = I. \tag{B.19}$$

Importantly, we can choose $F_{j,\phi}$ and $B_{j,\theta}$ satisfying B.19 *independent* of our data. For instance, they could both be the identity operator and achieve perfect reconstruction, but contain no information about the generative characteristics of our data. Compare this to a *bottleneck* with average pooling, i.e. an autoencoder with dimension reduction. Here, we consider the dimension reduction from V_{-j} to V_{-j+1}, where we split $V_{-j} = V_{-j+1} \oplus U_{-j+1}$. As we have seen in Theorem 3.2, through average pooling, we keep information in V_{-j+1}, and discard the information in U_{-j+1}. For simplicity, let $\text{embd}_{V_{-j}}$ be the inclusion of the projection $\text{proj}_{V_{-j+1}}$. Now to achieve perfect reconstruction

$$x = (B_{j,\theta} \circ \text{embd}_{V_{-j}} \circ \text{proj}_{V_{-j+1}} \circ F_{j,\phi})x, \tag{B.20}$$

we require $(\text{proj}_{U_{-j+1}}F_{j,\phi})x = 0$. Simply put, the encoder $F_{j,\phi}$ should make sure that the discarded information in the bottleneck is nullified.

We may marry this observation with a simple U-Net structure (without skip connection) with L^2-reconstruction and average pooling dimension reduction. Let V_{-j} be one of our multi-resolution approximation spaces and $\mathbb{D}(V_{-j})$ be the space of probability measures over V_{-j}. Recall in a multi-resolution basis we have $V_{-j} = V_{-j+1} \oplus U_{-j+1}$ where U_{-j+1} is the $-j+1$ orthogonal compliment within V_{-j}. For any $v \in V_{-j}$ we may write $v = \text{proj}_{V_{-j+1}} v \oplus \text{proj}_{U_{-j+1}} v$ and analyse the truncation error in V_{-j+1}, i.e. the discarded information, via

$$\|v - \text{embd}_{V_{-j}} \circ \text{proj}_{V_{-j+1}} v\|_2^2 = \|\text{proj}_{U_{-j+1}} v\|_2^2. \tag{B.21}$$

If we normalise this value to

$$\frac{\|v - \text{embd}_{V_{-j}} \circ \text{proj}_{V_{-j+1}} v\|_2^2}{\|v\|_2^2} = \frac{\|\text{proj}_{U_{-j+1}} v\|_2^2}{\|v\|_2^2} \in [0, 1], \tag{B.22}$$

then this is zero when v is non-zero only within V_{-j+1} and zero everywhere within U_{-j+1}. Suppose now that we have a measure $\nu_j \in \mathbb{D}(V_{-j})$, we could quantify *how much* of the norm of a sample from ν_j comes from the U_{-j+1} components by computing

$$\mathbb{E}_{v \sim \nu_j} \frac{\|\text{proj}_{U_{-j+1}} v\|_2^2}{\|v\|_2^2} = \int \frac{\|\text{proj}_{U_{-j+1}} v\|_2^2}{\|v\|_2^2} d\nu_j(v) \in [0, 1]. \tag{B.23}$$

This value forms a convex sum with its complement projection to $\text{proj}_{V_{-j+1}}$, demonstrating the splitting of mass across V_{-j+1} and U_{-j+1}, as we show in Lemma B.1.

Lemma B.1. Let $\nu_j \in \mathbb{D}(V_{-j})$ be atom-less at 0, then

$$\mathbb{E}_{v \sim \nu_j} \frac{\|\text{proj}_{V_{-j+1}} v\|_2^2}{\|v\|_2^2} + \mathbb{E}_{v \sim \nu_j} \frac{\|\text{proj}_{U_{-j+1}} v\|_2^2}{\|v\|_2^2} = 1. \tag{B.24}$$

Proof. For any $v \in V_{-j}$ we have $\|v\|_2^2 = \|\text{proj}_{V_{-j+1}} v\|_2^2 + \|\text{proj}_{U_{-j+1}} v\|_2^2$ due to orthogonality of V_{-j+1} and U_{-j+1}. As both $\|\text{proj}_{V_{-j+1}} v\|_2^2$ and $\|\text{proj}_{U_{-j+1}} v\|_2^2$ are projections, they are bounded by $\|v\|_2^2$ giving that the integrands in Eq. (B.24) are bounded by one, and so for all $v \neq 0$ (no point mass at 0) the expectation is bounded. □

From the splitting behaviour of masses in the L^2-norm observed in Lemma B.1 we see that

1. if $\mathbb{E}_{v\sim\nu_j}\|\text{proj}_{U_{-j+1}}v\|_2^2/\|v\|_2^2$ is large, then, on average, samples from ν_j have most of their size in the U_{-j+1} subspace; or,
2. if $\mathbb{E}_{v\sim\nu_j}\|\text{proj}_{U_{-j+1}}v\|_2^2/\|v\|_2^2$ is small, then, on average, samples from ν_j have most of their size in the V_{-j+1} subspace.

In the latter case, $\|\text{proj}_{U_{-j+1}}v\|_2^2 \approx 0$, i.e. $\text{embd}_{V_{-j}} \circ \text{proj}_{V_{-j+1}}v \approx v$. We get the heuristic $\text{embd}_{V_{-j}} \circ \text{proj}_{V_{-j+1}} \approx I$ on the measure ν_j, yielding a perfect reconstruction.

Let $\text{embd}_{V_{-j}} \circ \text{proj}_{V_{-j+1}}, I : V_{-j} \to V_{-j}$, then this heuristic performs the operator approximation

$$\mathbb{E}_{v\sim\nu_j}\|(\text{embd}_{V_{-j}} \circ \text{proj}_{V_{-j+1}} - I)v\|_2^2, \tag{B.25}$$

quantifying 'how close' these operators are on ν_j. For many measures, this (near) equivalence between operators will not hold. But what if instead, we had an operator $D : V_{-j} \to V_{-j}$ such that the push-forward of ν_j through this operator had this quality. Practically, this push-forward operator will be parameterised by neural networks, for instance later in the context of U-Nets. For simplicity, we will initially consider the case where D is linear on V_{-j}, then we consider when D is Lipschitz.

Lemma B.2. Given V_{-j} with the L^2-orthogonal decomposition $V_{-j} = V_{-j+1} \oplus U_{-j+1}$, let $D_{-j} : V_{-j} \to V_{-j}$ be an invertible linear operator and define $F_j : V_{-j} \to V_{-j+1}$ and $B_j : V_{-j+1} \to V_{-j}$ through

$$F_j = \text{proj}_{V_{-j+1}} \circ D_j, \qquad B_j = D_j^{-1} \circ \text{embd}_{V_{-j}}. \tag{B.26}$$

Then $B_jF_j \equiv I$ on V_{-j}, or otherwise, we have the truncation bound

$$\frac{\left\|\text{proj}_{U_{-j+1}}D_jv\right\|_2^2}{\|D_j\|_2^2} \leq \|(I - B_jF_j)v\|_2^2\,. \tag{B.27}$$

Proof. Consider the operator $D_j(I - B_jF_j) : V_{-j} \to V_{-j}$ which is linear and obeys the multiplicative bound $\|D_j(I - B_jF_j)\| \leq \|D_j\| \, \|I - B_jF_j\|$. This implies for any $v \in V_{-j}$,

$$\frac{\|D_j(I - B_jF_j)v\|_2^2}{\|D_j\|_2^2} \leq \|(I - B_jF_j)v\|_2^2 . \tag{B.28}$$

The numerator is equal to

$$\|D_j(I - B_jF_j)v\|_2^2 = \left\|(D_j - \mathrm{embd}_{V_{-j}} \circ \mathrm{proj}_{V_{-j+1}} \circ D_j)v\right\|_2^2 . \tag{B.29}$$

As we have the orthogonal decomposition $V_{-j} = V_{-j+1} \oplus U_{-j+1}$, we know

$$I = \mathrm{proj}_{V_{-j+1}} \oplus \mathrm{proj}_{U_{-j+1}} \tag{B.30}$$

$$= \mathrm{embd}_{V_{-j}} \circ \mathrm{proj}_{V_{-j+1}} + \mathrm{embd}_{V_{-j}} \circ \mathrm{proj}_{U_{-j+1}}, \tag{B.31}$$

and as $\left\|\mathrm{embd}_{V_{-j}}\right\|_2 = 1$, we get

$$\left\|(I - \mathrm{embd}_{V_{-j}} \circ \mathrm{proj}_{V_{-j+1}} \circ D_j)v\right\|_2^2 = \left\|\mathrm{proj}_{U_{-j+1}} \circ D_j v\right\|_2^2 . \tag{B.32}$$

So now as $\|D_j(I - B_jF_j)v\|_2^2 = \left\|\mathrm{proj}_{U_{-j+1}} \circ D_j v\right\|_2^2$, we may use $\|D_j(I - B_jF_j)v\|_2^2 \leq \|D_j\|^2 \, \|I - B_jF_jv\|_2^2$ to get the desired result. □

The quantity $\left\|\mathrm{proj}_{U_{-j+1}} F_j v\right\|_2^2 / \|F_j\|_2^2$ is analogous to the in Lemma B.1 discussed quantity $\|\mathrm{proj}_{U_{-j+1}} v\|_2^2 / \|v\|_2^2$, but we now have a 'free parameter', the operator D_j.

Next, suppose D_j is trainable with parameters θ. We do so by minimising the reconstruction cost

$$\mathbb{E}_{v \sim \nu_j} \|(I - B_jF_j)v\|_2^2 , \tag{B.33}$$

which upper-bounds our 'closeness metric' in Lemma B.2.

In the linear case (D_j is linear), to ensure that $D_{j,\theta}$ is invertible we may parameterise it by an (unnormalised) LU-decomposition of the identity

$$I = D_{j,\theta}^{-1} D_{j,\theta} = L_{j,\theta} U_{j,\theta}, \tag{B.34}$$

where the diagonal entries of $L_{j,\theta}$ and $U_{j,\theta}$ are necessarily inverses of one-another. This is a natural parameterisation when considering a U-Net with dimensionality reduction. Building from Lemma B.2, we can now consider the stacked U-Net (without skip connections), i.e. a U-Net with multiple downsampling/upsampling and forward/backward operators stacked on top of each other, in the linear setting. In Proposition B.1, we show that this LU-parameterisation forces the pivots of $U_{j,\theta}$ to tend toward zero.

Proposition B.1. Let $\{V_{-j}\}_{j=0}^{J}$ be a multi-resolution hierarchy of V_{-J} with the orthogonal decompositions $V_{-j} = V_{-j+1} \oplus U_{-j+1}$ and $F_{j,\phi}$, $B_{j,\theta} : V_{-j} \to V_{-j}$ be bounded linear operators such that $B_{j,\theta}F_{j,\phi} = I$. Define $\boldsymbol{F}_{j,\phi} : V_{-j} \to V_{-j+1}$ and $\boldsymbol{B}_{j,\theta} : V_{-j+1} \to V_{-j}$ by

$$\boldsymbol{F}_{j,\phi} \coloneqq \mathrm{proj}_{V_{-j+1}} \circ F_{j,\phi}, \qquad \boldsymbol{B}_{j,\theta} \coloneqq B_{j,\theta} \circ \mathrm{embd}_{V_{-j}}, \tag{B.35}$$

with compositions

$$\boldsymbol{F}_{j_1|j_2,\phi} \coloneqq \boldsymbol{F}_{j_1,\phi} \circ \cdots \circ \boldsymbol{F}_{j_2,\phi}, \qquad \boldsymbol{B}_{j_1|j_2,\phi} \coloneqq \boldsymbol{B}_{j_1,\phi} \circ \cdots \circ \boldsymbol{B}_{j_2,\phi}. \tag{B.36}$$

Then

$$\sum_{j=1}^{J} \frac{\left\|\mathrm{proj}_{U_{-j+1}} F_j v\right\|_2^2}{\|F_j\|_2^2} \leq \left\|(I - \boldsymbol{B}_{1|J,\theta}\boldsymbol{F}_{1|J,\phi})v\right\|_2^2. \tag{B.37}$$

Proof. The operator $\mathbf{F}_{1|J}$ is linear, and decomposes into a block operator form with pivots $\mathbf{F}_{j|J}$ for each $j \in \{1, \ldots, J\}$. Each $\mathbf{F}_{j|J}$ is L^2-operator norm bounded by $\|F_j\|_2$, so if

$$\lambda_{1|J} \coloneqq \mathrm{diag}(\|F_1\|_2, \ldots, \|F_J\|_2), \tag{B.38}$$

then $\|\lambda_{1|J}^{-1}\mathbf{F}_{1|J}\|_2 \leq 1$. Last, as the spaces $\{U_{-j}\}_{j=0}^{J}$ are orthogonal and $\mathbf{F}_{1|J}$ has triangular form:

$$\|\lambda_{1|J}^{-1}(F_{1|J} - \mathbf{F}_{1|J})v\|_2^2 = \sum_{j=1}^{J} \frac{\left\|\mathrm{proj}_{U_{-j+1}} F_j v\right\|_2^2}{\|F_j\|_2^2}, \tag{B.39}$$

and $\|\lambda_{1|J}^{-1}(F_{1|J} - \mathbf{F}_{1|J})v\|_2^2 \leq \|(I - \mathbf{B}_{1|J}\mathbf{F}_{1|J})v\|_2^2$. □

Here in the linear case, a U-Net's encoder is a triangular matrix where the basis vectors are the Haar wavelets. Proposition B.1 states that the pivots of this matrix are minimised. Adversely, this diminishes the rank of the autoencoder and pushes our original underdetermined problem to a singular one. In other words, the U-Net is in this case demanding to approximate the identity (via an LU-like-decomposition), a linear operator, with an operator of diminishing rank.

Proposition B.2. Let $\mathbb{D}(\mathbb{X})$ be the space of probability measures over $\mathbb{X}$, and assume for $\overline{F}_j, \overline{B}_j : \mathbb{D}(\mathbb{X}) \to \mathbb{D}(\mathbb{X})$ that these are inverses of one-another and $\overline{F}_j$ is Lipschitz, that is

$$\overline{F}_j \overline{B}_j = I, \qquad \mathcal{W}_2(\overline{F}_j \nu_1, \overline{F}_j \nu_2) \leq \|\overline{F}_j\|_2 W_2(\nu_1, \nu_2). \tag{B.40}$$

Then for any $\nu \in \mathbb{D}(\mathbb{X})$ with bounded second moment,

$$\mathbb{E}_{X_j \sim \overline{F}_j \nu} \frac{\|\mathrm{proj}_{U_{-j}} X_j\|_2^2}{\|\overline{F}_j\|_2^2} \leq \mathcal{W}_2(\nu, \overline{B}_j \circ P_{V_{-j}} \circ \overline{F}_j \nu). \tag{B.41}$$

Proof. First as $\overline{F}_j \overline{B}_j = I$ we know that

$$\mathcal{W}_2(\overline{F}_j \nu, P_{V_{-j}} \overline{F}_j \nu) = \mathcal{W}_2(\overline{F}_j \overline{B}_j \overline{F}_j \nu, \overline{F}_j \overline{B}_j P_{V_{-j}} \overline{F}_j \nu). \tag{B.42}$$

But for any $X \in V_{-j}$ we have the orthogonal decomposition

$$X = \mathrm{proj}_{V_{-j}} X \oplus \mathrm{proj}_{U_{-j}} X, \tag{B.43}$$

which respects the L^2-norm by

$$\|X\|_2^2 = \|\mathrm{proj}_{V_{-j}} X\|_2^2 + \|\mathrm{proj}_{U_{-j}} X\|^2, \tag{B.44}$$

and in particular,

$$\|X - \mathrm{proj}_{V_{-j}} X\|_2^2 = \|\mathrm{proj}_{U_{-j}} X\|^2. \tag{B.45}$$

This grants

$$(W_2(\overline{F}_j\nu, P_{V_{-j}}\overline{F}_j\nu))^2 = \inf_{\gamma\in\Gamma(\overline{F}_j\nu, P_{V_{-j}}\overline{F}_j\nu)} \mathbb{E}\|X-Y\|_2^2 \tag{B.46}$$

$$= \inf_{\gamma\in\Gamma(\overline{F}_j\nu, P_{V_{-j}}\overline{F}_j\nu)} \mathbb{E}\|\text{proj}_{V_{-j}}X - \text{proj}_{V_{-j}}Y\|_2^2 + \|\text{proj}_{U_{-j}}X\|_2^2 \tag{B.47}$$

$$= \inf_{\gamma\in\Gamma(\overline{F}_j\nu, P_{V_{-j}}\overline{F}_j\nu)} \mathbb{E}\|\text{proj}_{U_{-j}}X\|_2^2 \tag{B.48}$$

$$= (\mathcal{W}_2(\text{proj}_{U_{-j}}\overline{F}_j\nu, \delta_{\{0\}}))^2. \tag{B.49}$$

Now the Lipschitz of $\overline{F}_j$ yields

$$\frac{\mathcal{W}_2(\overline{F}_j\overline{B}_j\overline{F}_j\nu, \overline{F}_j\overline{B}_jP_{V_{-j}}\overline{F}_j\nu)}{\|\overline{F}_j\|_2} \leq \mathcal{W}_2(\overline{B}_j\overline{F}_j\nu, \overline{B}_jP_{V_{-j}}\overline{F}_j\nu) = \mathcal{W}_2(\nu, \overline{B}_jP_{V_{-j}}\overline{F}_j\nu). \tag{B.50}$$

Squaring and substituting grants

$$\frac{(\mathcal{W}_2(\text{proj}_{U_{-j}}\overline{F}_j\nu, \delta_{\{0\}}))^2}{\|\overline{F}_j\|_2^2} \leq \mathcal{W}_2(\nu, \overline{B}_jP_{V_{-j}}\overline{F}_j\nu). \tag{B.51}$$

□

To work on multiple resolution spaces, we need to define what the triangular operator over our space of measures is. For a cylinder set B on $V_{-J} = V_0 \oplus \bigoplus_{j=0}^{J} U_{-j}$ we can assume it has the form $\bigotimes_j B_j$ where B_j is a cylinder on U_j. Break ν_J into the multi-resolution sub-spaces by defining projection onto $\mathbb{D}(U_{-j})$ through

$$\text{proj}_{U_{-j}}(\nu_J)(B_j) \coloneqq \nu_J(B_j \otimes U_{-j}^{\perp}), \tag{B.52}$$

where B_j is a cylinder for U_{-j}. This projection of measures is respected by evaluation in that

$$\mathbb{E}_{X_j\sim\text{proj}_{U_{-j}}\nu_J}X_j = \int v_j d\text{proj}_{U_{-j}}\nu_J(v_j) = \int \text{proj}_{U_{-j}}v d\nu_J(v) = \mathbb{E}_{X_j\sim\nu_J}\text{proj}_{U_{-j}}X. \tag{B.53}$$

As $\|X\|_2^2 = \sum_j \text{proj}_{U_{-j}}\|\text{proj}_{U_{-j}}X\|_2^2$ due to the orthogonality of the spaces, then

$$\mathbb{E}_{X_j\sim\text{proj}_{U_{-j}}\nu_J}\|X_j\|_2^2 = \sum_j \mathbb{E}_{X_j\sim\text{proj}_{U_{-j}}\nu_J}\|X_j\|_2^2 = \sum_j \mathbb{E}_{X\sim\nu_J}\|\text{proj}_{U_{-j}}X\|_2^2. \tag{B.54}$$

Define the extension, with a convenient abuse of notation, of $\mathrm{proj}_{V_{-j+1}}$ on $\mathbb{D}(V_{-J})$ to be

$$\mathrm{proj}_{V_{-j+1}}(\nu_J) \coloneqq \mathrm{proj}_{V_{-j+1}}(\nu_J) \otimes \mathrm{proj}_{V_{-j+1}^{\perp}}(\nu_J). \tag{B.55}$$

If $F_{-j} : \mathbb{D}(V_{-j}) \to \mathbb{D}(V_{-j})$ are linear operators for $j \in \{0, \ldots, J\}$, extend each $F_j : \mathbb{D}(V_{-j}) \to \mathbb{D}(V_{-j})$ to $\mathbb{D}(V_{-j}) \times \mathbb{D}(V_{-j}^{\perp})$ through

$$\overline{F}_j \coloneqq F_j \oplus I. \tag{B.56}$$

For a measure $\nu_J \in \mathbb{D}(V_{-J})$ we can split it into $\mathbb{D}(V_{-j}) \times \mathbb{D}(V_{-j}^{\perp})$ via

$$\mathrm{proj}_{V_{-j}} \nu_J \times \mathrm{proj}_{V_{-j}^{\perp}} \nu_J, \tag{B.57}$$

which also remains a measure in $\mathbb{D}(V_{-J})$ as $\mathbb{D}(V_{-j}) \times \mathbb{D}(V_{-j}^{\perp}) \subset \mathbb{D}(V_{-J})$. Now the operator $\overline{F}_j$ acts on the product measure $\nu_j \otimes \nu_j^{\perp}$ by

$$\overline{F}_j(\nu_j \otimes \nu_j^{\perp}) = F_j\, \nu_j \otimes I\, \nu_j^{\perp}. \tag{B.58}$$

Now we may define the map $\boldsymbol{F}_j : \mathbb{D}(V_{-J}) \to \mathbb{D}(V_{-j}) \times \mathbb{D}(V_{-j}^{\perp})$ through

$$\boldsymbol{F}_j \coloneqq \overline{F}_j \mathrm{proj}_{V_{-j}}, \tag{B.59}$$

and its compositions by

$$\boldsymbol{F}_{j_1|j_2} = \boldsymbol{F}_{j_1} \circ \cdots \circ \boldsymbol{F}_{j_2}, \tag{B.60}$$

which too is an operator on $\mathbb{D}(V_{-J})$.

Further if we have a measure ν_j on V_{-j} we can form the embedding map

$$\mathrm{embd}_j \nu_j = \nu_j \otimes \bigotimes_{i=j}^{J} \delta_{\{0\}}, \tag{B.61}$$

which we extend to $\mathbb{D}(V_{-J})$ by a convenient abuse of notation

$$\mathrm{proj}_j \nu_J = \mathrm{proj}_j(\nu_J) \otimes \bigotimes_{i=j}^{J} \delta_{\{0\}}. \tag{B.62}$$

Let $B_{-j} : \mathbb{D}(V_{-j}) \to \mathbb{D}(V_{-j})$ be the linear operator which is the inverse of F_{-j}. Now if we extend $B_j : \mathbb{D}(V_{-j}) \to \mathbb{D}(V_{-j})$ to $\mathbb{D}(V_{-j}) \times \mathbb{D}(V_{-j}^{\perp})$ like before through

$$\overline{B}_j \coloneqq B_j \oplus I, \tag{B.63}$$

so the map $\overline{B}_j \mathrm{embd}_j$ is well defined on $\mathbb{D}(V_{-J})$. Now analogously define $\boldsymbol{B}_j$ and its compositions by

$$\boldsymbol{B}_j \coloneqq \overline{B}_j \mathrm{embd}_{V_{-j}} \qquad\qquad \boldsymbol{B}_{j_1|j_2} = \boldsymbol{B}_{j_2} \circ \cdots \circ \boldsymbol{B}_{j_1}. \tag{B.64}$$

In an analogous way, the operator $\boldsymbol{F}_{j_1|j_2}$ is 'upper triangular' and $\boldsymbol{B}_{j_1|j_2}$ is 'lower triangular'. In this way, we are again seeking a lower/upper (LU-) decomposition of the identity on $\mathbb{D}(V_{-J})$. Now we may prove Theorem 3.3.

Theorem 3.3. Let $\{V_{-j}\}_{j=0}^{J}$ be a multi-resolution hierarchy of V_{-J} where $V_{-j} = V_{-j+1} \oplus U_{-j+1}$, and further, let $F_{j,\phi}$, $B_{j,\theta} : \mathbb{D}(V_{-j}) \mapsto \mathbb{D}(V_{-j})$ be such that $B_{j,\theta} F_{j,\phi} = I$ with parameters ϕ and θ. Define $\boldsymbol{F}_{j_1|j_2,\phi} \coloneqq \boldsymbol{F}_{j_1,\phi} \circ \cdots \circ \boldsymbol{F}_{j_2,\phi}$ by $\boldsymbol{F}_{j,\phi} : \mathbb{D}(V_{-j}) \mapsto \mathbb{D}(V_{-j+1})$ where $\boldsymbol{F}_{j,\phi} \coloneqq \mathrm{proj}_{V_{-j+1}} \circ F_{j,\phi}$, and analogously define $\boldsymbol{B}_{j_1|j_2,\theta}$ with $\boldsymbol{B}_{j,\theta} \coloneqq B_{j,\theta} \circ \mathrm{embd}_{V_{-j}}$. Then, the sequence $\{\boldsymbol{B}_{1|j,\theta}(\boldsymbol{F}_{1|J,\phi}\nu_J)\}_{j=0}^{J}$ forms a discrete multi-resolution bridge between $\boldsymbol{F}_{1|J,\phi}\nu_J$ and $\boldsymbol{B}_{1|J,\theta}\boldsymbol{F}_{1|J,\phi}\nu_J$ at times $\{t_j\}_{j=1}^{J}$, and

$$\sum_{j=0}^{J} \mathbb{E}_{X_{t_J} \sim \nu_J} \left\| proj_{U_{-j+1}} X_{t_j} \right\|_2^2 / \left\| \boldsymbol{F}_{j|J,\phi} \right\|_2^2 \leq (\mathcal{W}_2(\boldsymbol{B}_{1|J,\theta}\boldsymbol{F}_{1|J,\phi}\nu_J, \nu_J))^2, \tag{B.65}$$

where $\mathcal{W}_2$ is the Wasserstein-2 metric and $\left\| \boldsymbol{F}_{j|J,\phi} \right\|_2$ is the Lipschitz constant of $\boldsymbol{F}_{j|J,\phi}$.

Proof. All we must show is that successively chaining the projections from Proposition B.2 decomposes like in Proposition B.1. For $X_1, X_2 \sim \nu$, $\mathcal{W}_2(F_j F_{j+1}\nu, P_{-j+2}F_{j-1}P_{-j+1}F_j\nu)$ consider f_j, f_{j-1} as realised paths for our kernel and write

$$\begin{aligned}
&\|f_j f_{j-1} X_1 - \mathrm{proj}_{V_{-j+2}} f_{j-1} \mathrm{proj}_{V_{-j+1}} f_j X_2\|_2^2 \\
&= \|\mathrm{proj}_{V_{-j+1}}(f_j f_{j-1} X_1 - \mathrm{proj}_{V_{-j+2}} f_{j-1} \mathrm{proj}_{V_{-j+1}} f_j X_2)\|_2^2 \\
&\quad + \|\mathrm{proj}_{U_{-j+1}}(f_j f_{j-1} X_1 - \mathrm{proj}_{V_{-j+2}} f_{j-1} \mathrm{proj}_{V_{-j+1}} f_j X_2)\|_2^2
\end{aligned}$$

due to the triangular form and the orthogonality of the multi-resolution basis. Let $\nu_{-j+1} = \text{proj}_{V_{-j+1}}\nu_{-j}$, then as $\text{proj}_{V_{-j+1}}$ commutes with any term equivalent to the identity operator on V_{-j+1}, the first term becomes

$$\|f_{j-1}X_{1,t_{j+1}} - \text{proj}_{V_{-j+2}}f_{j-1}X_{2,t_{j+1}})\|_2^2, \tag{B.66}$$

where $X_{1,t_{j+1}}, X_{2,t_{j+1}} \sim \nu_{-j+1}$. When an optimal coupling is made, this term becomes $\|\text{proj}_{U_{-j+2}}X_{1,t_{j+1}}\|_2^2$. The second term has $\text{proj}_{U_{-j+1}}\text{proj}_{V_{-j+2}}$ nullified, and again commutes where appropriate making this

$$\|\text{proj}_{U_{-j+1}}X_{1,t_{j+1}}\|_2^2. \tag{B.67}$$

We may again use the triangular form to utilise the identify

$$\|\text{proj}_{U_{-j+1}}F_j\|_2^2 \leq \|F_j\|_2^2, \tag{B.68}$$

to define

$$\lambda_{j|J} \coloneqq \text{diag}(\|\text{proj}_{U_{-j+1}}F_{j|J}\|_2^2, \ldots, \|\text{proj}_{U_{-J+1}}F_{J|J}\|_2^2) \tag{B.69}$$

so that

$$\mathcal{W}_2(\lambda_{j|J}^{-1}(F_{j|J}\nu_1), \lambda_{j|J}^{-1}(F_{j|J}\nu_2)) \leq \mathcal{W}_2(\nu_1, \nu_2). \tag{B.70}$$

Piecing the decomposition and scaling together, we yield

$$\mathbb{E}_{\nu_{-j+2}}\|\text{proj}_{U_{-j+2}}X_{1,t_{j+1}}\|_2^2/\|F_{j-2|j}\|_2^2 + \mathbb{E}_{\nu_{-j+1}}\|\text{proj}_{U_{-j+1}}X_{1,t_{j+1}}\|_2^2/\|F_{j-2|j}\|_2^2 \tag{B.71}$$

$$\leq (\mathcal{W}_2(\nu, \mathbf{B}_{j-2|j}\mathbf{F}_{j-2|j}))^2. \tag{B.72}$$

Iterating over j in the fashion given yields the result. Last, measures within

$$\mathcal{U}_{\boldsymbol{BF}} \coloneqq \{\nu_J \,|\, \boldsymbol{F}_{j|J}\gamma_J = \overline{F}_{j|J} \otimes \bigotimes_{i=j}^{J}\delta_{\{0\}}\}, \tag{B.73}$$

are invariant under $\boldsymbol{B}_{J|1}\boldsymbol{F}_{1|J}$, further, $\boldsymbol{B}_{J|1}\boldsymbol{F}_{1|J}$ projects onto this set. To see this, take any measure $\nu_J \in \mathbb{D}(V_{-J})$ and apply $\boldsymbol{F}_{j|J}$. The information in $V_{-j}^{\perp}$ split by $\boldsymbol{P}_j$ is replaced by $\delta_{\{0\}}$ in the backward pass. Thus $\boldsymbol{B}_{J|1}\boldsymbol{F}_{1|J}\boldsymbol{B}_{J|1}\boldsymbol{F}_{1|J} = \boldsymbol{B}_{J|1}\boldsymbol{F}_{1|J}$. □

B.1.4 U-Nets in V_{-J}

Here we show how U-Nets can be seen as only computing the non-truncated components of a multi-resolution diffusion bridge on V_{-J} — the computations are performed in V_{-j} for $j < J$ at various layers. This amounts to showing the embedding presented in Theorem 3.1.

Theorem 3.1. Let $B_j : [t_j, t_{j+1}) \times \mathbb{D}(V_{-j}) \mapsto \mathbb{D}(V_{-j})$ be a linear operator (such as a diffusion transition kernel, see Appendix B.1) for $j < J$ with coefficients $\mu^{(j)}, \sigma^{(j)} : [t_j, t_{j+1}) \times V_{-j} \mapsto V_{-j}$, and define the natural extensions within V_{-J} in bold, i.e. $\boldsymbol{B}_j \coloneqq B_j \oplus \boldsymbol{I}_{V_{-j}^\perp}$. Then the operator $\boldsymbol{B} : [0, T] \times \mathbb{D}(V_{-J}) \mapsto \mathbb{D}(V_{-J})$ and the coefficients $\boldsymbol{\mu}, \boldsymbol{\sigma} : [0, T] \times V_{-J} \mapsto V_{-J}$ given by

$$\boldsymbol{B} \coloneqq \sum_{j=0}^{J} \mathbb{1}_{[t_j, t_{j+1})} \cdot \boldsymbol{B}_j, \quad \boldsymbol{\mu} \coloneqq \sum_{j=0}^{J} \mathbb{1}_{[t_j, t_{j+1})} \cdot \boldsymbol{\mu}^{(j)}, \quad \boldsymbol{\sigma} \coloneqq \sum_{j=0}^{J} \mathbb{1}_{[t_j, t_{j+1})} \cdot \boldsymbol{\sigma}^{(j)},$$

induce a multi-resolution bridge of measures from the dynamics for $t \in [0, T]$ and on the standard basis as $dX_t = \boldsymbol{\mu}_t(X_t)dt + \boldsymbol{\sigma}_t(X_t)dW_t$ (see Appendix B.1.4 for the details of this integration) for $X_t \in V_{-J}$, i.e. a (backward) multi-resolution diffusion process.

Proof. At time $t = 0$ we have that $\text{supp}\nu_0 \subset V_0 = \{0\}$, so $\mathbb{D}(V_0) = \delta_{\{0\}}$. For the s in the first time interval $[t_0, t_1)$ it must be the case $\nu_s = \delta_{\{0\}}$, so $\mu_s^{(j)}, \sigma_s^{(j)} = 0$ and $B_0(s) \equiv I$. The extension is thus $\boldsymbol{B}_0(s) \equiv I$ on V_{-J}. At $t = t_1$, the operator $\boldsymbol{B}_1 \equiv I$ on $V_1^\perp$ grants $\boldsymbol{\mu}_s, \boldsymbol{\sigma}_s = 0$. On V_1, $\boldsymbol{B}_1$ is an operator with domain in V_1, granting $\text{supp}\nu_{t_1} \subset V_1$. For s within the interval (t_1, t_2) we maintain $\text{supp}\nu_s \subset V_1$, and by induction we can continue this for any $s \in [t_j, t_{j+1}) \subsetneq [0, 1]$. Let E_j be a basis of V_{-j}, then as $\boldsymbol{\mu}_s, \boldsymbol{\sigma}_s = 0$ on $V_{-j}^\perp$ the diffusion SDE on $[t_j, t_{j+1}) \times V_{-j}$ given in the basis E_j by

$$dX_t^{(j)} = \mu_t^{(j)}(X_t)dt + \sigma_t^{(j)}(X_t)dW_t \tag{B.74}$$

embeds into an SDE on V_{-J} with basis E_J by

$$dX_t = \boldsymbol{\mu}_t(X_t)dt + \boldsymbol{\sigma}_t(X_t)dW_t, \tag{B.75}$$

which maintains $X_t \in V_{-j}$ as $\boldsymbol{\sigma}_t \equiv 0$ on the complement. □

In practice, we will compute the sample paths made from Equation B.74, but we can in theory think of this as Equation B.75. The U-Net sequential truncation of spaces, then sequential inclusion of these spaces is what forms the multi-dimensional bridge with our sampling models.

B.1.5 Forward Euler Diffusion Approximations

Here we show that the backward cell structure of state-of-the-art HVAEs approximates an SDE evolution within each resolution.

Theorem 3.4. Let $t_J \coloneqq T \in (0,1)$ and consider (the p_θ backward pass) $\boldsymbol{B}_{\theta,1|J} : \mathbb{D}(V_{-J}) \mapsto \mathbb{D}(V_0)$ given in multi-resolution Markov process in the standard basis:

$$dZ_t = (\overleftarrow{\mu}_{1,t}(Z_t) + \overleftarrow{\mu}_{2,t}(Z_t))dt + \overleftarrow{\sigma}_t(Z_t)dW_t, \tag{B.76}$$

where $\text{proj}_{U_{-j+1}} Z_{t_j} = 0$, $\|Z_t\|_2 > \|Z_s\|_2$ with $0 \leq s < t \leq T$ and for a measure $\nu_J \in \mathbb{D}(V_{-J})$ we have $Z_0 \sim \boldsymbol{F}_{\phi,J|1}\nu_J$. Then, VDVAEs approximates this process, and its residual cells are a type of two-step forward Euler discretisation of this Stochastic Differential Equation (SDE).

Proof. The evolution

$$dZ_t = (\overleftarrow{\mu}_{1,t}(Z_t) + \overleftarrow{\mu}_{2,t}(Z_t))dt + \overleftarrow{\sigma}_t(Z_t)dW_t, \tag{B.77}$$

subject to $Z_0 = 0, \|Z_t\|_2 > \|Z_s\|_2$ and X_T, $Z_0 \sim \boldsymbol{F}_{\phi,J|1}\nu_J$. By Theorem 3.3 we know $\boldsymbol{F}_{\phi,J|1}\nu_J$ enforces the form

$$\text{proj}_{V_1}\overline{F}_{\phi,J|1}\nu_J \otimes \bigotimes_{j=1}^{J-1} \delta_{\{0\}} \tag{B.78}$$

when ϕ is trained with a reconstruction loss. By Theorem 3.5, the full cost used imposes $\text{proj}_{V_1}\overline{F}_{\phi,J|1}\nu_J = \delta_{\{0\}}$, further, VDVAE initialises $Z_0 = \delta_{\{0\}}$. This enforces $Z_0 = 0$ as $Z_0 \sim \delta_{\{0\}}$. For the backward SDE, consider the splitting

$$dZ_t^{(1)} = \overleftarrow{\mu}_{1,t}(Z_t^{(1)})dt + \overleftarrow{\sigma}_t(Z_t^{(1)})dW_t, \qquad dZ_t^{(2)} = \overleftarrow{\mu}_{2,t}(Z_t^{(2)})dt, \tag{B.79}$$

where $dZ_t = dZ_t^{(1)} + dZ_t^{(2)}$ when $Z_t = Z_t^{(1)} = Z_t^{(2)}$. For the split SDE make the forward-Euler discretisation

$$Z_{i+1}^{(1)} = Z_i^{(1)} + \int_i^{i+1} \overleftarrow{\mu}_{1,t}(Z_t^{(1)})dt + \int_i^{i+1} \overleftarrow{\sigma}_t(Z_i^{(1)})dW_t \approx Z_i^{(1)} + \overleftarrow{\mu}_{1,i}(Z_i^{(1)}) + \overleftarrow{\sigma}_i(Z_i^{(1)})(W_1). \tag{B.80}$$

Now the second deterministic component can also be approximated with a forward-Euler discretisation

$$Z_{i+1}^{(2)} = Z_i^{(2)} + \int_i^{i+1} \overleftarrow{\mu}_{2,t}(Z_t^{(2)})dt \approx Z_i^{(2)} + \overleftarrow{\mu}_{2,i}(Z_i^{(2)}). \tag{B.81}$$

As $Z_0 = 0$, we need only show the update at a time i, so assume we have Z_i. First we update in the SDE step, so make the assignment and update

$$Z_i^{(1)} \leftarrow Z_i, \qquad Z_{i+1}^{(1)} = Z_i^{(1)} + \overleftarrow{\mu}_{i,1}(Z_i^{(1)}) + \overleftarrow{\sigma}_i(Z_i^{(1)})\Delta W_1. \tag{B.82}$$

Now assign $Z_i^{(2)} \leftarrow Z_{i+1}^{(1)}$ so we may update in the mean direction with

$$Z_{i+1}^{(2)} = Z_i^{(2)} + \overleftarrow{\mu}_{i,2}(Z_i^{(2)}), \tag{B.83}$$

with the total update $Z_{i+1} \leftarrow Z_{i+1}^{(2)}$. This gives the cell update for NVAE in Figure B.1. To help enforce the growth $\|Z_t\|_2 > \|Z_s\|_2$, VDVAE splits $Z_i^{(1)} = Z_i + Z_{i,+}$ where $Z_{i,+}$ increases the norm of the latent process Z_t. This connection and the associated update are illustrated in Figure B.1 [left]. Note here that if no residual connection through the cell was used (just the re-parameterisation trick in a VAE), the model would degenerate to a hierarchical VAE with a single step.

Remark B.1. To simplify the stepping notation in the HVAE backward cells (Figures 3.4 and B.1), we use $Z_i^{(1)} = Z_i + \overleftarrow{\mu}_{1,i}(Z_i) + \overleftarrow{\sigma}_i(Z_i)(W_1)$ and $Z_i^{(2)} = Z_i^{(1)} + \overleftarrow{\mu}_{i,2}(Z_i^{(1)})$ so that the index i refers to all computations of the i^{th} backward cell.

□

B.1.6 Time-homogenuous model

Recall VDVAE has the continuous time analogue

$$dZ_t = (\overleftarrow{\mu}_{1,t}(Z_t) + \overleftarrow{\mu}_{2,t}(Z_t))dt + \overleftarrow{\sigma}_t(Z_t)dW_t, \tag{B.84}$$

where $Z_0 = 0,\ \|Z_t\|_2 > \|Z_s\|_2$ with $0 \leq s < t \leq T$ and for a measure $\nu_J \in \mathbb{D}(V_{-J})$. Due to Theorem 3.5, we know that the initial condition of VDVAE's U-Net is the point mass $\delta_{\{0\}}$. As the backwards pass flows from zero to positive valued functions, this direction is increasing and the equation is stiff with few layers. The distance progression from zero is our proxy for time, and we can use its 'position' to measure this. Thus, the coefficients $\overleftarrow{\mu}_{t,1}$, $\overleftarrow{\mu}_{t,2}$, $\overleftarrow{\sigma}_t$ need not have a time dependence as this is already encoded in the norm of the Z_t processes. Thus, the time-homogeneous model postulated in the main text is:

$$dZ_t = (\overleftarrow{\mu}_1(Z_t) + \overleftarrow{\mu}_2(Z_t))dt + \overleftarrow{\sigma}(Z_t)dW_t, \tag{B.85}$$

$$Z_0 = 0,\ \|Z_t\|_2 > \|Z_s\|_2. \tag{B.86}$$

In practice, the loss of time dependence in the components corresponds to weight sharing the parameters across time, as explored in the experimental section. Weight sharing, or a time-homogeneous model, is common for score based diffusion models [92, 206], and due to our identification we are able to utilise this for HVAEs.

B.1.7 HVAE Sampling

Here we use our framework to comment on the sampling distribution imposed by the U-Net within VDVAE.

Theorem 3.5. Consider the SDE in Eq. (3.4), trained through the ELBO in Eq. 1.2. Let $\tilde{\nu}_J$ denote the data measure and $\nu_0 = \delta_{\{0\}}$ be the initial multi-resolution bridge measure imposed by VDVAEs. If $q_{\phi,j}$ and $p_{\theta,j}$ are the densities of $B_{\phi,1|j}\boldsymbol{F}_{J|1}\tilde{\nu}_J$ and $B_{\theta,1|j}\nu_0$ respectively, then a VDVAE optimises the boundary condition $\min_{\theta,\phi} KL(q_{\phi,0,1}||q_{\phi,0}p_{\theta,1})$, where a double index indicates the joint distribution.

Proof. We need to only show two things. First, due to Theorems 3.3 and 3.4, we know that the architecture imposes

$$\text{proj}_{V_1}\overline{F}_{\phi,J|1}\nu_J \otimes \bigotimes_{j=1}^{J-1}\delta_{\{0\}}, \tag{B.87}$$

so we must analyse how $\text{proj}_{V_1}\overline{F}_{\phi,J|1}\nu_J$ is trained. Second, we use Theorem 3.4 to view the discretised version of the continuous problem, and identify the error in the two-step forward Euler splitting.

On the first point, VDVAE uses an ELBO reconstruction with a KL divergence between the backwards pass of the data $\overline{B}_{\phi,J|1}\overline{F}_{\phi,J|1}\tilde{\nu}_J$ (the 'q_ϕ-distribution'), and the backwards pass of the model imposed by the U-Net $\overline{B}_{\phi,J|1}\nu_0$ (the 'p_θ distribution'). As Z_0 is zero initialised, we know $\nu_0 = \delta_{\{0\}}$. We need to show the cost function used imposes this initialisation on $\overline{B}_{\phi,1|0}\overline{F}_{\phi,J|0}\tilde{\nu}_J$. Let $X_T \sim \overline{F}_{\phi,J|0}\tilde{\nu}_J$, call the distribution of this $q_{\phi,0}$. We also use $Z_1 \sim q_{\phi,1}$ for a sample from $\overline{B}_{\phi,1|0}\overline{F}_{\phi,J|0}\tilde{\nu}_J$ and $Z_1 \sim p_{\theta,1}$ for a sample from $\overline{B}_{\phi,1|0}\nu_0$. For a realisation x of X_T, VDVAE computes

$$KL(q_{\phi,1|0}(\,\cdot\,|X_T = x)||p_{\theta,1|0}(\,\cdot\,|Z_0 = 0)) = KL(q_{\phi,1|0}(\,\cdot\,|X_T = x)||p_{\theta,1}(\,\cdot\,)), \tag{B.88}$$

which in training is weighted by each datum, so the total cost in this term is

$$\int KL(q_{\phi,1|0}(\,\cdot\,|X_T = x)||p_{\theta,1}(\,\cdot\,))q_{\phi,0}(X_T = x)dx. \tag{B.89}$$

But this is equal to,

$$\int\int \log\left(\frac{q_{\phi,1|0}(Z_1 = z|X_T = x)}{p_{\theta,1}(Z_1 = z_1)}\right) q_{\phi,1|0}(Z_1 = z|X_T = x)q_{\phi,0}(X_T = x)dzdx \tag{B.90}$$

$$= \int\int \log\left(\frac{q_{\phi,0,1}(Z_1 = z, X_T = x)}{p_{\theta,1}(Z_1 = z_1)q_{\phi,0}(X_T = x)}\right) q_{\phi,0,1}(Z_1 = z, X_T = x)dzdx \tag{B.91}$$

$$= KL(q_{\phi,0,1}(Z_1, X_T)||p_{\theta,1}(Z_1)q_{\phi,0}(X_T)) = KL(q_{\phi,0,1}||p_{\theta,1}q_{\phi,0}). \tag{B.92}$$

The distribution of $p_{\theta,1}$ is Gaussian as a one time step diffusion evolution from the initial point mass $\nu_0 = \delta_{\{0\}}$.

□

Theorem 3.1 states that the choice of the initial latent variable in VDVAE imposes a boundary condition on the continuous SDE formulation. Further, this boundary condition is enforced into the final output X_T of the encoder within VDVAE.

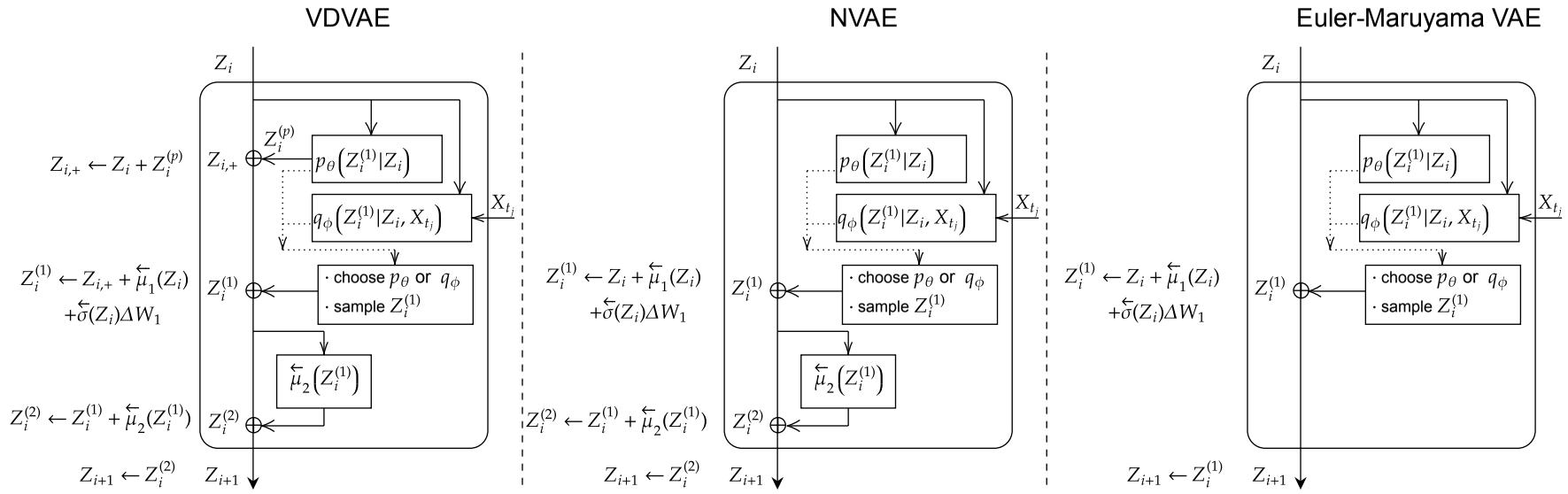


Figure B.1: HVAE top-down cells are resembling two-step forward Euler discretisations of a continuous-time diffusion process in Eq. 3.4. We here provide the residual cell structures of [left] VDVAE [34], [middle] NVAE [221] and [right] a Euler-Maruyama VAE. Either q_ϕ (conditional) or p_θ (unconditional) are used in the sampling step (indicated by the dotted lines) during training and generation, respectively. X_{t_j} is an input from the in effect non-stochastic bottom-up pass, Y_i is the input from the previous, and Y_{i+1} the output to the next residual cell. $\oplus$ indicates element-wise addition.

B.2 Background

B.2.1 Multi-Resolution Hierarchy and thought experiment

Let $\mathbb{X} \subset \mathbb{R}^m$ be compact and $L^2(\mathbb{X})$ be the space of square-integrable functions over this set. We are interested in decomposing $L^2(\mathbb{X})$ across multiple resolutions.

Definition 3.1 (abbreviated). A *multi-resolution hierarchy* is one of the form

$$\cdots \subset V_1 \subset V_0 \subset V_{-1} \subset \cdots \tag{B.93}$$

$$\overline{\bigcup_{j\in\mathbb{Z}} V_{-j}} = L^2(\mathbb{R}^m) \tag{B.94}$$

$$\bigcap_{j\in\mathbb{Z}} V_{-j} = \{0\} \tag{B.95}$$

$$f(\cdot) \in V_{-j} \iff f(2^j \cdot) \in V_0 \tag{B.96}$$

$$f(\cdot) \in V_0 \iff f(\cdot - n) \in V_0, \text{ for } n \in \mathbb{Z}. \tag{B.97}$$

Each V_{-j} is a finite truncation of $L^2(\mathbb{X})$. What we are interested in is to consider a function $f \in L^2(\mathbb{X})$ and finding a finite dimensional approximation in V_{-J}, say, for $J > 0$. Further, for gray-scale images, $\mathbb{X} = [0,1]^2$, the space of pixel-represented images. To simplify notation, we just consider $\mathbb{X} = [0,1]$ for the examples below, but we can extend this to gray-scale images, and to colour images with a Cartesian product.

The 'pixel' multi-resolution hierarchy is given by the collection of sub-spaces

$$V_{-j} = \{f \in L^2([0,1]) \mid f|_{[2^{-j}\cdot k, 2^{-j}\cdot(k+1))} = c_k,\, k \in \{0,\ldots,2^j-1\},\, c_k \in \mathbb{R}\}. \tag{B.98}$$

It can be readily checked that these sub-spaces obey the assumptions of Definition 3.1. An example image projected into such sub-spaces, obtained from a discrete Haar wavelet transform, is illustrated in Fig. B.2. We call it the pixel space of functions as elements of this set are piece-wise constant on dyadically split intervals of resolution 2^j, i.e a pixelated image. For each V_{-j} there is an obvious basis of size

2^j where we store the coefficients $(c_0, c_1, \ldots, c_{2^j-1}) \in \mathbb{R}^{2^j}$. The set of basis vectors for it is the *standard basis* $\{e_i\}_{i=0}^{2^j-1}$ which are 0 for all co-ordinates except for the $i^{th}+1$ entry which is 1. This basis is not natural to the multi-resolution structure of V_{-j}. This is because all the basis functions change when we project down to V_{-j+1}. We want to use the multi-resolution structure to create a basis which naturally relates V_{-j}, V_{-j+1}, and any other sub-space. To do this consider $V_{-j} \cap V_{-j+1}^{\perp} \subset V_{-j}$. Define this orthogonal compliment to be $U_{-j+1} \coloneqq V_{-j+1}^{\perp}$, then see $V_{-j} = V_{-j+1} \oplus U_{-j+1}$. Doing this recursively finds $V_{-j} = V_0 \oplus \bigoplus_{i=0}^{-j+1} U_i$, and taking the limit

$$L^2(\mathbb{X}) = \bigoplus_{i=0}^{-\infty} U_i \oplus V_0. \tag{B.99}$$

Each of the sub-spaces $\{U_{-j}\}_{j=0}^{\infty}$ are mutually orthogonal as each $V_{-j} \perp U_{-j}$. Now suppose we had a basis set $\mathbf{\Psi}_j$ for each U_{-j} and $\mathbf{\Phi}_0$ for V_0. As these spaces are orthogonal, so are the basis sets to each other, too. We can make a basis for V_{-j} with $\text{span}(\mathbf{\Phi}_0, \mathbf{\Psi}_0, \cdots, \mathbf{\Psi}_{-j+1})$. For the above examples, V_0 needs only a single basis function $\phi_{0,k} = \mathbb{1}_{[k,k+1)}$, further if $\psi = \sqrt{2}(\mathbb{1}_{[0,1/2)} - \mathbb{1}_{[1/2,1)})$, then given the functions $x \mapsto \psi_{j,k}(x) \coloneqq 2^{j/2} \cdot \psi(2^{-j}(x-k))$ we have $\{\psi_{j,k}\}$ is a basis for V_{-j}.

B.2.2 U-Net

In practice, a U-Net [197] is a neural network structure which consists of a forward pass (encoder) and backward pass (decoder), wherein layers in the forward pass have some form of dimension reduction, and layers in the backward pass have some form of dimension embedding. Furthermore, there are 'skip connection' between corresponding layers on the same level of the forward and backward pass.

We now formalise this notion, referring to an illustration of a U-Net in Fig. B.3. In black, we label the latent spaces to be V_{-j} for all j, where the original data is in V_{-J} and the U-Net 'bend' (bottleneck) occurs at V_0. We use $f_{j,\theta}$ to be the forward component, or encoder, of the U-Net, and similarly $b_{j,\theta}$ as the backward component or decoder, operating on the latent space V_{-j}. P_{-j+1} refers to the dimension

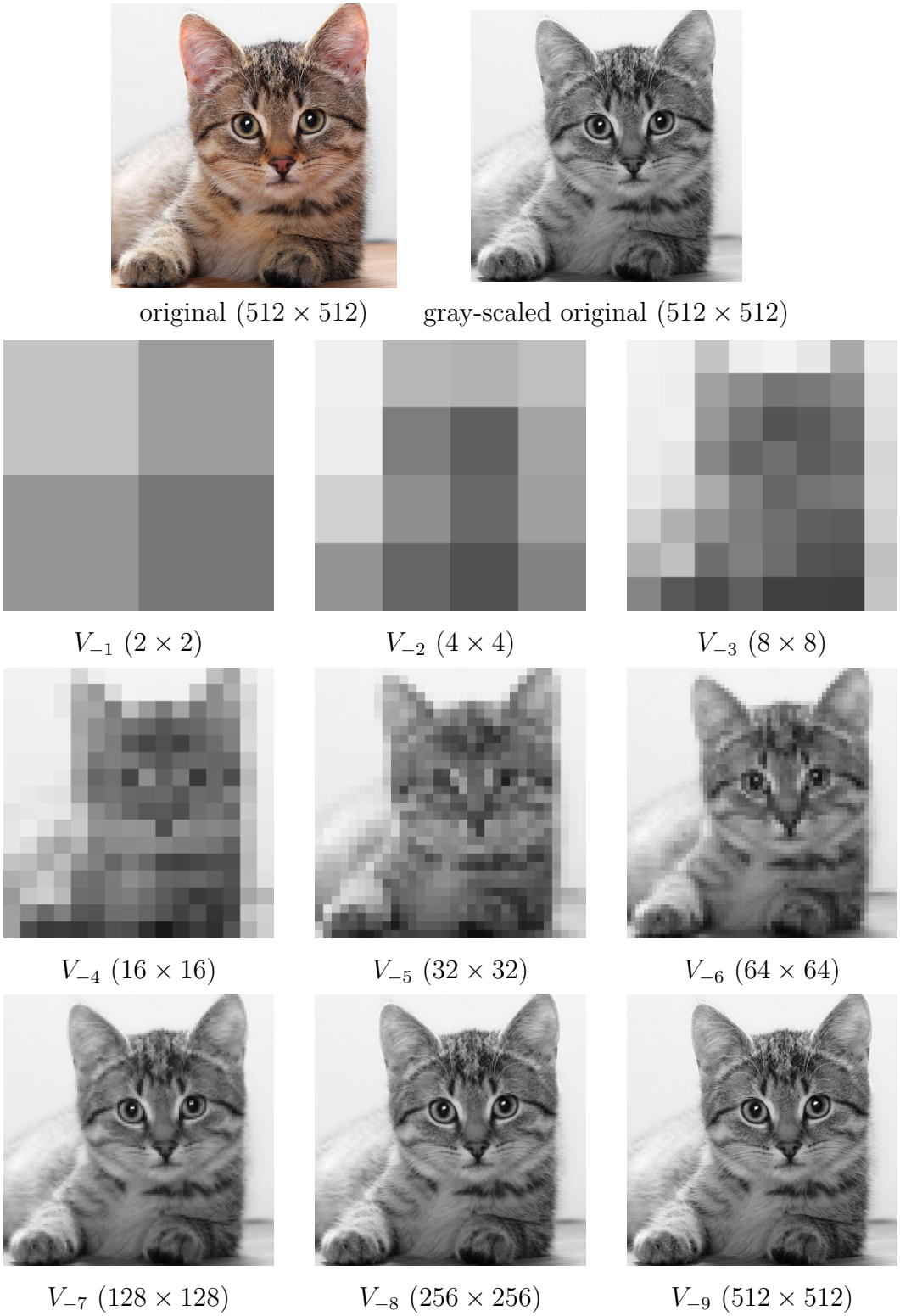


Figure B.2: The thought experiment discussed in §3.2. The original colour image [top-left], its gray-scale version [top-right], and its Haar wavelet projections to the approximation spaces V_{-j} for $j \in \{1, \ldots, 9\}$.

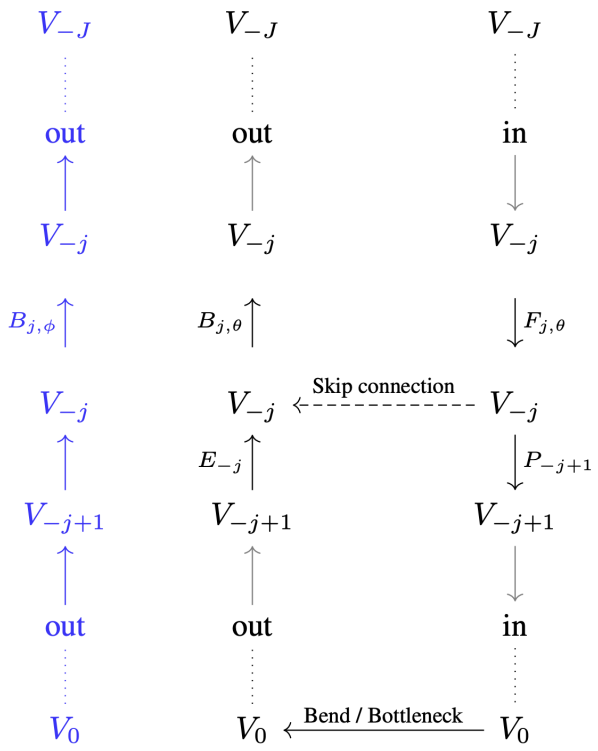


Figure B.3: The repeated structure in a U-Net, where V_{-j+1} is a lower dimensional latent space compared to V_{-j}. $f_{j,\theta}, b_{j,\theta}$ are in practice typically parameterised by neural networks (e.g. convolutional neural networks); P_{-j+1} is a dimension reduction operation (e.g. average pooling) to a lower-dimensional latent space; and, E_{-j} is a dimension embedding operation (e.g. deterministic interpolation) to a higher-dimensional latent space. This structure is repeated to achieve a desired dimension of the latent space at the U-Net bottleneck.

reduction operation between latent space V_{-j} and V_{-j+1}, and E_{-j} refers to its corresponding dimension embedding operation between latent spaces V_{-j+1} and V_{-j}. A standard dimension reduction operation in practice is to take P_{-j+1} as average pooling, reducing the resolution of an image. Similarly, the embedding step may be some form of deterministic interpolation of the image to a higher resolution. We note that the skip connection in Fig. B.3 occur before the dimension reduction step, in this sense, lossless information is fed from the image of $f_{j,\theta}$ into the domain of $b_{j,\theta}$.

In blue, we show another backward process $b_{j,\phi}$ that is often present in U-Net architectures for generative models. This second backward process is used for unconditional sampling. In the context of HVAEs, we may refer to it as the (hierarchical) prior (and likelihood model). It is trained to match its counterpart

in black, without any information from the forward process. In HVAEs, this is enforced by a KL-divergence between distributions on the latent spaces V_{-j}. The goal of either backward process is as follows:

1. $b_{j,\theta}$ must be able to reconstruct the data from $f_{j,\theta}$, and in this sense it is reasonable to require $b_{j,\theta}f_{j,\theta} = I$;
2. $b_{j,\phi}$ must connect the data to a known sampling distribution.

The second backward process can be absent when the backward process $b_{j,\theta}$ is imposed to be the inverse of $f_{j,\theta}$, such as in Normalising Flow based models, or reversible score-based diffusion models. In this case the invertibility is assured, and the boundary condition that the encoder connects to a sampling distribution must be enforced. For the purposes of our study, we will assume that in the absence of dimension reduction, the decoder is constrained to be an inverse of the encoder. This is a reasonable assumption: for instance, in HVAEs near perfect data reconstructions are readily achieved.

For variational autoencoders, the encoder and decoder are not necessarily deterministic and involve resampling. To encapsulate this, we will work with the data as a measure and have $F_{\theta,j}$ and $B_{\theta,j}$ as the corresponding kernels imposed by $f_{j,\theta}$ and $b_{j,\theta}$, respectively.

With all of these considerations in mind, for the purposes of our framework we provide a definition of an idealised U-Net which is an approximate encapsulation of all models using a U-Net architecture.

Definition B.1. (Idealised U-Net for generative modelling)
For each $j \in \{0, \dots, J\}$, let $F_{j,\theta}, B_{j,\theta} : \mathbb{D}(V_{-j}) \mapsto \mathbb{D}(V_{-j})$ such that $B_{j,\theta}F_{j,\theta} \equiv I_{V_{-j}}$. A U-Net with (average pooling) dimension reduction P_{-j+1} and dimension embedding E_{-j} is the operator $\mathbf{U} : \mathbb{D}(V_{-J}) \mapsto \mathbb{D}(V_{-J})$ given by

$$\mathbf{U} \coloneqq B_{J,\theta}E_{-J} \circ \cdots \circ B_{1,\theta}E_{-1} \circ P_0F_{1,\theta} \circ \cdots \circ P_{-J+1}F_{J,\theta}, \qquad B_jF_j \equiv I. \tag{B.100}$$

Remark B.2. The condition $B_{j,\theta}F_{j,\theta} \equiv I_{V_{-j}}$ in our idealised U-Net (for unconditional sampling here) is either imposed directly (reversible flow based model), or approximated via skip connections. For instance, in our HVAE case, we have both a U-Net without skip connections (the p distribution) and a U-Net with skip connections (the q distribution). The U-Net related to the q distribution learns how to reconstruct our data from the reconstruction term in the ELBO cost function. The U-Net related to the p distribution learns to mimic the q distribution via the KL term in the ELBO of the HVAE, whose decoder is trained to invert its encoder, i.e. $B_{j,\phi}F_{j,\phi} \equiv I_{V_{-j}}$, but the p U-Net lacks skip connections. Thus, in the HVAE context, we are analysing U-Nets which must simultaneously reconstruct our data and lose their reliance on their skip connections due to the condition that the q U-Net must be approximately equal to the p U-Net.

B.2.3 Sampling of Time Steps in HVAEs

Monte Carlo sampling of time steps in ELBO of HVAEs.

We here provide one additional theoretical result. We show that the ELBO of an HVAE can be written as an expected value over uniformly distributed time steps.

Previous work [120] [92] (Eq. (13), respectively) showed that the diffusion loss term $\mathcal{L}_T(\mathbf{x})$ in the ELBO of discrete-time diffusion models can be written as

$$\mathcal{L}_T(\mathbf{x}) = \frac{T}{2}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I}),i\sim U\{1,T\}}\left[(\mathrm{SNR}(s) - \mathrm{SNR}(t))\left\|\mathbf{x} - \hat{\mathbf{x}}_\theta\left(\mathbf{z}_t; t\right)\right\|_2^2\right] \tag{B.101}$$

which allows maximizing the variational lower-bound via a Monte Carlo estimator of Eq. B.101, sampling time steps.

Inspired by this result for diffusion models, we provide a similar form of the ELBO for an HVAE with factorisation as in Eqs. (1.3)-(1.4) (and the graphical model

in Fig. 1.3). An HVAE's ELBO can be written as

$$\log p(\mathbf{x}) \geq \mathbb{E}_{\vec{\mathbf{z}}\sim q(\vec{\mathbf{z}}|\mathbf{x})}\left[\log p(\mathbf{x}|\vec{\mathbf{z}})\right] - L\,\mathbb{E}_{l\sim\mathrm{Unif}(1,L)}\left[\mathbb{E}_{\vec{\mathbf{z}}\sim q(\vec{\mathbf{z}}|\mathbf{x})}\log\frac{q(\mathbf{z}_l|\mathbf{z}_{>l},\mathbf{x})}{p(\mathbf{z}_l|\mathbf{z}_{>l})}\right].$$

Proof.

$$\begin{aligned}
\log p(\mathbf{x}) &\geq \mathbb{E}_{\vec{\mathbf{z}}\sim q(\vec{\mathbf{z}}|\mathbf{x})}\left[\log p(\mathbf{x}|\vec{\mathbf{z}})\right] - \mathrm{KL}\left[q(\vec{\mathbf{z}}|\mathbf{x})||p(\vec{\mathbf{z}})\right] \\
&= \mathbb{E}_{\vec{\mathbf{z}}\sim q(\vec{\mathbf{z}}|\mathbf{x})}\left[\log p(\mathbf{x}|\vec{\mathbf{z}})\right] - \int \mathrm{d}\vec{\mathbf{z}} q(\vec{\mathbf{z}}|\mathbf{x}) \log\left[\frac{\prod_{l=1}^{L} q(\mathbf{z}_l|\mathbf{z}_{>l},\mathbf{x})}{\prod_{l=1}^{L} p(\mathbf{z}_l|\mathbf{z}_{>l})}\right] \\
&= \mathbb{E}_{\vec{\mathbf{z}}\sim q(\vec{\mathbf{z}}|\mathbf{x})}\left[\log p(\mathbf{x}|\vec{\mathbf{z}})\right] - \sum_{l=1}^{L}\int \mathrm{d}\vec{\mathbf{z}} q(\vec{\mathbf{z}}|\mathbf{x}) \log\left[\frac{q(\mathbf{z}_l|\mathbf{z}_{>l},\mathbf{x})}{p(\mathbf{z}_l|\mathbf{z}_{>l})}\right] \\
&= \mathbb{E}_{\vec{\mathbf{z}}\sim q(\vec{\mathbf{z}}|\mathbf{x})}\left[\log p(\mathbf{x}|\vec{\mathbf{z}})\right] - L\,\mathbb{E}_{l\sim\mathrm{Unif}(1,L)}\left[\mathbb{E}_{\vec{\mathbf{z}}\sim q(\vec{\mathbf{z}}|\mathbf{x})}\log\frac{q(\mathbf{z}_l|\mathbf{z}_{>l},\mathbf{x})}{p(\mathbf{z}_l|\mathbf{z}_{>l})}\right].
\end{aligned}$$

□

This allows reducing the computational and memory costs of the KL-terms in the loss and depends on how many Monte Carlo samples are drawn. However, in contrast to diffusion models, all intermediate stochastic layers (up to the top-most and bottom-most layer chosen when sampling time steps in the recognition and generative model, respectively) still need to be computed as each latent variable's distribution depends on all previous ones.

B.3 Code, computational resources, existing assets used

Code. We provide our PyTorch code base at https://github.com/FabianFalck/unet-vdvae. Our implementation is based on, modifies and extends the official implementation of VDVAE [34]. Below, we highlight key contributions:

- We implemented weight-sharing of individual ResNet blocks for a certain number of repetitions.

- We implemented the datasets and the preprocessing of MNIST and CelebA, which were previously not used with VDVAE.
- We implemented the option of synchronous and asynchronous processing in time (see Appendix B.7.4).
- We implemented Fourier features with hyperparameters choosing their frequencies following VDM [120]. One can concatenate them at three different locations as options.
- We simplified the multi-GPU implementation.
- We implemented an option to convert the VDVAE cell into a non-residual cell (see Appendix B.7.4).
- We implemented logging of various metrics and plots with weight&biases.
- We implemented gradient checkpointing [33] as an option in the decoder of VDVAE where the bulk of the computation occurs. We provide two implementations of gradient checkpointing, one based on the official PyTorch implementation which is unfortunately slow when using multiple GPUs, and a prototype for a custom implementation based on https://github.com/csrhddlam/pytorch-checkpoint.

The `README.md` contains instructions on installation, downloading the required datasets, the setup of weights&biases, and how to reproduce our main results.

Computational resources. For the majority of time during this project, we used two compute clusters: The first cluster is a Microsoft Azure server with two Nvidia Tesla K80 graphic cards with 11GB of GPU memory each, which we had exclusive access to. The second cluster is an internal cluster with 12 Nvidia GeForce GTX 1080 graphic cards and 10GB of GPU memory each, shared with a large number of users. In the late stages of the project, in particular to perform runs on ImageNet32, ImageNet64 and CelebA, we used a large-scale compute

cluster with A100 graphic cards with 40GB of GPU memory each. We refer to the acknowledgements section for further details.

In the following, we provide a rough estimate of the total compute required to reproduce our main experiments. Compute time until convergence scales with the depth of the HVAEs. For the shallower HVAEs in our small-scale experiments in §B.7.1, training times range from several days to a week. For our larger-scale experiments on MNIST and CIFAR10, training times range between 1 to 3 weeks. For our deepest runs on ImageNet32 and CelebA, training times range between 2.5 to 4 weeks.

For orientation, in Table B.1, we provide an estimate of the training times of our large-scale runs in Table 3.1. We note that these runs have been computed on different hardware, i.e. the training times are only to some degree comparable, yet give an indication.

Table B.1: A large-scale study of parameter efficiency in HVAEs. For all our runs in Table 3.1, we report their stochastic depth and estimated training time.

	Method	Depth	Training time
MNIST (28×28)			
	`WS-VDVAE` (ours)	57	$\approx$ 5 days
	`VDVAE`* (ours)	43	$\approx$ 5 days
CIFAR10 (32×32)			
	`WS-VDVAE` (ours)	268	$\approx$ 18 days
	`WS-VDVAE` (ours)	105	$\approx$ 13 days
	`VDVAE`* (ours)	43	$\approx$ 9 days
ImageNet (32×32)			
	`WS-VDVAE` (ours)	169	$\approx$ 20 days
	`WS-VDVAE` (ours)	235	$\approx$ 24 days
	`VDVAE`* (ours)	78	$\approx$ 16 days
CelebA (64×64)			
	`WS-VDVAE` (ours)	125	$\approx$ 27 days
	`VDVAE`* (ours)	75	$\approx$ 21 days

Existing assets used. In the experiments, our work directly builds on top of the official implementation of VDVAE [34] (MIT License). We use the

datasets reported in Appendix B.4. In our implementation, we make use of the following existing assets and list them together with their licenses: PyTorch [180], highlighting the torchvision package for image benchmark datasets, and the gradient checkpointing implementation (custom license), Numpy [85] (BSD 3-Clause License) Weights&Biases [19] (MIT License), Apex [174] (BSD 3-Clause "New" or "Revised" License), Pickle [225] (license not available), Matplotlib [99] (PSF License), ImageIO [123] (BSD 2-Clause "Simplified" License), MPI4Py [46] (BSD 2-Clause "Simplified" License), Scikit-learn [181] (BSD 3-Clause License), and Pillow [220] (custom license).

B.4 Datasets

In our experiments, we make use of the following datasets: MNIST [133], CIFAR10 [127], ImageNet32 [36, 55], ImageNet64 [36, 55], and CelebA [145]. We briefly discuss these datasets, focussing on their preprocessing, data splits, data consent and commenting on potential personally identifiable information or offensive content in the data. We refer to the training set as images used during optimisation, the validation set as images used to guide training (e.g. to compute evaluation metrics during training) but not used for optimisation directly, and the test set as images not looked at during training and only to compute performance of completed runs. For all datasets, we fix the training-validation-test split over different runs, and we scale images to be approximately centred and having a standard deviation of one based on statistics computed on the respective training set. If not stated otherwise, we use a modified version of the implementation of these datasets in [34].

MNIST. The MNIST dataset [133] contains gray-scale handwritten images of 10 digit classes ('0' to '9') with resolution 28×28. It contains $60,000$ training and $10,000$ test images, respectively. From the training images, we use $55,000$ images as the training set and 5000 images as the validation set. We use all $10,000$ test images as the testing set. We build on top of the implementation provided

in NVAE [221] (`https://github.com/NVlabs/NVAE/blob/master/datasets.py`), which itself uses torchvision [180], and dynamically binarize the images, meaning that pixel values are binary, as drawn from a Bernoulli distribution with the probabilities given by the scaled gray-scale values in $[0, 1]$. Furthermore, we pad each image with zeros so to obtain the resolution 32×32.

The dataset is highly standardised and cropped to individual digits so that offensive content or personally identifiable information can be excluded. As the original NIST database from which MNIST was curated is no longer available, we cannot comment on whether consent was obtained from the subjects writing and providing these digits [67].

CIFAR10. The CIFAR10 dataset [127] contains coloured images from 10 classes ('airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck') with resolution 32×32. It contains $50,000$ training and $10,000$ test images, respectively. We split the training images into $45,000$ images in the training set and 5000 images in the validation set, and use all $10,000$ test images as the test set.

CIFAR10 was constructed from the so-called 80 million tiny images dataset by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton [128]. On the official website of the 80 million tiny images dataset, the authors state that this larger dataset was officially withdrawn by the authors on June 29th, 2020 due to offensive images being identified in it [186]. The authors of the 80 million tiny images dataset do not comment on whether CIFAR10, which is a subset of this dataset, likewise contains these offensive images or is unaffected. [127] states that the images in the 80 million tiny images dataset were retrieved by searching the web for specific nouns. The authors provide no information to which degree consent was obtained from the people who own these images.

ImageNet32. The ImageNet32 dataset, a downsampled version of the ImageNet database [36, 55], contains $1,281,167$ training and $50,000$ test images from $10,000$

classes with resolution 32×32. From the training images, $5,000$ images as the validation set and the remaining $1,276,167$ as the training set, and further use all $50,000$ test images as the test set.

ImageNet is a human curated collection of images downloaded from the web via search engines. While ImageNet used Amazon Mechanical Turk to lable the images, we were unable to find information on processes which ensured no personally identifiable or offensive content was contained in the images, which is somewhat likely given the "in-the-wild" nature of the dataset. The ImageNet website states that the copyright of the images does not belong to authors of ImageNet.

ImageNet64. The ImageNet64 dataset, a second downsampled version of the ImageNet database [36, 55], likewise contains $1,281,167$ training and $50,000$ validation images with resolution 64×64. We use the same data splits as for ImageNet32. Refer to the above paragraph on ImageNet32 for discussion of personally identifiable information, offensive content and consent.

CelebA. The CelebA dataset [145] contains $162,770$ training, $19,867$ validation and $19,962$ test images with resolution 64×64 which we directly use as our training, validation and test set, respectively. Our implementation is a modified version of the one provided in NVAE [221] (`https://github.com/NVlabs/NVAE/blob/master/datasets.py`).

CelebA images are "obtained from the Internet". The authors state that these images are not the property of the authors of associated institutions [146]. As this dataset shows the faces of humans, these images are personally identifiable. We were unable to identify a process by which consent for using these images was obtained, or how potential offensive content was prevented.

B.5 Potential negative societal impacts

Our work provides mainly theoretical and methodological contributions to U-Nets and HVAEs, and we hence see no direct negative societal impacts. Since U-Nets are widely used in applications, our theoretical results and any future work derived from them may downstream improve such applications, and thus also enhance their performance in malicious uses. In particular, U-Nets are widely used in generative modelling, and here, our work may have an effect on the quality of 'deep fakes', fake datasets or other unethical uses of generative modelling. For HVAEs, our work may inspire novel models which may lead to improved performance and stability of these models, also when used in applications with negative societal impact.

B.6 Model and training details

On the stability of training runs. VDVAE uses several techniques to improve the training stability of HVAEs: First, gradient clipping [162] is used, which reduces the effective learning rate of a mini-batch if the gradient norm surpasses a specific threshold. Second, gradient updates are skipped entirely when gradient norms surpass a second threshold, typically chosen higher than the one for gradient clipping. Third, gradient updates are also skipped if the gradient update would cause an overflow, resulting in NaN values.

In spite of the above techniques to avoid deterioration of training in very deep HVAEs, particularly when using a lot of weight-sharing, we experienced stability problems during late stages of training. These were particularly an issue on CIFAR10 in the late stages of training (on average roughly after 2 weeks of computation time), and often resulted in NaN values being introduced or posterior collapse. We did not extensively explore ways to prevent these in order to do minimal changes compared to vanilla VDVAE. We believe that an appropriate choice of the learning rate (e.g. with a decreasing schedule in later iterations) in combination with other changes to

the hyperparameters may greatly help with these issues, but principled fixes of, for instance, the instabilities identified in Theorem 3.5 are likewise important.

Gradient checkpointing, and other alternatives to reduce GPU memory cost. A practical limitation of training deep HVAEs (with or without weight-shared layers) is their GPU memory cost: Training deeper HVAEs means storing more intermediate activations in GPU memory during the forward pass, when memory consumption reaches its peak at the start of the backward pass. This limits the depth of the networks that can be trained on given GPU resources. To address this issue, particularly when training models which may not even fit on used hardware, we provide a prototype of a custom[1] *gradient checkpointing* implementation. Checkpointing occurs every few ResNet blocks which trades off compute for memory and can be used as an option. In gradient checkpointing, activations are stored only at specific nodes (checkpoints) in the computation graph, saving GPU memory, and are otherwise recomputed on-demand, requiring one additional forward pass per mini-batch [33]. Training dynamics remain unaltered. We note that other techniques exist specifically targeted at residual networks: For example, [98] propose to stochastically drop out entire residual blocks at training time [2]. This technique has two disadvantages: It changes training dynamics, and peak memory consumption varies between mini-batches, where particularly the latter is an inconvenient property for the practitioner as it may cause out-of-memory errors.

B.7 Additional experimental details and results

In this section, we provide additional experimental details and results.

[1]Our implementation deviates from the official PyTorch implementation of gradient checkpointing which is slow when using multiple GPUs, and is based on https://github.com/csrhddlam/pytorch-checkpoint.

[2]This technique is called "stochastic depth" as the active depth of the network varies at random. In this work, however, we go with our earlier definition of this term which refers to the number of stochastic layers in our network, and thus avoid using its name to prevent ambiguities.

Hyperparameters and hyperparameter tuning. In the following, we describe the hyperparameters chosen in our experiments. As highlighted in the main text, we use the state-of-the-art hyperparameters of VDVAE [34] wherever possible. This was possible for CIFAR10, ImageNet32 and ImageNet64. On MNIST and CelebA, VDVAE [34] did not provide experimental results. For MNIST, we took the hyperparameters of CIFAR10 as the basis and performed minimal hyperparameter tuning, mostly increasing the batch size and tuning the number and repetitions of residual blocks. For CelebA, we used the hyperparameters of ImageNet64 with minimal hyperparameter tuning, focussing on the number and repetitions of the residual blocks. For all datasets, the main hyperparameter we tuned was the number and repetitions (through weight-sharing) of residual blocks ceteris paribus, i.e. without searching over the space of other important hyperparameters. As a consequence, it is likely that further hyperparameter tuning would improve performance as changing the number of repetitions changes (the architecture of) the model.

We provide three disjunct sets of hyperparameters: *global* hyperparameters (Table B.2), which are applicable to all runs, *data-specific* hyperparameters (Table B.3), which are applicable to specific datasets, and *run-specific* hyperparameters, which vary by run. The run-specific hyperparameters will be provided in the respective subsections of §B.7, where applicable.

In the below tables, 'factor of # channels in conv. blocks' refers to the multiplicative factor of the number of channels in the bottleneck of a (residual) block used throughout VDVAE. '# channels of z_l' refers to C in the shape [C, H, W] of the latent conditional distributions in the approximate posterior and prior, where height H and width W are determined by the resolution of latent $\boldsymbol{z}_l$. Likewise, '# channels in residual state' refers to C in the shape [C, H, W] of the residual state flowing through the decoder of VDVAE. 'Decay rate γ of evaluation model' refers to the multiplicative factor by which the latest model parameters are weighted during training to update the evaluation model.

Table B.2: Global hyperparameters.

factor of # channels in conv. blocks	0.25
Gradient skipping threshold	3000
Adam optimizer: Weight decay	0.01
Adam optimizer: β_1	0.9
Adam optimizer: β_2	0.9

Table B.3: Data-specific hyperparameters.

Dataset	MNIST	CIFAR10	ImageNet32	ImageNet64	CelebA
Learning rate	0.0001	0.0002	0.00015	0.00015	0.00015
# iterations for learning rate warm-up	100	100	100	100	100
Batch size	200	16	8	4	4
Gradient clipping threshold	200	200	200	220	220
# channels of $\boldsymbol{z}_l$	8	16	16	16	16
# channels in residual state	32	384	512	512	512
Decay rate γ of evaluation model	0.9999	0.9999	0.999	0.999	0.999

B.7.1 "More from less": Parameter efficiency in HVAEs

In this experiment, we investigate the effect of repeating ResNet blocks in the bottom-up and top-down pass via weight-sharing. `rN` indicates that a ResNet block is repeated `N` times through weight-sharing where `r` is to be treated like an operator and `N` is a positive integer. In contrast, `xN`, already used in the official implementation of VDVAE, indicates `N` number of ResNet blocks without weight-sharing.

In Tables B.4 and B.5, we provide the NLLs on the test set at convergence corresponding to the NLLs on the validation set during training which we reported in Fig. 3.5. In general, weight-sharing tends to improve NLL, and models with significantly less parameters reach or even surpass other models with more parameters. We refer to the main text for the intuition of this behavior. In Table B.5 (CIFAR10), we find that the not weight-sharing runs have test NLLs noticeably deviating from the results on the validation set, yet the overall trend of more weight-sharing improving NLL tends to be observed. This is in line with our general

Table B.4: A small-scale study on parameter efficiency of HVAEs on *MNIST*. We compare models with one, two, three and four parameterised blocks per resolution ($\{x1, x2, x3, x4\}$) against models with a single parameterised block per resolution weight-shared $\{2, 3, 5, 10, 20\}$ times ($\{r2, r3, r5, r10, r20\}$). We report NLL ($\downarrow$) measured on the test set, corresponding to the results on the validation set in Fig. 3.5. NLL performance increases with more weight-sharing repetitions and surpasses models without weight-sharing but with more parameters.

Neural architecture	# Params	NLL ($\downarrow$)
`r1/x1`	107k	≤ 86.87
`r2`	107k	≤ 85.25
`r3`	107k	≤ 84.92
`r5`	107k	≤ 83.92
`r10`	107k	≤ 82.67
`r20`	107k	≤ 81.84
`x2`	140k	≤ 84.44
`x3`	173k	≤ 82.64
`x4`	206k	≤ 82.46

Table B.5: A small-scale study on parameter efficiency of HVAEs on *CIFAR10*. We compare models with with one, two, three and four parameterised blocks per resolution ($\{x1, x2, x3, x4\}$) against models with a single parameterised block per resolution weight-shared $\{2, 3, 5, 10, 20\}$ times ($\{r2, r3, r5, r10, r20\}$). We report NLL ($\downarrow$) measured on the test set, corresponding to the results on the validation set in Fig. 3.5. NLL performance tends to increase with more weight-sharing repetitions. However, in contrast to the validation set (see Fig. 3.5) where this trend is evident, it is less so on the test set.

Neural architecture	# Params	NLL ($\downarrow$)
`r1/x1`	8.7m	≤ 4.17
`r2`	8.7m	≤ 4.93
`r3`	8.7m	≤ 4.78
`r5`	8.7m	–
`r10`	8.7m	≤ 4.32
`r20`	8.7m	≤ 3.54
`x2`	13.0m	≤ 5.77
`x3`	17.3m	≤ 3.07
`x4`	21.6m	≤ 3.01

observation that our HVAE models are particularly unstable on CIFAR10.

In Table B.6, we provide key run-specific hyperparameters for the large-scale runs corresponding to Table 3.1 in the main text. Two points on the architecture

of the encoder and the decoder are worth noting: First, note that the decoder typically features more parameters and a larger stochastic depth than the encoder. We here follow VDVAE which observed this distribution of the parameters to be beneficial. Second, note that while we experienced a benefit of weight-sharing, there is a diminishing return of the number of times a specific cell is repeated. Hence, we typically repeat a single block for no more than 10-20 times, beyond which performance does not improve while computational cost increases linearly with the number of repetitions. Exploring how to optimally exploit the benefit of weight-sharing in HVAEs would be an interesting aspect for future work.

B.7.2 HVAEs secretly represent time and make use of it

In this experiment, we measure the L_2 norm of the residual state at every ResNet block in both the forward (bottom-up/encoder) and backward (top-down/decoder) model. Let x_i be the output of ResNet block i in the bottom-up model, and y_i be the input of ResNet block i in the top-down model for one batch. In the following, augmenting Fig. 3.6 on MNIST in the main text, we measure $\|x_i\|_2$ or $\|y_i\|_2$, respectively, over 10 batches. We also use this data to compute appropriate statistics (mean and standard deviation) which we plot. We measure the state norm in the forward and backward pass for models trained on CIFAR10 and ImageNet32 in Figs. B.4 and B.5, respectively. We note that the forward pass of the ImageNet32 has a slightly unorthodox, yet striking pattern in terms of state norm magnitude, presumably caused by an overparameterisation of the model. In summary, these findings provide further evidence that the residual state norm of VDVAEs represents time.

When normalising the residual state in our experiments in Table 3.2 (case "normalised"), we do so at the same positions where we measure the state norm above. At the output of every forward ResNet block x_i and the input of every

Table B.6: A large-scale study of parameter efficiency in HVAEs. We here provide key run-specific hyperparameters corresponding to the results reported in Table 3.1 in the main text. Note that the row order of our runs directly corresponds with Table 3.1. δ refers to gradient clipping threshold. γ refers to the gradient skipping threshold. We use the same nomenclature for number of cells (`x`) and number of repetitions for one block (`r`) as before. In addition, as in VDVAE's official code base, we use `d` to indicate average pooling, where the integer before `d` indicates the resolution on which we pool, and the integer after indicates the down-scaling factor. Further, `m` indicates interpolating, where we up-scale from a source (integer after `m`) to a target resolution (integer before `m`).

Dataset	Method	Batch size	δ	γ	Encoder Architecture	Decoder Architecture
MNIST 28×28	WS-VDVAE (ours)	70	-	200	32r3,32r3,32r3,32r3,32r3,32d2, 16r3,16r3,16r3,16d2, 8x6,8d2,4x3,4d4,1x3	1x1,4m1,4x2,8m4, 8x5,16m8,16r3,16r3,16r3,16r3,16r3,32m16, 32r3,32r3,32r3,32r3,32r3,32r3,32r3,32r3, 32r3,32r3,32r3
	VDVAE* (ours)	70	-	200	32x11,32d2,16x6,16d2, 8x6,8d2,4x3,4d4,1x3	1x1,4m1,4x2,8m4,8x5,16m8,16x10,32m16,32x21
CIFAR10 32×32	WS-VDVAE (ours)	16	400	4000	32r12,32r12,32r12,32r12,32r12,32r12, 32d2,16r12,16r12,16r12,16r12,16d2, 8r12,8r12,8r12,8r12,8r12,8r12,8d2, 4r12,4r12,4r12,4d4,1r12,1r12,1r12	1r12,4m1,4r12,4r12,8m4, 8r12,8r12,8r12,8r12,8r12, 16m8,16r12,16r12,16r12,16r12,16r12, 32m16,32r12,32r12,32r12,32r12, 32r12,32r12,32r12,32r12,32r12
	WS-VDVAE (ours)	16	200	2500	32r3,32r3,32r3,32r3,32r3,32r3,32r3, 32r3,32r3,32r3,32r3,32r3,32r3,32d2, 16r3,16r3,16r3,16r3,16r3,16r3,16d2, 8x6,8d2,4x3,4d4,1x3	1x1,4m1,4x2,8m4,8x5, 16m8,16r3,16r3,16r3,16r3,16r3, 16r3,16r3,16r3,16r3,16r3,32m16, 32r3,32r3,32r3,32r3,32r3,32r3,32r3, 32r3,32r3,32r3,32r3,32r3,32r3,32r3, 32r3,32r3,32r3,32r3,32r3,32r3,32r3, 32r3,32r3,32r3
	VDVAE* (ours)	16	200	400	32x11,32d2,16x6,16d2, 8x6,8d2,4x3,4d4,1x3	1x1,4m1,4x2,8m4,8x5, 16m8,16x10,32m16,32x21
ImageNet 32×32	WS-VDVAE (ours)	8	200	5000	32r10,32r10,32r10,32r10,32d2, 16r10,16r10,16r10,16d2,8x8,8d2, 4x6,4d4,1x6	1x2,4m1,4x4,8m4,8x9,16m8, 16r10,16r10,16r10,16r10,16r10,32m16, 32r10,32r10,32r10,32r10,32r10,32r10, 32r10,32r10,32r10,32r10
	WS-VDVAE (ours)	8	200	5000	32r6,32r6,32r6,32r6,32r6, 32r6,32r6,32r6,32r6,32d2, 16r6,16r6,16r6,16r6,16r6,16d2, 8x8,8d2,4x6,4d4,1x6	1x2,4m1,4x4,8m4,8x9, 16m8,16r6,16r6,16r6, 16r6,16r6,16r6,16r6,16r6,16r6, 16r6,16r6,32m16,32r6,32r6, 32r6,32r6,32r6,32r6,32r6,32r6, 32r6,32r6,32r6,32r6,32r6,32r6,32r6,32r6, 32r6,32r6,32r6,32r6,32r6,32r6,32r6,32r6,32r6
	VDVAE* (ours)	8	200	300	32x15,32d2,16x9,16d2,8x8,8d2, 4x6,4d4,1x6	1x2,4m1,4x4,8m4,8x9,16m8,16x19,32m16,32x40
CelebA 64×64	WS-VDVAE (ours)	4	220	3000	64r3,64r3,64r3,64r3,64r3,64r3,64r3,64r3, 64d2,32r3,32r3,32r3,32r3,32r3,32r3, 32r3,32r3,32r3,32r3,32r3,32r3,32d2, 16r3,16r3,16r3,16r3,16r3,16r3,16d2, 8r3,8r3,8r3,8d2,4r3,4r3,4r3,4d4, 1r3,1r3,1r3	1r3,1r3,4m1,4r3,4r3,4r3, 8m4,8r3,8r3,8r3,8r3, 16m8,16r3,16r3,16r3,16r3,16r3,16r3,16r3, 32m16,32r3,32r3,32r3,32r3,32r3,32r3, 32r3,32r3,32r3,32r3,32r3,32r3,32r3, 32r3,32r3,32r3,64m32,64r3,64r3,64r3,64r3, 64r3,64r3,64r3,64r3
	VDVAE* (ours)	4	220	3000	64x11,64d2,32x20,32d2, 16x9,16d2,8x8,8d2,4x7,4d4,1x5	1x2,4m1,4x3,8m4,8x7,16m8, 16x15,32m16,32x31,64m32,64x12

backward ResNet block y_i, we assign

$$x_i \leftarrow \frac{x_i}{\|x_i\|_2} \qquad\qquad y_i \leftarrow \frac{y_i}{\|y_i\|_2}$$

for every mini-batch during training. This results in a straight line in these plots for the "normalised" case. As the natural behavior of VDVAEs is—as we measured—to learn a non-constant norm, normalising the state norm has a deteriorating consequence, as we observe in Table 3.2. In contrast, the regular, unnormalised runs (case "non-normalised") show well-performing results.

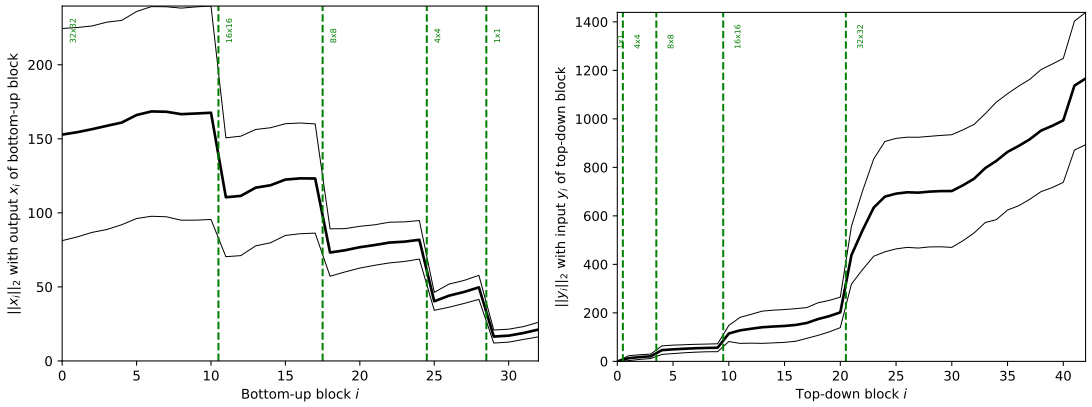


Figure B.4: HVAEs are secretly representing time *on CIFAR10*: We measure the L_2-norm of the residual state at every residual block i for the [Left] forward (bottom-up) pass, and [Right] the backward (top-down) pass, respectively, over 10 batches with 100 data points each. The thick line refers to the average and the thin, outer lines refer to ± 2 standard deviations.

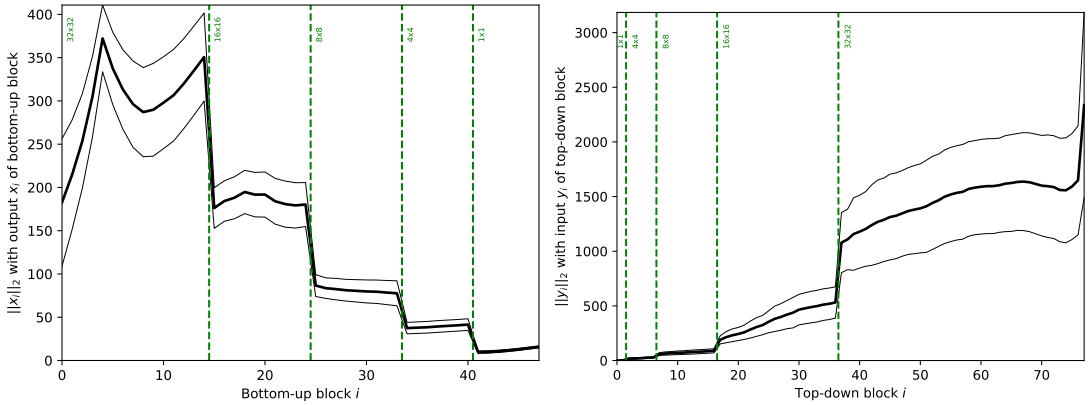


Figure B.5: HVAEs are secretly representing time *on ImageNet32*: We measure the L_2-norm of the residual state at every residual block i for the [Left] forward (bottom-up) pass, and [Right] the backward (top-down) pass, respectively, over 10 batches with 100 data points each. The thick line refers to the average and the thin, outer lines refer to ± 2 standard deviations.

We further analysed the normalised state norm experiments in Table 3.2. The normalised MNIST and CIFAR10 runs terminated early (indicated by ✗), more precisely after 18 hours and 4.5 days of training, respectively. From the very start of the optimisation, the normalised models have poor training behavior. To show this, in Fig. B.6, we illustrate the NLL on the validation set during training for the three normalised runs as compared to regular, non-normalised training. Validation ELBO only improves for a short time, after which the normalised runs deteriorate, showing no further improvement or even a worse NLL.

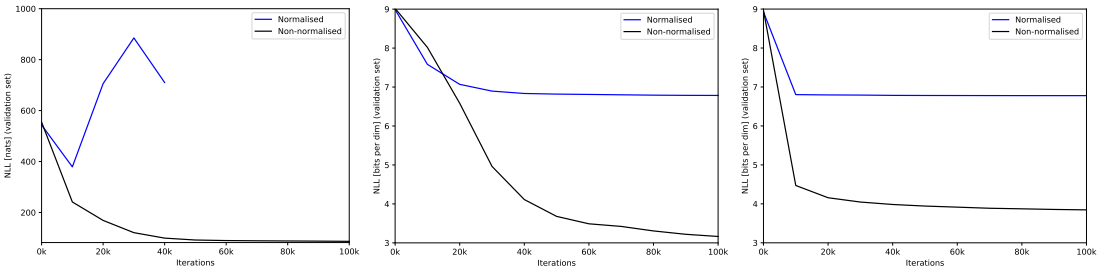


Figure B.6: On the training dynamics of VDVAE with and without a normalised residual state norm. NLL (↓) measured on the validation set of MNIST [left], CIFAR10 [middle] and ImageNet32 [right]. The normalised runs suffer from poor training dynamics from the very start of the optimisation and even terminate early on MNIST and CIFAR10, indicating that VDVAE makes use of the time representing state norm during training.

B.7.3 Sampling instabilities in HVAEs

When retrieving unconditional samples from our models, we scale the variances in the unconditional distributions with a temperature factor τ, as is common practice. We tune τ "by eye" to improve the fidelity of the retrieved images, yet do not cherry pick these samples. In Figs. B.7 to B.11, we provide additional, not cherry-picked unconditional samples for models trained on CIFAR10, ImageNet32, ImageNet64, MNIST and CelebA, extending those presented in Fig. 3.7. As shown earlier, the instabilities in VDVAE result in poor unconditional samples for CIFAR10, ImageNet32 and ImageNet64, but relatively good samples for MNIST and CelebA.

In addition, we here also visualise the representational advantage of HVAEs. Fig. B.12 shows samples where we gradually increase the number of samples from the posterior vs. the prior distributions in each resolution across the columns. This means that in column 1, we sample the first latent $\boldsymbol{z}_l$ *in each resolution* from the (on encoder activations conditional) posterior q, and all other latents from the prior p. A similar figure, but gradually increasing the contribution of the posterior across the blocks of all resolutions (i.e. column 1 samples $\boldsymbol{z}_l$ from the posterior in the very first resolution only) is shown in VDVAE [34, Fig. 4]. Fig. B.12

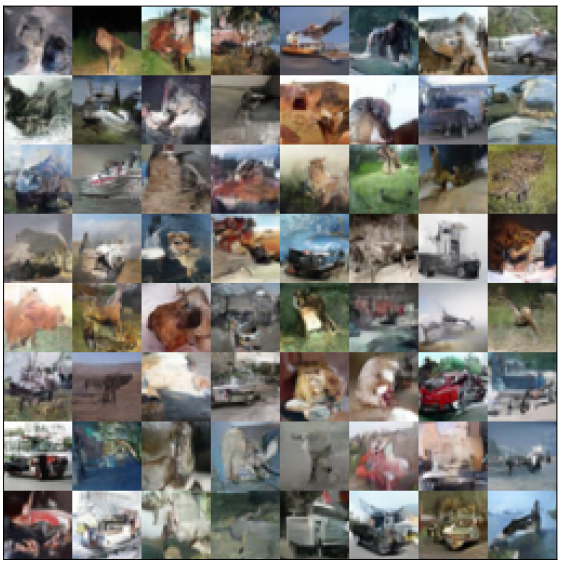


Figure B.7: Further unconditional samples (not cherry-picked) of `VDVAE`* on *CIFAR10*, augmenting those presented in Fig. 3.7. While samples on MNIST and CelebA demonstrate high fidelity and diversity, samples on CIFAR10, ImageNet32 and ImageNet64 are diverse, but unrecognisable, demonstrating the instabilities identified by Theorem 3.5. We chose the temperature as $\tau = 0.9$.

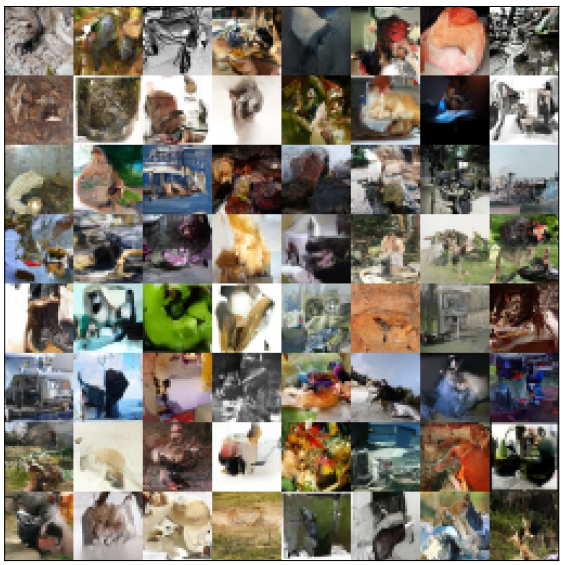


Figure B.8: Further unconditional samples (not cherry-picked) of `VDVAE`* on *ImageNet32*, augmenting those presented in Fig. 3.7. While samples on MNIST and CelebA demonstrate high fidelity and diversity, samples on CIFAR10, ImageNet32 and ImageNet64 are diverse, but unrecognisable, demonstrating the instabilities identified by Theorem 3.5. We chose the temperature as $\tau = 1.0$.

Figure B.9: Further unconditional samples (not cherry-picked) of `VDVAE`* on *ImageNet64*, augmenting those presented in Fig. 3.7. While samples on MNIST and CelebA demonstrate high fidelity and diversity, samples on CIFAR10, ImageNet32 and ImageNet64 are diverse, but unrecognisable, demonstrating the instabilities identified by Theorem 3.5. Temperatures τ are tuned for maximum fidelity. We chose the temperature as $\tau = 0.9$.

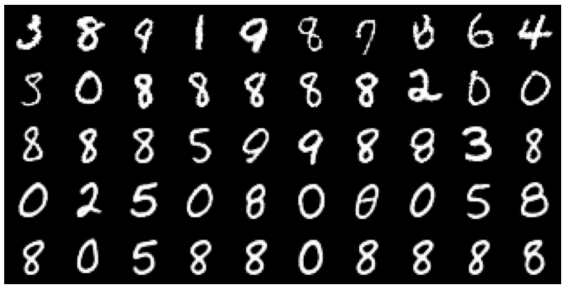


Figure B.10: Further unconditional samples (not cherry-picked) of `VDVAE`* on *MNIST*, augmenting those presented in Fig. 3.7. While samples on MNIST and CelebA demonstrate high fidelity and diversity, samples on CIFAR10, ImageNet32 and ImageNet64 are diverse, but unrecognisable, demonstrating the instabilities identified by Theorem 3.5. We chose the temperatures as $\tau \in \{1.0, 0.9, 0.8, 0.7, 0.5\}$ (corresponding to the rows).

B.7.4 Ablation studies

Number of latent variables. The number of latent variables increase when increasing the stochastic depth through weight-sharing. Thus, an important ablation study is the question whether simply increasing the number of latent variables improves HVAE performance, which may explain the weight-sharing effect. On CIFAR10, we find that this is not the case: In Table B.7, we analyse the effect of

Figure B.11: Further unconditional samples (not cherry-picked) of `VDVAE`* on *CelebA*, augmenting those presented in Fig. 3.7. While samples on MNIST and CelebA demonstrate high fidelity and diversity, samples on CIFAR10, ImageNet32 and ImageNet64 are diverse, but unrecognisable, demonstrating the instabilities identified by Theorem 3.5. We chose the temperature as $\tau = 0.5$.

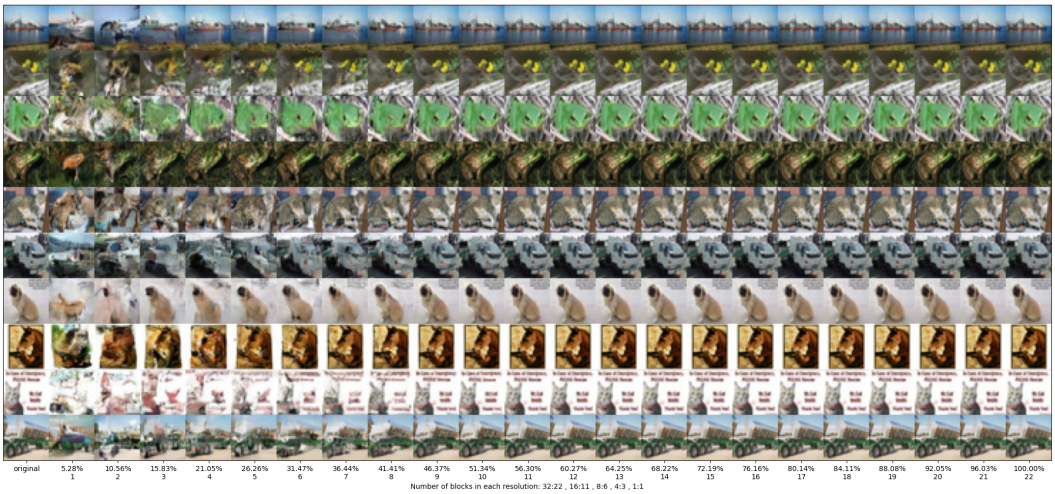


Figure B.12: Samples drawn from our model when gradually increasing the contribution of the approximate posterior. In each column with integer s, we sample the first s latent variables from the approximate posterior in each resolution, i.e. $\mathbf{z}_i \sim q(\mathbf{z}_i|\mathbf{z}_{>i})$ (up to the maximum number of latent variables in each resolution), and $\mathbf{z}_j \sim p(\mathbf{z}_j|\mathbf{z}_{>j})$ for all other latent variables. The percentage number indicates the fraction of the number of latent variables among all latent variables sampled from the approximate posterior. In the left-most column, we visualise corresponding input images.

increasing the number of latent variables ceteris paribus. Furthermore, in Fig. B.13, we report validation NLL during training for the same runs. In this experiment, we

realise the increase in number of latent variables by increasing the number of channels in each latent variable $\mathbf{z}_l$ exponentially while slightly decreasing the number of blocks so to keep the number of parameters roughly constant. Both results indicate that the number of latent variables, at least for this configuration on CIFAR10, do not add performance and hence cannot explain the weight-sharing performance.

Table B.7: On the effect of the number of latent variables on CIFAR10. We report the NLL on the test set at convergence.

# of latent variables	# Params	NLL (↓)
396k	39m	2.88
792k	39m	2.88
1.584m	39m	2.87
3.168m	39m	2.88

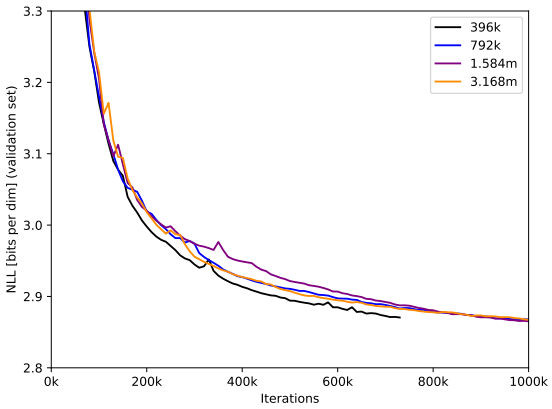


Figure B.13: On the effect of the number of latent variables. We report NLL on the validation set of CIFAR10 during training.

Fourier features. In this experiment, we are interested in the effect of Fourier features imposed onto a Haar wavelet basis due to the inductive bias of the U-Net. Intuitively, we would expect that Fourier features do not add performance as the U-Net already imposes a good basis for images. We now validate this hypothesis experimentally: We compute Fourier features in every ResNet block at three different locations as additional channels and varying frequencies. We implement Fourier features closely following VDM [120]: Let $h_{i,j,k}$ be an element of a hidden activation of the network, for example of a sampled latent $h = \mathbf{z}_l$, in spatial position (i, j)

and channel k. Then, for each scalar $h_{i,j,k}$, we add two transformed scalars for each frequency governed by β as follows:

$$f^{\beta}_{i,j,k} = \sin\left(z_{i,j,k}2^{\beta}\pi\right), \text{ and } g^{\beta}_{i,j,k} = \cos\left(z_{i,j,k}2^{\beta}\pi\right).$$

In our experiments, we experiment with different choices for β, but typically select two values at a time (as in VDM), increasing the number of channels in the resulting activation by a factor of five. Fourier features are computed on and concatenated to activations at three different locations (in three separate experiments): At the input of the ResNet block, after sampling, and for the input of the two branches parameterising the posterior and prior distributions.

In Tables B.8 and B.9, we report performance when concatenating Fourier features at every ResNet block in these three locations. In all cases, Fourier features deteriorate performance in this multi-resolution wavelet basis, particularly for high-frequencies which often lead to early termination due to numeric overflows. However, if training only a single-resolution model where no basis is enforced, training does not deteriorate, not even for high-frequency Fourier features, yet performance can neither be improved. Furthermore, we experimented with computing and concatenating the Fourier features only to the input image of the model, hypothesising numerical instabilities caused by computing Fourier transforms at every ResNet block, and report results in Table B.10. Here, performance is significantly better as runs no longer deteriorate, but Fourier features still do not improve performance compared to not using Fourier features at all.

On the effect of a multi-resolution bridge. State-of-the-art HVAEs have a U-Net architecture with pooling and, hence, are multi-resolution bridges (see Theorem 3.4). We investigate the effect of multiple resolutions in HVAEs (here with spatial dimensions $\{32^2, 16^2, 8^2, 4^2, 1^2\}$) against a single resolution (here with spatial dimension 32^2). We choose the number of blocks for the single resolution model such that they are distributed in the encoder and decoder proportionally to the multi-resolution model and the total number of parameters are equal in

Table B.8: Fourier features introduced and concatenated in every ResNet block at three different locations on *MNIST*. VDVAE typically deteriorates or has poor performance.

Exponent β	NLL
Loc. 1	
$[1, 2]$	≤ 78.4
$[3, 4]$	≤ 80.55
$[5, 6]$ (✗)	–
Loc. 2	
$[1, 2]$	≤ 554.50
$[3, 4]$ (✗)	–
$[5, 6]$ (✗)	–
Loc. 3	
$[1, 2]$ (✗)	–
$[2, 3]$	≤ 306.67
$[3, 4]$	≤ 345.67
Loc. 1 & single-res.	
$[3, 4]$	≤ 87.55
$[5, 6]$	≤ 86.96
$[7, 8]$	≤ 91.67
No Fourier Features	≤ 79.81

Table B.9: Fourier features introduced and concatenated in every ResNet block at three different locations on *CIFAR10*. VDVAE typically deteriorates or achieves a poor performance.

Exponent β	NLL
Loc. 1	
$[3, 4]$ (✗)	–
$[5, 6]$ (✗)	–
$[7, 8]$	≤ 8.94
Loc. 2	
$[3, 4]$ (✗)	–
$[5, 6]$ (✗)	–
$[7, 8]$ (✗)	–
Loc. 3	
$[3, 4]$ (✗)	–
$[5, 6]$	≤ 8.94
$[7, 8]$	≤ 8.99
No Fourier Features	≤ 2.87

Table B.10: Fourier features introduced on the input image of the model only, with results on *CIFAR10*. While performing better than if introduced at every ResNet block, still Fourier features do not improve performance compared to using no Fourier features at all.

Exponent β	NLL
Fourier Features on input only	
$[3, 4]$	≤ 2.95
$[5, 6]$	≤ 2.96
$[7, 8]$	≤ 2.89
No Fourier Features	≤ 2.87

both, ensuring a fair comparison. As we show in Table B.11, the multi-resolution models perform slightly better than their single-resolution counterparts, yet we would have expected this difference to be more pronounced. We also note that it may be worth measuring other metrics for instance on fidelity, such as the FID score [89]. Additionally, multi-resolution models have a representational advantage due to their Haar wavelet basis representation (illustrated in Appendix B.7.4, Fig. B.12).

Table B.11: Single- vs. multi-resolution HVAEs.

# Resolutions	# Params	NLL
MNIST		
Single	328k	≤ 81.40
Multiple	339k	≤ 80.14
CIFAR10		
Single	39m	≤ 2.89
Multiple	39m	≤ 2.87
ImageNet32		
Single	119m	≤ 3.68
Multiple	119m	≤ 3.67

On the importance of a stochastic differential equation structure in HVAEs. A key component of recent HVAEs is a residual cell, as outlined in §3.4. The residual connection makes HVAEs discretise an underlying SDE, as we outlined in this work. Experimentally, it was previously noted as being crucial for stability of very deep HVAEs. Here, we are interested in ablating the importance of

imposing an SDE structure into HVAEs: We compare models with a residual HVAE cell (as in VDVAE) with a non-residual HVAE cell which is as close to VDVAE as possible to ensure a fair comparison. The non-residual VDVAE cell does not possess a residual state which flows through the backbone architecture. We achieve this by removing the connection between the first and second element-wise addition in VDVAE's cell (see [34, Fig. 3]), which is equivalent to setting $Z_{i,+} = 0$. Hence, in the non-residual cell, during training and evaluation, the reparameterised sample is directly taken forward. Note that this is distinct from the Euler-Maruyama cell which features a residual connection. Our experiments confirm that a *residual* cell is key for training stability, as illustrated in Table B.12 and Fig. B.14: Without a residual state flowing through the decoder, models quickly experience posterior collapse of the majority of layers during training.

Table B.12: Residual vs. non-residual VDVAE cell. The residual HVAE strongly outperforms a non-residual VDVAE cell, where the latter's training deteriorates. This is also analysed in Fig. B.12. We report NLL on the test set at convergence, or at the last model checkpoint before deterioration of training.

Cell type	NLL
MNIST	
Residual VDVAE cell	≤ 80.05
Non-residual VDVAE cell	≤ 112.58
CIFAR10	
Residual VDVAE cell	≤ 2.87
Non-residual VDVAE cell	≤ 3.66
ImageNet	
Residual VDVAE cell	≤ 3.667
Non-residual VDVAE cell (✗)	≤ 4.608

Synchronous vs. asynchronous processing in time. During the bottom-up pass, VDVAE takes forward the activation of the last time step in each resolution which is passed to every time step in the top-down pass on the same resolution (see Fig. 3 in VDVAE [34]). In this ablation study, we were interested in this slightly peculiar choice of an *asynchronous* forward and backward process and to what degree it is important for performance. We thus compare an asynchronous

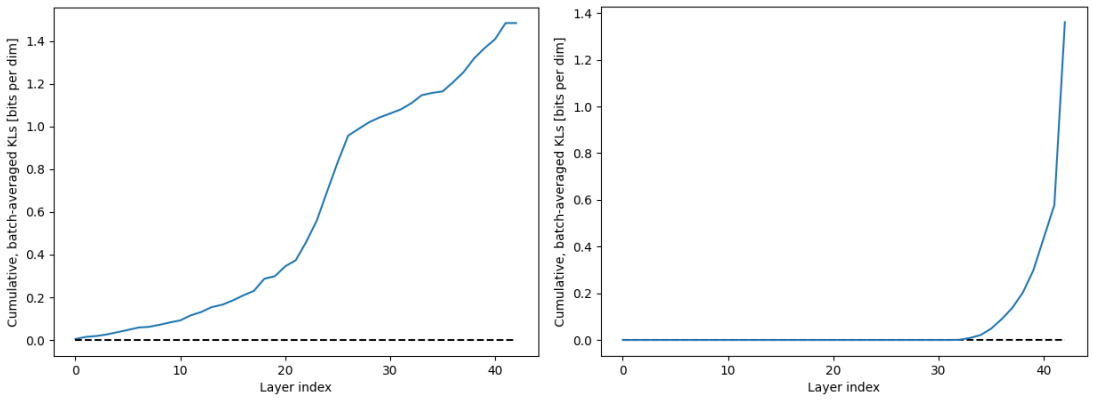


Figure B.14: Cumulative sum of KL-terms in the ELBO of a residual and non-residual VDVAE, averaged over a batch at convergence. We report the two CIFAR10 runs in Table B.12. The posterior collapses for the majority of the latent variables in the non-residual VDVAE cell case [right], but carries information for all latent variables in the regular, residual cell case [left].

model, with skip connections as in VDVAE [34], with a *synchronous* model, where activations from the bottom-up pass are taken forward to the corresponding time step in the top-down pass. In other words, in the synchronous case, the skip connection mapping between time steps in the encoder and decoder is 'bijective', and it is not 'injective', but 'surjective' in the asynchronuous case. We realise the synchronous case by choosing the same number of blocks in the encoder as `VDVAE`* has in the decoder, i.e. constructing a 'symmetric' model. To ensure a fair comparison, both models (synchronous and asynchronous) are further constructed to have the same number of parameters. In Table B.13, we find that synchronous and asynchronous processing achieve comparable NLL, indicating that the asynchronous design is not an important contributor to performance in VDVAE. We note, however, that an advantage of the asynchronous design, which is exploited by VDVAE, is that the bottom-up and top-down architectures can have different capacities, i.e. have a different number of ResNet blocks. VDVAE found that a more powerful decoder was beneficial for performance [34].

Table B.13: Synchronous vs. asynchronous processing in time. We report NLL on the test set on CIFAR10 and ImageNet32, respectively.

Processing	NLL
CIFAR10	
Synchronous	≤ 2.85
Asynchronous	≤ 2.86
ImageNet32	
Synchronous	≤ 3.69
Asynchronous	≤ 3.69

C

Appendix of *A Unified Framework for U-Net Design and Analysis*

Contents

C.1 Theoretical Details and Technical Proofs

C.1.1 Proofs of theoretical results in the main text

Theorem 4.1. Suppose U_i^* and U^* are solutions of the L^2 regression problem. Then, $\mathcal{L}^2_{i|j}(U_i^*) \leq \mathcal{L}^2_{|j}(U^*)$ with equality as $i \to \infty$. Further, if $Q_i U^*$ is V_i-measurable, then $U_i^* = Q_i U^*$ minimises $\mathcal{L}^2_i$.

Proof. Let $\mathcal{U} = (\mathcal{V}, \mathcal{W}, \mathcal{E}, \mathcal{D}, \mathcal{P}, U_0)$ be a U-Net in canonical form, that is where each encoder map E_i in $\mathcal{E}$ is set to be the identity map, and further assume W is a Hilbert space with $\mathcal{W}$ being a sequence of subspaces spanned by orthogonal basis vectors for W. For a V-valued random variable v, define the σ-algebra

$$\mathcal{H}_j \coloneqq \sigma(P_j v) = \sigma(v_j). \tag{C.1}$$

Now the filtration

$$\mathcal{H}_1 \subset \mathcal{H}_2 \subset \cdots \tag{C.2}$$

increases to $\mathcal{H} \coloneqq \sigma(v)$. These are the σ-algebras generated by v_j. For data $w \in W$, we may define the losses

$$\mathcal{L}^2_{i|j}(U_{i|j}) \coloneqq \frac{1}{|\mathcal{S}|} \sum_{(w_i, v_j) \in \mathcal{S}} \|w_i - U_{i|j}(v_j)\|^2, \tag{C.3}$$

for $U_{i|j} : V_j \mapsto W_i$ where $w_i = Q_i w$. Let $\mathcal{F}_{i|j}$ be the set of measurable functions from V_j to W_i, then the conditional expectation $\mathbb{E}(w_i|v_j)$ is given by the solution of the regression problem

$$\mathbb{E}(w_i|v_j) = \operatorname*{argmin}_{U_{i|j} \in \mathcal{F}_{i|j}} \mathcal{L}^2_{i|j}(U_{i|j}), \tag{C.4}$$

where the conditioning is respect to the σ-algebra $\mathcal{H}_j$ for the response variable w_i. For a fixed $i < i'$ and any $j > 0$ we have that

$$\mathcal{L}^2_{i'|j}(U_{i'|j}) = \frac{1}{|\mathcal{S}|} \sum_{(w_{i'},v_j)\in\mathcal{S}} \|w_{i'} - U_{i'|j}(v_j)\|^2 \tag{C.5}$$

$$= \frac{1}{|\mathcal{S}|} \sum_{(w_{i'},v_j)\in\mathcal{S}} \left(\|Q_i(w_{i'} - U_{i'|j}(v_j))\|^2 + \|Q_i^{\perp}(w_{i'} - U_{i'|j}(v_j))\|^2\right) \tag{C.6}$$

$$= \mathcal{L}^2_{i|j}(Q_i U_{i'|j}) + \frac{1}{|\mathcal{S}|} \sum_{(w_{i'},v_j)\in\mathcal{S}} \|Q_i^{\perp}(w_{i'} - U_{i'|j}(v_j))\|^2. \tag{C.7}$$

Any $U_{i'|j}$ admits the parameterisation $U_{i'|j} = U_{i|j} + U^{\perp,i}_{i'|j}$ where $U_{i|j}$ is in $\mathcal{F}_{i|j}$ and $U^{\perp,i}_{i'|j} \in \mathcal{F}_{i'|j}$ and vanishes on W_i. Under this parameterisation we gain

$$\mathcal{L}^2_{i'|j}(U_{i'|j}) = \mathcal{L}^2_{i|j}(U_{i|j}) + \frac{1}{|\mathcal{S}|} \sum_{(w_{i'},v_j)\in\mathcal{S}} \|Q_i^{\perp}(w_{i'} - U^{\perp,i}_{i'|j}(v_j))\|^2, \tag{C.8}$$

and further,

$$\operatorname*{argmin}_{U_{i'|j}\in\mathcal{F}_{i'|j}} \mathcal{L}^2_{i'|j}(U_{i'|j}) = \operatorname*{argmin}_{U_{i|j}\in\mathcal{F}_{i|j}} \mathcal{L}^2_{i|j}(U_{i|j}) + \operatorname*{argmin}_{U^{\perp,i}_{i'|j}} \frac{1}{|\mathcal{S}|} \sum_{(w_{i'},v_j)\in\mathcal{S}} \|Q_i^{\perp}(w_{i'} - U^{\perp,i}_{i'|j}(v_j))\|^2. \tag{C.9}$$

In particular, when $i' \to \infty$, if we define $\mathcal{L}^2_{|j}$ to be the loss on W then

$$\operatorname*{argmin}_{U_{|j}\in\mathcal{F}_{|j}} \mathcal{L}^2_{|j}(U_{|j}) = \operatorname*{argmin}_{U_{i|j}\in\mathcal{F}_{i|j}} \mathcal{L}^2_{i|j}(U_{i|j}) + \operatorname*{argmin}_{U^{\perp,i}_{|j}} \sum_{(w,v_j)\in\mathcal{S}} \|Q_i^{\perp}(w - U^{\perp,i}_{|j}(v_j))\|^2. \tag{C.10}$$

So for any j, we have that $\mathcal{L}^2_{i|j}(U^*_{i|j}) \leq \mathcal{L}^2_{|j}(U^*_{|j})$. Further, as $i \to \infty$, the truncation error

$$\sum_{(w,v_j)\in\mathcal{S}} \|Q_i^{\perp}(w - U^{\perp,i}_{|j}(v_j))\|^2, \tag{C.11}$$

tends to zero. Now assume that $Q_i U^*$ is V_i-measurable then $Q_i U^* \in \mathcal{F}_{i|j}$ and if this did not minimise $\mathcal{L}^2_{i|j}$, then choosing the minimiser of $\mathcal{L}^2_{i|j}$ and constructing $U = U^*_i + U^{\perp,i}$ will have $\mathcal{L}^2(U) < \mathcal{L}^2(U^*)$, contradicting U^* being optimal.

□

Proposition 4.1. If $\mathcal{U}$ is a residual U-Net, then U_i is a ResNet preconditioned on $U_i^{\text{pre}}(v_i) = U_{i-1}(\tilde{v}_{i-1})$, where $\tilde{v}_{i-1} = P_{i-1}(E_i(v_i))$.

Proof. For a ResNet $R(v_i) = R^{\text{pre}}(v_i) + R^{\text{res}}(v_i)$ select $R^{\text{pre}}(v_i) = U_{i-1}(\tilde{v}_{i-1})$ where $\tilde{v}_{i-1} = P_{i-1}(E_i(v_i))$. □

Theorem 4.2. For time $t \geq 0$ and $j \geq i$, $Q_i X_j(t) \overset{d}{=} X_i(t)$. Furthermore if $X_i(t) = \sum_{j=0}^{i} \widehat{X}^{(j)}(t) \cdot \widehat{\boldsymbol{\phi}}_j$, be the decomposition of $X_i(t)$ in its Haar wavelet frequencies (see Appendix C.3). Each component $\widehat{X}^{(j)}(t)$ of the vector has variance 2^{j-1} relative to the variance of the base Haar wavelet frequency.

Proof. Let $X_i \in V_i$ represented in the standard basis $\Phi = \{\phi_k : k = 1, \ldots, 2^i\}$ giving

$$X_i = \sum_k X_i^{(k)} \phi_k. \tag{C.12}$$

To transform this into its Haar wavelet representation where average-pooling is conjugate to basis projection, we can use the map $T_i : V_i \mapsto V_i$ defined by

$$T_i = \Lambda_i H_i, \tag{C.13}$$

where H_i is the Haar-matrix on resolution i and Λ_i is a diagonal scaling matrix with entries

$$(\Lambda_i)_{k,k} = 2^{-i+j-1}, \quad \text{if } k \in \{2^{j-1}, \ldots, 2^j\}, \tag{C.14}$$

for $j > 0$ and equal to 2^{-i} for the initial case $j = 0$. For example, the matrix for a four-pixel image is

$$2^{-2} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2^1 & 0 \\ 0 & 0 & 0 & 2^1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix} = \Lambda_2 H_2. \tag{C.15}$$

Now given the vector of coefficients $\mathbf{X}_i = (X_i^{(k)})_{k=1}^{2^i}$, the coefficients in the Haar frequencies can be given by

$$\widehat{\mathbf{X}}_i \coloneqq T_i(\mathbf{X}_i). \tag{C.16}$$

We define the vectors

$$(\widehat{X_j})_k = (T_i(\mathbf{X}_i))_{k+2^{j-1}}, \tag{C.17}$$

for $j \in \{1, \ldots, 2^{j-1}\}$. Now we may write X_i in terms of its various frequency components as

$$X_i = \sum_{j=0}^{i} \widehat{X}^{(j)} \cdot \widehat{\boldsymbol{\phi}}_j. \tag{C.18}$$

Suppose that X_i is a time varying random variable on V_i that evolve according to

$$X_i^{(k)}(t) \coloneqq \sqrt{1-\alpha_t} X_i^{(k)} + \sqrt{\alpha_t}\varepsilon^{(k)}, \tag{C.19}$$

when represented in its pixel basis where $\varepsilon^{(k)}$ is a standard normally distributed random variable. We may represent X_i in its frequencies, with a random and time dependence on the coefficient vectors,

$$X_i = \sum_{j=0}^{i} \widehat{X}^{(j)}(t) \cdot \widehat{\boldsymbol{\phi}}_j. \tag{C.20}$$

To analyse the variance of $\widehat{X}^{(j)}(t)$, we may analyse the variance of a standard normally distributed random vector $\varepsilon_i = (\varepsilon^k)_{k=1}^{2^i}$ under the mapping T_i (the variance of the data is not considered in this proof by assumption). Due to the symmetries in the matrix representation for T_i, the variance for each element of $\widehat{X}^{(j)}(t)$ is the same, and by direct computation

$$\text{VAR } (T_i\varepsilon_i)_k = 2^{-i+j-1} \quad \text{for } i > 0,\ k \in \{2^{j-1}, \ldots, 2^j\}, \tag{C.21}$$

and 2^{-i} for when $i = 0$. To see this note that

$$\text{VAR } (T_i\varepsilon_i) = \text{VAR}(\Lambda_i H_i \varepsilon_i) = \Lambda_i^2 \text{VAR}(H_i\varepsilon_i), \tag{C.22}$$

as Λ_i is a diagonal matrix. Now each of the elements of ε_i are independent and normally distributed, so if $(H_i)_{k,:}$ is the k^{th} row of H_i, then

$$\text{VAR}(H_i\varepsilon_i)_k = \text{VAR}((H_i)_{k,:}\varepsilon_i) = \|(H_i)_{k,:}\|_0, \tag{C.23}$$

where $\|(H_i)_{k,:}\|_0$ is the amount of non-zero elements of $(H_i)_{k,:}$ as each entry of H_i is 0, or ± 1. For $(H_i)_{k,:}$ where $k \in \{2^{j-1}, \ldots, 2^j\}$ there are 2^{i-j+1} entries. Further, in this position we have $(\Lambda_i)_k = 2^{-i+j-1}$, so putting this together finishes the count. Now see that for the evolution of $X_i^{(k)}(t)$, the random part of Eq. (C.19) is simply a Gaussian

random vector multiplied by $\sqrt{\alpha_t}$. Therefore we have that $\mathrm{VAR}X_i^{(k)}(t) = \alpha_t 2^{-i+j-1}$. To get the relative frequency, for k we divide this variance by the variance of the base frequency, the zeroth resolution. This yields $\alpha_t 2^{-i+j-1}/\alpha_t 2^{-i} = 2^{j-1}$.

Now we show how the diffusion on a given resolution i can be embedded into a higher resolution of $i+1$, which maintains consistency with the projection map Q_i. Define the diffusion on $i+1$ to be

$$X_{i+1}^{(k)}(t) \coloneqq \sqrt{1-\alpha_t} X_{i+1}^{(k)} + \sqrt{2\alpha_t}\varepsilon^{(k)} \tag{C.24}$$

Let $\widehat{X}_i^{(j)}(t)$ be the Haar coefficients for the initial diffusion defined on resolution i and $\widehat{X}_i^{(j)}(t)$ be the coefficients defined on resolution $i+1$. Both of these random vectors are linear transformations of Gaussian random vectors, so we need only confirm that the means and variances on the first j entries agree to show that these are equal in distribution. If $X_i^{(k)} = \frac{X_{i+1}^{(k)} + X_{i+1}^{(k+1)}}{2}$, that is the initial data on resolution $i+1$ averages to the initial data on resolution i, then by construction the means of the random vectors $\widehat{X}_i^{(j)}(t)$ and $\widehat{X}_{i+1}^{(j)}(t)$ agree for the first i resolutions. Note that the variances for the components of the noise for either process are 2^{-i+j-1} when the diffusion on $i+1$ is given by Eq. (C.24). In general, for the process on $i+\ell$ to embed into our original diffusion on resolution i we require scaling the noise term by a factor of $2^{\ell/2}$. Again, this shows that the noising process on a frequency i has the natural extension noising the high-frequency components exponentially faster than the reference frequency i. □

We may comment on how this is represented in its natural sequence space. Assume that $X_i \in V_i \subset L^2([0,1])$. There is a natural mapping from V_i to ℓ_2 defined by

$$\ell_2 \coloneqq \{s | \sum_{k=0}^{\infty} s_k^2 < \infty\} \tag{C.25}$$

through the mapping $\pi : L^2([0,1]) \mapsto \ell_2$ via

$$\pi(X_i) = (\widehat{X}^{(1)}, \widehat{X}^{(2)}, \dots). \tag{C.26}$$

For elements $X_i \in V_i$ the image of π is not surjective and has image contained in ℓ_{00} — the set of infinite sequences which eventually are zero. In this case we have a finite dimensional mapping. Theorem 2 simply states that the forward noising process in a diffusion model, under transformation into its Haar basis, has variance

$$\text{VAR } (\pi X_i)_k = 2^{-i+j-1} \quad \text{for } i > 0,\ k \in \{2^{j-1}, \dots, 2^j\}, \tag{C.27}$$

implying that the sequence image of the datum has monotonically increasing variance in the Haar wavelet sequence space.

C.1.2 Diffusions on the sphere.

Suppose that we have $L^2(\mathbb{X})$ valued data on the globe and we would like to form a diffusion model on this domain. Instead of constructing a local change of coordinate system over a manifold and projecting our diffusion to a square domain [51], we could simply design our U-Net to encode our geometry. As an example of this, we can take a standard triangulation of the globe, then apply our triangular basis U-Net directly to our data.

For the data on the globe, we could then choose a resolution i to refine our triangular domain to, see Figure C.1 for a refinement level of $i = 2$ which could be used to tessellate the globe. For each 'triangle pixel' in our refinement, we could define the diffusion process

$$X_t^{\mathbf{i}} = \sqrt{1-\widetilde{\alpha}} X_0^{\mathbf{i}} + \sqrt{\widetilde{\alpha}} \varepsilon^{\mathbf{i}}, \tag{C.28}$$

where $\mathbf{i}$ is a string of length $i+1$ encoding which triangle on the globe the pixel is in, then which 'triangular pixel' we are in. We can construct Haar wavelets on the triangle (see Figure 4.3.3), and hence create natural projection maps in L^2 over this domain. In this case W_0 is the span of constant functions over each large triangle for the tessellation of the globe, and each W_i is W_0 along with the span of the wavelets of resolution i over each triangle. Like in the standard diffusion case,

we choose $V_i = W_i$ and can use the mapping given in Section 4.5.3 to construct the natural U-Net design for this geometry, and for this basis. Similarly, for any space that we can form a triangulation for, we can use this construction to create the natural Haar wavelet space for this geometry, define a diffusion analogously, and implement the natural U-Net design for learning.

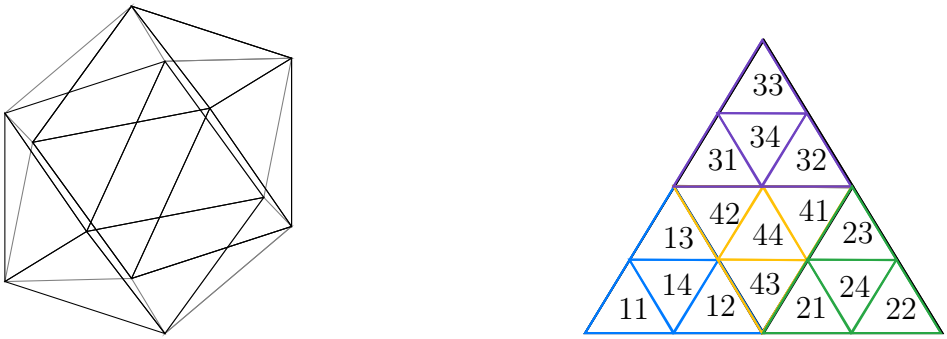


Figure C.1: Triangulation of the globe with a self-similar triangle which admits a Haar wavelet basis. Upon refinement through the self-similarity of the triangle we receive finer and finer approximations of the globe, and of functions over it.

C.1.3 U-Nets on Finite Elements

PDE surrogate modelling is a nascent research direction in machine learning research. We have seen in our PDE experiment (see Section 4.5) that our Multi-ResNet design outperforms Residual U-Nets on a square domain. We would like to mimic this in more general geometries and function spaces. Here, we will show on the unit interval how to design a U-Net over basis functions which enforce boundary constraints and smoothness assumptions, mimicking the design of Finite Elements used for PDE solvers.

Recall our model problem

$$\Delta u = f, \qquad u(0) = u(1) = 0, \tag{C.29}$$

which when multiplied by a test function $\phi \in \mathcal{H}_0^1([0,1])$ and integrated over the domain puts the equation into its weak form, and by integration by parts

$$-\langle \phi', u' \rangle_{\mathcal{H}_0^1} = \int \phi f, \tag{C.30}$$

we recover the standard bilinear representation of Laplaces equation. Now if our solution $u \in \mathcal{H}_0^1([0,1])$ and we have an orthogonal basis $\{\phi_{i,k}\}$ for $\mathcal{H}_0^1([0,1])$, then the coefficients for the basis elements of the solution can be solved by finding the solution of a linear system with a diagonal mass matrix. Thus, in this way, the information needed to solve the linear system up to some finite resolution i, only depends on data up to that resolution in this weak form. This is a Galerkin truncation of the PDE used for forward solvers on finite resolutions, but also gives us the ideal assumptions we need for the construction of our U-Net (the measurability constraint in Theorem 4.1).

For an explicit example, we may make an orthogonal basis for $\mathcal{H}_0^1([0,1])$ with the initial basis function [Left] of Figure 4.4 and its refined children on the [Right] of Figure 4.4.

In general we can refine further and create a basis of resolution i through

$$\phi_{k,j}(x) = \phi(2^k x + j/2^k), \qquad \phi(x) = 2x \cdot 1_{[0,1/2)}(x) + (2-2x) \cdot 1_{[1/2,1]}(x),$$

which are precisely the integrals of the Haar wavelets up to a given resolution. We would again construct W_i in our U-Net as the span of the first resolution i basis functions, where we now have additionally encoded the boundary constraints and smoothness requirements of our PDE solution into our U-Net design.

C.2 Additional experimental details and results

In this section, we provide further details on our experiments, and additional experimental results.

Model and training details. We used slightly varying U-Net architectures on the different tasks. While we refer to our code base for details, they generally feature a single residual block per resolution for E_i and D_i, with 2 convolutional layers, and group normalisation. We in general choose P_i as average pooling. We

further use loss functions and other architectural components which are standard for the respective tasks. This outlined U-Net architecture often required us to make several changes to the original repositories which we used. This may also explain small differences to the results that these repositories reported; however, our results are comparable. When using Algorithm 1 during staged training, we require one 'head' and 'tail' network on each resolution which processes the input and output respectively, and hence increases the number of parameters of the overall model relative to single-stage training.

Hyperparameters and hyperparameter tuning. We performed little to no hyperparameter tuning in our experiments. In particular, we did not perform a search (e.g. grid search) over hyperparameters. In general, we used the hyperparameters of the original repositories as stated in Appendix C.4, and changed them only when necessary, for instance to adjust the number of parameters so to enable a fair comparison. There is hence a lot of potential to improve the performance of our experiments, for instance of the Multi-ResNet, or the U-Net on triangular data. We refer to our code repository for specific hyperparameter choices and further details, in particular the respective `hyperparam.py` files.

Evaluation and metrics. We use the following three performance metrics in our experimental evaluation: FID score [89], rollout mean-squared-error (r-MSE) [79], and Sørensen–Dice coefficient (Dice) [57, 210]. As these are standard metrics, we only highlight key points that are worth noting. We compute the FID score on a holdout dataset not used during training, and using an evaluation model where weights are updated with the training weights using an exponential moving average (as is common practice). The r-MSE is computed as an MSE over pieces of the PDE trajectory against its ground-truth. Each piece is predicted in an autoregressive fashion, where the model receives the previous predicted piece and historic observations as input [79]. All evaluation metrics are in general computed on the test set and averaged over three random seeds after the same number of iterations in each table, if not stated otherwise. The results are furthermore not

depending on these reported evaluation metrics: in our code base, we compute and log a large range of other evaluation metrics, and we typically observe similar trends as in those reported.

C.2.1 Analysis 1: The role of the encoder in a U-Net

In this section, we provide further experimental results analysing the role of the encoder, and hence Residual U-Nets with Multi-ResNets. We realise the Multi-ResNet by replacing the parameterised encoder with a multi-level Discrete Wavelet Transform (DWT). More specifically, the skip connection on resolution j relative to the input resolution is computed by taking the lower-lower part of the DWT at level j (LL_j), and then inverting LL_j to receive a coarse, projected version of the original input image.

Generative modelling with diffusion models. In Table C.1, we analyse the role of the encoder in generative modelling with diffusion models. Following our analysis on PDE modelling and image segmentation, we compare (Haar wavelet) Residual U-Nets with (Haar wavelet) Multi-ResNets where we add the saved parameters in the encoder back into the decoder. We report performance in terms of an exponential moving average FID score [89] on the test set. In contrast to the other two tasks, we find that diffusion models benefit from a parameterised encoder. In particular, the Multi-ResNet does not outperform the Residual U-Net with approximately the same number of parameters, as is the case in the other two tasks. Referring to our theoretical results in §4.3.1, the input space of Haar wavelets may be a suboptimal choice for diffusion models, requiring an encoder learning a suitable change of basis. A second possibility is the FID score itself, which is a useful, yet flawed metric to quantify the fidelity and diversity of images [169, 200]. To underline this point, in Figure C.2 we compare samples from two runs with the Residual U-Net and Multi-ResNet as reported in Table C.1. We observe that it is very difficult to tell "by eye" which model produces higher quality samples, in spite of the differences in

FID. A third possibility is that the allocation of the saved parameters is suboptimal. We demonstrate the importance of beneficially allocating the saved parameters in the decoder in the context of PDE modelling below.

Table C.1: Quantitative performance of the (Haar wavelet) Multi-ResNet compared to a classical (Haar wavelet) Residual U-Net in generative modelling with diffusion models on `CIFAR10`. We report FID on the test set.

Dataset	Neural architecture	# Par.	FID ↓
CIFAR10 32×32	Residual U-Net	35.5 M	7.86 ± 0.25
	Multi-ResNet, no par. added in dec. (*ours*)	25.8 M	14.87 ± 0.50
	Multi-ResNet, saved par. added in dec. (*ours*)	32.4 M	12.44 ± 0.22

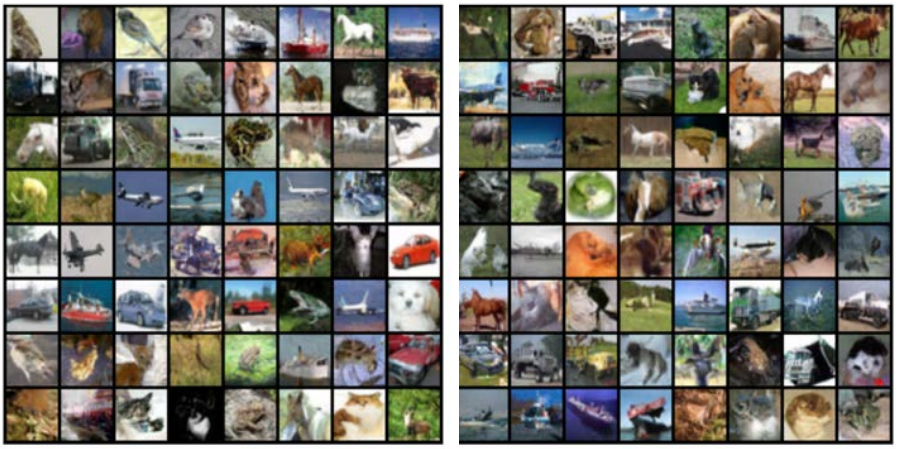


Figure C.2: Samples of a DDPM-type diffusion model with a Residual U-Net [Left] and a Multi-ResNet [Right]. We trained both models for 1.2 M iterations at which we obtained the samples.

Lastly, we present Figure 4.7 from the main text at a larger resolution in Figure C.3.

PDE modelling. We further present additional quantitative results comparing the (Haar wavelet) Multi-ResNet with (Haar wavelet) Residual U-Nets on our PDE modelling tasks. These shall highlight the importance of allocating the saved parameters in Multi-ResNets to achieve competitive performance. In Table C.2, we compare two versions of allocating the parameters we save in Multi-ResNets, which have no parameters in their encoder, on the `Navier-stokes` and `Shallow water`

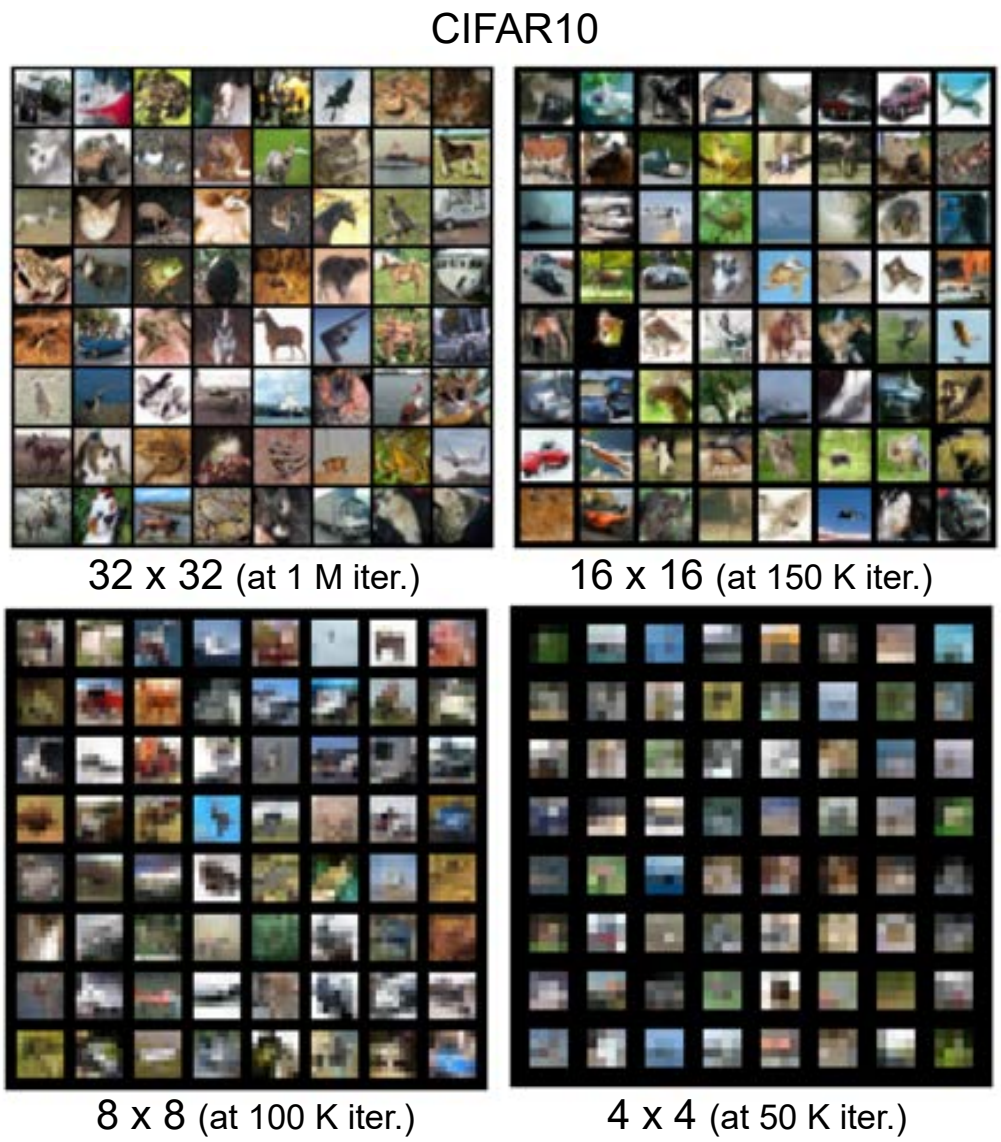


Figure C.3: Preconditioning enables multi-resolution training and sampling of diffusion models. We train a diffusion model [92] with a Residual U-Net architecture using Algorithm 1 with four training stages and no freezing on `CIFAR10`. This Figure is identical with Figure 4.7, but larger.

PDE modelling datasets, respectively. In 'v1', also presented in the main text in Table 4.1 but repeated for ease of comparison, we add the saved parameters by increasing the number of residual blocks per resolution. This in effect corresponds to more discretisation steps of the underlying ODE on each resolution [66]. In 'v2', we add the saved parameters by increasing the number of hidden channels in the ResNet.

Our results show that 'v1' performs significantly better than 'v2'. In particular, 'v2' does not outperform the Residual U-Net on the two PDE modelling tasks.

This indicates that the design choice of how to allocate parameters in $\mathcal{D}$ matters for the performance of Multi-ResNets. We note that in our experiments, beyond this comparison, we did not explore optimal ways of allocating parameters in Multi-ResNets. Hence, future work should explore this aspect thoroughly with large-scale experiments. Due to its 'uni-directional', 'asymmetric' structure, the Multi-ResNet may further be combined with a direction of transformer-based architectures which aim at outperforming and replacing the U-Net [104, 183]. With reference to our earlier presented, inferior results with diffusion models, it is possible that merely designing a better decoder would make Multi-ResNets superior to classical Residual U-Nets for diffusion models.

Table C.2: Quantitative performance of the (Haar wavelet) Multi-ResNet compared to a classical (Haar wavelet) Residual U-Net on two PDE modelling. This table is an augmented version of Table 4.1 in the main text, where 'v1' indicates the run from the main text, a Multi-ResNet with saved parameters added in the encoder, and 'v2' indicates an alternative parameter allocation. We report Mean-Squared-Error over a rolled out trajectory in time (r-MSE) on the test set, rounded to four decimal digits.

Dataset	Architecture	# Par.	r-MSE ↓
Navier-stokes 128×128	Residual U-Net	34.5 M	$0.0057 \pm 2 \cdot 10^{-5}$
	Multi-ResNet, no par. added in dec. (*ours*)	15.7 M	$0.0107 \pm 9 \cdot 10^{-5}$
	Multi-ResNet, saved par. added in dec. v1 (*ours*)	34.5 M	$\mathbf{0.0040 \pm 2 \cdot 10^{-5}}$
	Multi-ResNet, saved par. added in dec. v2 (*ours*)	34.6 M	$0.0093 \pm 5 \cdot 10^{-5}$
Shallow water 96×192	Residual U-Net	34.5 M	0.1712 ± 0.0005
	Multi-ResNet, no par. added in dec. (*ours*)	15.7 M	0.4899 ± 0.0156
	Multi-ResNet, saved par. added in dec. v1 (*ours*)	34.5 M	$\mathbf{0.1493 \pm 0.0070}$
	Multi-ResNet, saved par. added in dec. v2 (*ours*)	34.6 M	0.3811 ± 0.0091

In Figure C.4 we illustrate the full prediction of our Haar wavelet Multi-ResNet when modelling a PDE trajectory simulated from the `Navier-stokes` and `Shallow water` equations, respectively, corresponding to its truncated version in Figure 4.6 [Left]. We unroll the trajectories over five timesteps for which we predict the current

state. Note that we train the Multi-ResNet by predicting the next time step in the trajectory only. We do not condition on previous timesteps.

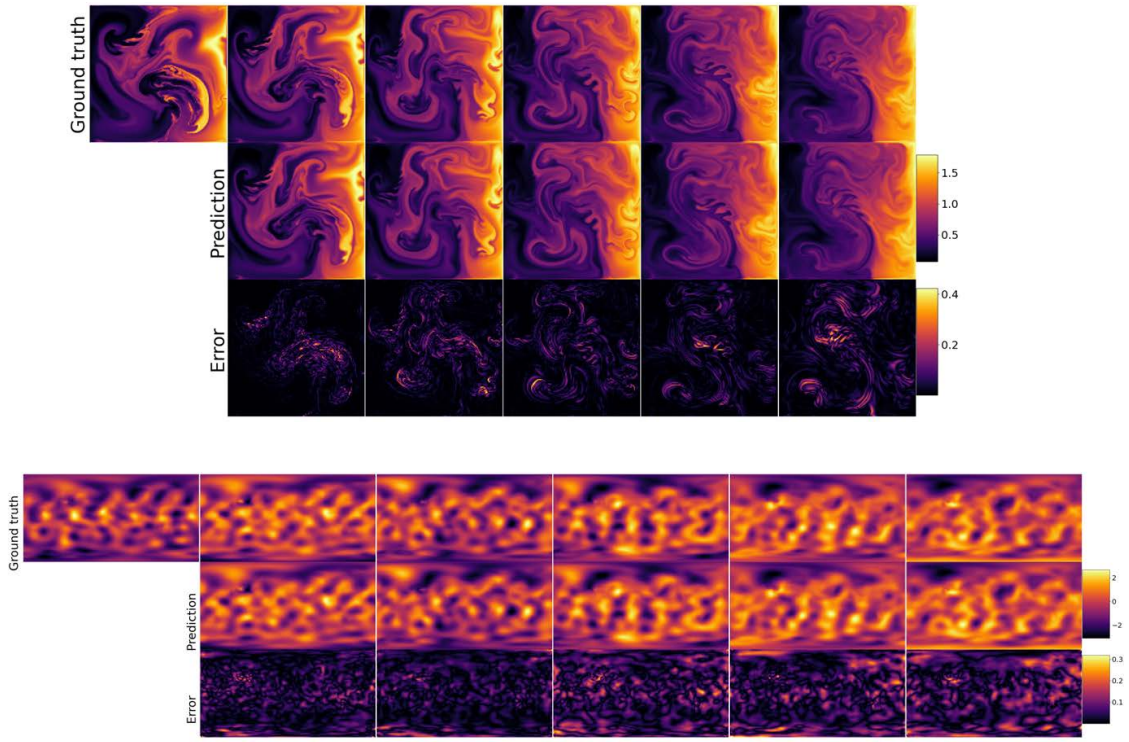


Figure C.4: PDE modelling and image segmentation with a Wavelet-encoder Multi-ResNet. Rolled out PDE trajectories (ground-trouth, prediction, L^2-error between ground-truth and prediction) from the `Navier-Stokes` [top], and the `Shallow-Water` equation [bottom]. This is the complete version of the truncated Figure 4.6 [Left] showing further timesteps for the trajectory. Figure and code as modified from [79, Figure 1].

We also present results comparing our Multi-ResNet, Residual U-Nets and the Fourier Neural Operator (FNO) [141], another competitive model for PDE modelling, in Table C.3. We compare models of similar size in terms of parameter count. We also present the same comparison in PDEArena [79] in Table C.4 for reference, noting that our experimental configuration slightly differs. Both tables indicate that U-Nets outperform FNO.

Image segmentation. In Figure C.5, we present further MRI images with overlayed ground-truth (green) and prediction (blue) masks, augmenting Figure 4.6 [Right] in the main text. The predictions are obtained from our best-performing Multi-ResNet. Note that some MRI images do not contain any ground-truth lesions.

Table C.3: Quantitative performance of an FNO model compared to U-Nets. We compare the FNO model to a (Haar wavelet) Multi-ResNets and classical (Haar wavelet) Residual U-Nets with similar number of parameters, on two PDE modelling tasks.

Dataset	Neural architecture	# Par.	r-MSE ↓
Navier-stokes 128×128	Residual U-Net	34.5 M	$0.0057 \pm 2 \cdot 10^{-5}$
	Multi-ResNet, saved par. added in dec.	34.5 M	$\mathbf{0.0040 \pm 2 \cdot 10^{-5}}$
	FNO 128-8 mode8 (*new*)	33.7 M	0.0253 ± 0.0
Shallow water 96×192	Residual U-Net	34.5 M	0.1712 ± 0.0005
	Multi-ResNet, saved par. added in dec.	34.5 M	$\mathbf{0.1493 \pm 0.0070}$
	FNO 128-8 mode8 (*new*)	33.7 M	1.2333 ± 0.0115

Table C.4: Experimental results as reported in PDEArena [79]): Quantitative performance of an FNO model, in comparison to U-Nets of similar size. Values as reported in Table 8 in [7] (5200 trajectories) for Navier-Stokes, and in [Table 2 in [7] (5600 trajectories)] for Shallow water. Here, U-Net 2015 64 clearly outperforms FNO 128-8 mode8. This result is consistent across different numbers of trajectories (dataset sizes), and different data configurations (e.g. velocity function formulation vs. vorticity stream function formulation on Shallow water) in the several tables reported in [7].

Dataset	Neural architecture	# Par.	r-MSE ↓
Navier-stokes 128×128	U-Net 2015 64	31 M	0.01386 ± 0.00004
	FNO 128-8 mode8	33.7 M	0.03836 ± 0.00037
Shallow water 96×192	U-Net 2015 64	31 M	0.1026 ± 0.0161
	FNO 128-8 mode8	33.7 M	0.8549 ± 0.0124

C.2.2 Analysis 2: Staged training enables multi-resolution training and inference

We begin by providing further experimental details. For each dataset, we train each resolution for the following number of iterations/epochs: `MNIST` $[5K, 5K, 5K, 5K]$ iterations, `CIFAR10` $[50K, 50K, 50K, 450K]$ iterations, `Navier-Stokes` $[5, 5, 5, 35]$ epochs, `Shallow water` $[2, 2, 2, 14]$ epochs.

Generative modelling with diffusion models. In Figs. C.6 and C.7, we illustrate samples of a diffusion model (DDPM) trained on `MNIST` with Algorithm 1, with and without freezing, respectively. These samples show that in both cases, the model produces reasonable samples. Algorithm 1 with freezing has the advantage that the final model can produce samples on all resolutions, instead of only the

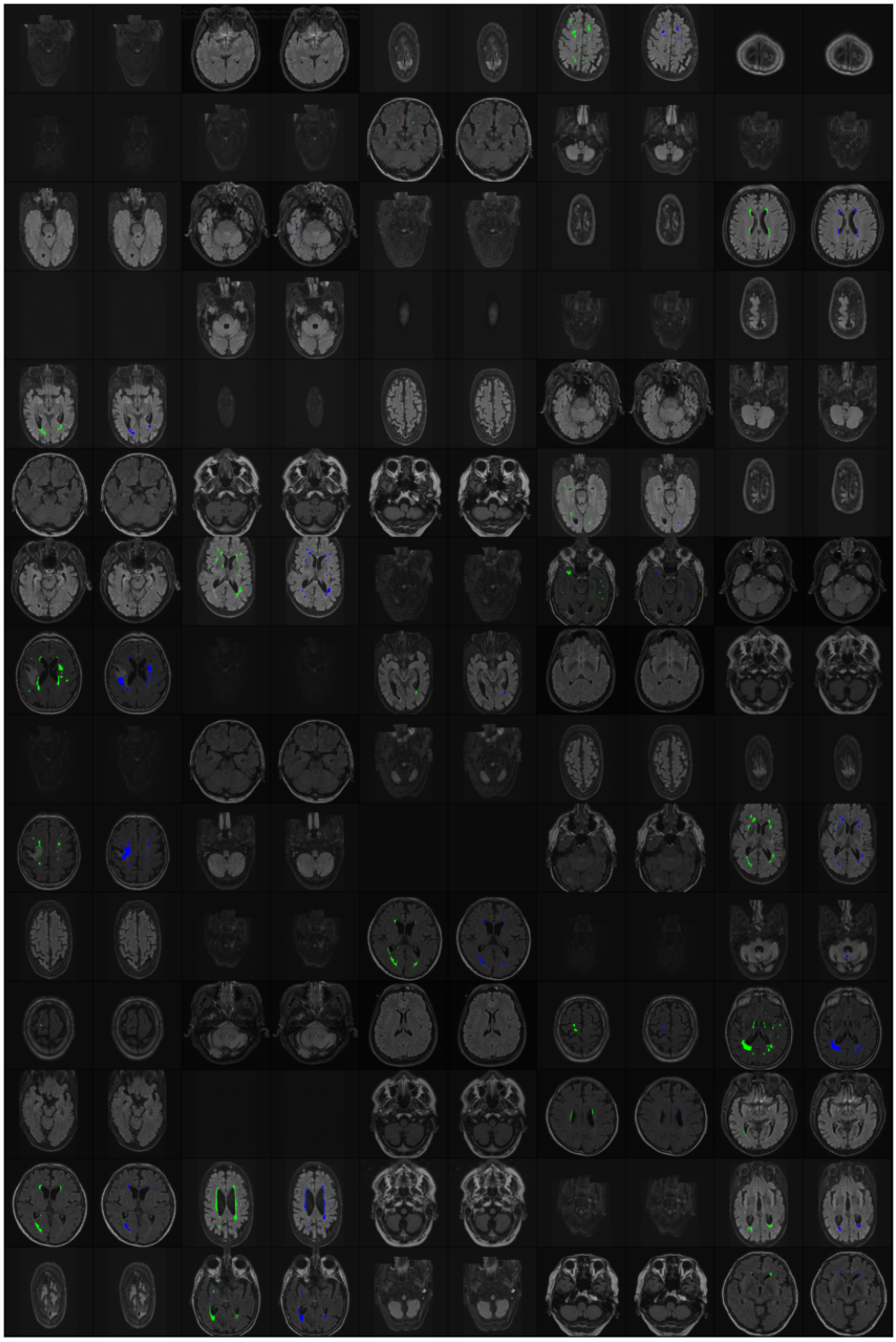


Figure C.5: MRI images from `WMH` with overlayed ground-truth masks (green) and predictions (blue) obtained from our best-performing Multi-ResNet model.

intermediate models at the end of each training stage. Preliminary tests showed that as perhaps expected, freezing lower-resolution U-Net weights tends to produce worse

performance in terms of quantitative metrics measured on the highest resolution. In particular, in Figure C.8 we compare samples and FID scores from a DDPM model trained on `CIFAR10`, with and without freezing. As can be clearly seen from both the FID score and the quality of the samples, training with Algorithm 1 but without the freezing option produces significantly better samples. This is why we did not further explore the freezing option.

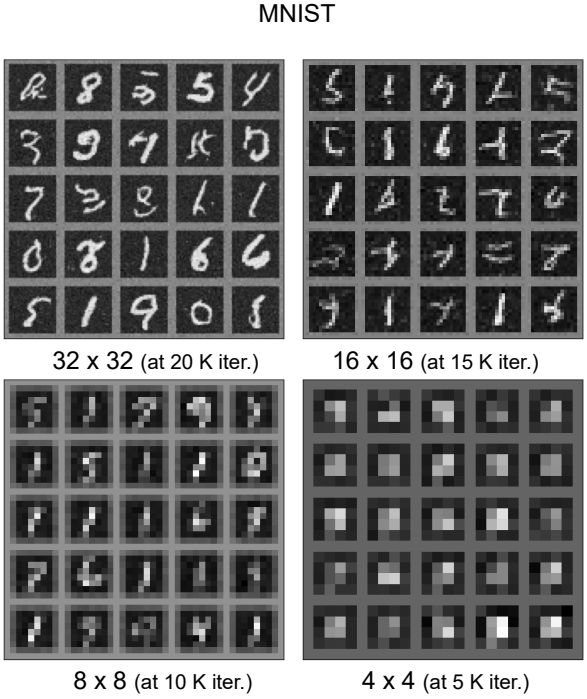


Figure C.6: Staged training of Multi-ResNets enables multi-resolution training and sampling of diffusion models. We train a diffusion model [92] with a Residual U-Net architecture using Algorithm 1 with four training stages and *with freezing* on `MNIST` corresponding to Figure 4.7. We show samples at the end of each training stage.

PDE modelling. In Table C.5, we investigate Residual U-Nets and Multi-ResNets using Algorithm 1 on `Navier-Stokes` and `Shallow water`. We here find that staged training makes the runs perform substantially worse. In particular, the standard deviation is higher, and some runs are outliers with particularly poor performance. This is in contrast to our experiments on generative modelling with diffusion models where staged training had only an insignificant impact on the resulting performance. We note that we have not further investigated this

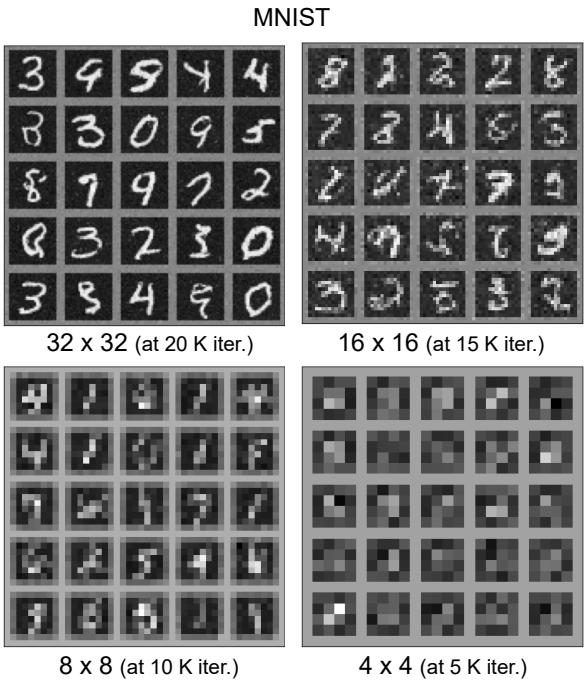


Figure C.7: Staged training of Multi-ResNets enables multi-resolution training and sampling of diffusion models. We train a diffusion model [92] with a Residual U-Net architecture using Algorithm 1 with four training stages and *no freezing* on `MNIST` corresponding to Figure 4.7. We show samples at the end of each training stage.

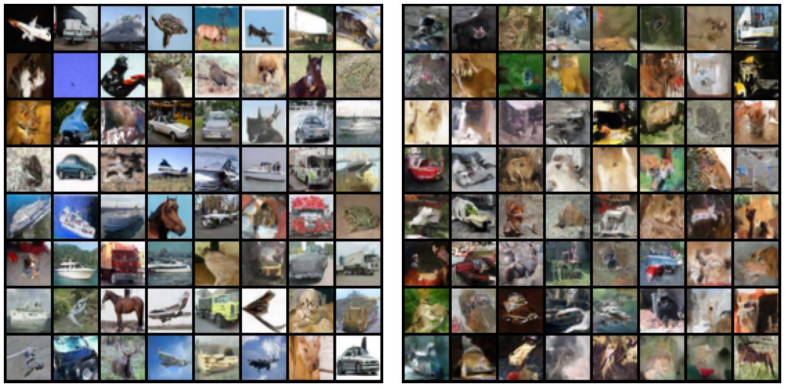


Figure C.8: A comparison of samples of a DDPM model [92] with a Residual U-Net trained with Algorithm 1 with [Right] and without [Left] freezing.

question, and believe that there are a multitude of ways which could be attempted to stabilise and improve the performance of staged training, for instance in PDE modelling, which we leave for future work.

Table C.5: Staged training with Algorithm 1 on `Navier-Stokes` and `Shallow water`. Quantitative performance of the (Haar wavelet) Multi-ResNet compared to a classical (Haar wavelet) Residual U-Net.

Dataset	Neural architecture	# Par.	r-MSE ↓
Navier-stokes 128×128	Residual U-Net	37.8 M	0.0083 ± 0.0034
	Multi-ResNet, saved par. added in dec. (*ours*)	37.8 M	0.0105 ± 0.0102
Shallow water 96×192	Residual U-Net	37.7 M	0.4640 ± 0.4545
	Multi-ResNet, saved par. added in dec. (*ours*)	37.7 M	0.7715 ± 0.8327

C.2.3 Analysis 3: U-Nets encoding topological structure

Experimental details. We begin by discussing `MNIST-Triangular`, the dataset we used in this experiment. In a nutshell, `MNIST-Triangular` is constructed by shifting the MNIST 28×28 dimensional squared digit into the lower-left half of a 64×64 squared image with similar background colour as in MNIST in that lower-left half of triangular shape. In the upper-right half, we use a gray background colour to indicate that the image is supported only on the lower-left of the squared image. We illustrate `MNIST-Triangular` in Figure C.9. We note that `MNIST-Triangular` has more than four times the number of pixels compared to `MNIST`, yet we trained our DDPM U-Net with hyperparameters and architecture for `MNIST` without hyperparameter tuning, explaining their perhaps slightly inferior sample quality in Figure 4.8.

To process the data into the subspaces $\mathcal{W} = (W_i)$ given in Section 4.3.3, we need to formulate the 'triangular pixels' shown in Figure 4.3.3. To do this we use the self-similarity property of the triangular domain to divide the data into a desired resolution. For each image, we then sample the function value at the center of our triangular pixel and store this in a lexicographical ordering corresponding to the *codespace address* [14] (e.g. '21') of that pixel. In Figure C.10 we illustrate the coding map at depth two.

Once we have made this encoding map, we are able to map our function values into an

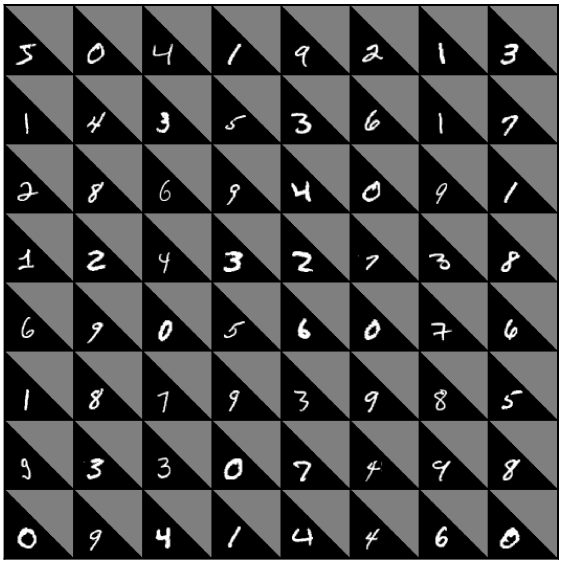


Figure C.9: Example images from the `MNIST-Triangular` dataset.

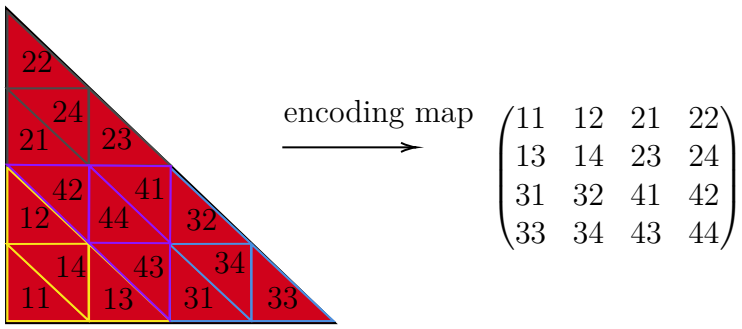


Figure C.10: The coding map from the triangular Haar wavelets to their code-space addresses. Such a construction can always be made on a self-similar object with certain separation properties, such as the *Open Set Condition* [12].

array and perform the projections on $V = W$ by push-forwarding through projection of the Haar wavlets on the triangle to our array through the lexicographic ordering used. For instance, we show in Figure C.11 the array, plotted as an image, revealing that this processing is inherently different to diffusions on a square domain.

This transformation, when defined on the infinite resolution object, is discontinuous on a dense set of $\triangle$, yet encodes all of the necessary data to perform a diffusion model over in our triangular MNIST model seen in Figure 4.8. This is because the transformation we are using is still continuous almost everywhere, and so the parts of the signal that we are losing accounts for a negligible amount of the function approximation in the diffusion process.

Figure C.11: An example of the encoding map taking data from a triangular domain and mapping it to an array of each 'triangular pixel' value under lexicographical ordering.

C.2.4 Ablation studies

In several ablation studies, we further analysed Multi-ResNets and our experimental results. We present these below.

Ablation 1: The importance of skip connections in U-Nets

We begin by presenting a key result. As Multi-ResNets perform a linear transformation in the encoder, which significantly reduces its expressivity, one might hypothesise that the skip connections in a U-Net can be removed entirely. Our results show that skip connections are crucial in U-Nets, particularly in Multi-ResNets.

In Table C.6 we compare Residual U-Nets and Multi-ResNets with and without skip connections focussing on PDE Modelling (`Navier-stokes` and `Shallow water`). The 'without skip connections' case is practically realised by feeding zero tensors along the skip connections. This has the benefit of requiring no change to the architecture enabling a fair comparison, while feeding no information via the skip connections as if they had been removed. We find that performance significantly deteriorates when using no skip connections. This effect is particularly strong in Multi-ResNets. Multi-ResNets cannot compensate the lack of skip connections by learning the encoder in such a way that representations learned on the lowest resolution space V_0 account for not having access to higher-frequency information

via the skip connections when approximating the output via D_i in the output space W_i on a higher resolution space. This result shows that the encoder actually compresses the information which it provides to the decoder. The decoder crucially depends on this compressed input.

Table C.6: Skip connections in U-Nets are crucial. We compare two Multi-ResNets, (Haar wavelet) Residual U-Nets and (Haar wavelet) Multi-ResNets, with and without skip connections, focussing on the two PDE modelling datasets (`Navier-Stokes` and `Shallow water`).

	Neural architecture	# Par.	r-MSE ↓
Navier-stokes 128×128	Residual U-Net w/ skip con.	34.5 M	$\mathbf{0.0057 \pm 2 \cdot 10^{-5}}$
	Residual U-Net w/o skip con.	34.5 M	$0.0078 \pm \cdot 10^{-5}$
	Multi-ResNet w/ skip con.	34.5 M	$\mathbf{0.0040 \pm 2 \cdot 10^{-5}}$
	Multi-ResNet w/o skip con.	34.5 M	0.0831 ± 0.1080
Shallow water 128×128	Residual U-Net w/ skip con.	34.5 M	$\mathbf{0.1712 \pm 0.0005}$
	Residual U-Net w/o skip con.	34.5 M	0.4950 ± 0.02384
	Multi-ResNet w/ skip con.	34.5 M	$\mathbf{0.1493 \pm 0.0070}$
	Multi-ResNet w/o skip con.	34.5 M	0.6302 ± 0.01025

Ablation 2: The importance of preconditioning in U-Nets

In this section, we are interested in analysing the importance of preconditioning across subspaces in U-Nets. To this end, we analyse whether and how much a U-Net benefits from the dependency in the form of preconditioning between its input and output spaces $\mathcal{V}$ and $\mathcal{W}$. In Table C.7, we compare a Residual U-Net with multiple subspaces with a ResNet with only one subspace on `Navier-Stokes` and `Shallow water`, respectively. We obtain the ResNet on a single subspace from the Residual U-Net by replacing P_i as well as the (implicit) embedding operations with identity operators, and additionally feed zeros across the skip connections. This is because the function of the skip connections is superfluous due to them not being able to feed a compressed representation of the input. We note that the performance of the ResNet can potentially be improved by allocating the number of channels different to the increasing and decreasing choice in a U-Net, but we have not explored this. We further train on `Shallow water` for 10 epochs evaluating on

the test set, and on `Navier-Stokes` for approximately 400 K iterations evaluating on the validation set. This is due to significantly longer running times of the ResNets constructed as outlined above. We note that performance decreases very slowly after this point, and the trend of this experiment is unambiguously clear, not requiring extra training time.

The results in Table C.7 show that preconditioning via the U-Net's self-similarity structure is a key reason for their empirical success. The Residual U-Net on multiple subspaces outperforms the ResNet on a single subspace by a large margin. This also justifies the focus of studying preconditioning in this work.

Table C.7: On the effect of subspace preconditioning vs. a plain ResNet. Quantitative performance of the (Haar Wavelet) Residual U-Net over multiple input and output subspaces compared to a ResNet on a single subspace, both trained on `Navier-Stokes` and `Shallow water`.

Dataset	Neural architecture	# Par.	r-MSE ↓
Navier-stokes 128×128	Residual U-Net (multiple subspaces)	34.5 M	$\mathbf{0.0086 \pm 9.7 \cdot 10^{-5}}$
	ResNet (one subspace)	34.5 M	0.3192 ± 0.0731
Shallow water 96×192	Residual U-Net (multiple subspaces)	34.5 M	$\mathbf{0.3454 \pm 0.02402}$
	ResNet (one subspace)	34.5 M	2.9443 ± 0.2613

Ablation 3: On the importance of the wavelet transform in Multi-ResNets

In Table C.8 we analyse the effect of different wavelets for the DWT we compute in P_i. We selected examples from different wavelet families, all of them being orthogonal. A good overview of different wavelets which can be straight-forwardly used to compute DWT can be found on this URL: `https://wavelets.pybytes.com/`. On `Navier-Stokes` we analyse the effect of choosing Daubechies 2 and 10 wavelets systematically over different random seeds. On `Shallow water` we explore many different Wavelet families generally without computing random seeds. We find that the Wavelets we choose for P_i have some, but rather little impact on model performance. We lastly note that the downsampling operation has also been studied in [95] in the context of score-based diffusion models.

Table C.8: On the importance of choosing different wavelets for the Discrete Wavelet Transforms (DWT) in Multi-ResNets. We report quantitative performance on the test set of `Navier-Stokes` and `Shallow water`.

Dataset	Neural architecture	# Par.	r-MSE ↓
Navier-stokes 128×128	Multi-ResNet w/ Haar wavelet DWT	34.5 M	$\mathbf{0.0040 \pm 2 \cdot 10^{-5}}$
	Multi-ResNet w/ Daubechies 2 DWT	34.5 M	$0.0041 \pm 3 \cdot 10^{-5}$
	Multi-ResNet w/ Daubechies 10 DWT	34.5 M	$0.0068 \pm 1.1 \cdot 10^{-4}$
Shallow water 96×192	Multi-ResNet w/ Haar wavelet DWT	34.5 M	0.1493 ± 0.0070
	Multi-ResNet w/ Daubechies 2 DWT	34.5 M	**0.1081**
	Multi-ResNet w/ Daubechies 10 DWT	34.5 M	0.1472
	Multi-ResNet w/ Coiflets 1 DWT	34.5 M	0.1305
	Multi-ResNet w/ Discrete Meyer DWT	34.5 M	0.1237

C.2.5 On the importance of preconditioning in residual learning: Synthetic experiment

We begin by providing a second example with $w(v) = v^3$ in Figure C.12, corresponding to Figure 4.2 in §4.1. Note that in comparison to the problem $w(v) = v^2$ in Figure 4.2, the effect of the two preconditioners swaps in Figure C.12: Using $R^{\text{pre}}(v) = v$ as the preconditioner produces an almost perfect approximation of the ground-truth function, while choosing $R^{\text{pre}}(v) = |v|$ results in deteriorate performance. This is because $R^{\text{pre}}(v) = v$ is a 'good' initial guess for $w(v) = v^3$, but a very 'poor' guess for $w(x) = v^2$, and vice versa for $R^{\text{pre}}(v) = |v|$. This illustrates that the choice of using a good preconditioner is *problem-dependent*, and can be crucial to solve an optimisation problem at hand.

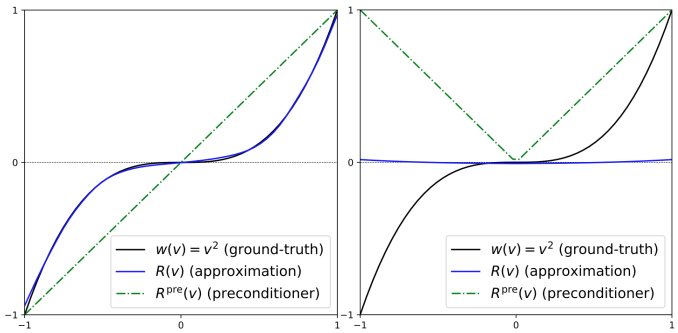


Figure C.12: The importance of *preconditioning*. We learn the ground-truth functions $w(v) = \{v^3\}$ using a ResNet $R^{\text{res}}(v) = R^{\text{pre}}_{\ell}(v) + R(v)$ with preconditioners [Left] $R^{\text{pre}}_1(v) = v$ and [Right] $R^{\text{pre}}_2(v) = |v|$.

Having discussed the results of our synthetic experiment (see Figs. 4.2 and C.12) which demonstrates the importance of preconditioning, we here provide further details on the experimental setting. We train a ResNet $R^{\text{res}}(v) = R_\ell^{\text{pre}}(v) + R(v)$ over a grid of values $\{(v_i, w_i)\}_{i=1}^N$ where $v_i \in [-1, 1]$, $w_i = w^{(k)}(v_i)$, $N = 50$, and $w^{(1)}(v) = v^2, w^{(2)}(v) = v^3$, $R_\ell^{\text{pre}}(v) = v, R_\ell^{\text{pre}}(v) = |v|$ depending on the experiment scenario indicated by the superscripts (k, ℓ), respectively. We do not worry about generalisation in this example and use the same train and test distribution. This functional relationship is easy to learn by any neural network R of sufficient size, which would overshadow the effect of preconditioning. To construct a ResNet which is 'just expressive enough' to learn $w^{(k)}$, we constrain the expressivity of R in two ways: First, we choose R to be a small fully-connected network with 3 linear layers (input, hidden, output) with $[1, 20, 1]$ neurons, respectively, interleaved with the ReLU activation function. Second, inspired by invertible ResNets [15], we normalize the weights θ_j of R such that $\|\theta_j\| < 1$ for all j where $\|\cdot\|$ is the Frobenius norm, and repeatedly apply R with $D = 100$ times via weight-sharing. In practice, $\|\theta_j\|$ will be close to 1, and as $\|\theta_j\|$ intuitively measures the 'step size' of a neural network, we constrain the neural network to '100 steps of unit length'. As an alternative, one could limit the maximum number of training steps, which would indirectly constrain the model's expressiveness. We note that our reported results are robust across a variety of hyperparameter combinations leading to qualitatively the same conclusions.

C.3 Background

C.3.1 Related work

U-Nets. The U-Net [197] is a go-to, state-of-the-art architecture across various research domains and tasks. Among the most influential U-Net papers are U-Net++ [263] and Attention U-Net [175] for image segmentation, 3D U-Net [37] for volumetric segmentation, U-Nets in diffusion models, starting with the first

high-resolution demonstration in DDPM [92], the nnU-Net [102], a U-Net which automatises parts of its design, a probabilistic U-Net [124], conditional U-Nets [64], U-Nets for cell analysis detection and counting [69], and U-Nets for road extraction [257]. A large number of adaptations and variants of the U-Net exist. We refer to [204] for an overview of such variants in the context of image segmentation for which the U-Net was first invented. In particular, many U-Net papers use a ResNet architecture. [101] find that a key improvement for the seminal U-Net [197] is to use residual blocks instead of feed-forward convolutional layers. Beyond their use on data over a squared domain, there exist custom implementations on U-Nets for other data types, for instance on the sphere [258], on graphs [71, 148] or on more general, differentiable 3D geometries [88]. However, we note that a framework which unifies their designs and details the components for designing U-Nets on complicated data types and geometries is lacking. This paper provides such a framework.

Our work directly builds on and is motivated by the paper by Falck & Williams [66] which first connected U-Nets and multi-resolution analysis [47]. This paper showed the link between U-Nets and wavelet approximation spaces, specifically the conjugacy of Haar wavelet projection and average pooling in the context of U-Nets, which our work crucially relies on. The design of U-Nets and their connection to wavelets has also been studied in [144, 191]. Falck & Williams further analysed the regularisation properties of the U-Net bottleneck under specific assumptions restricted to the analysis of auto-encoders without skip connections, and argued by recursion what additional information is carried across each skip without this being rigorously defined. Our work in this paper is however augmenting their work in various ways. We list five notable extensions: First, we provide a unified definition of U-Nets which goes beyond and encompasses the definition in [66], and highlights the key importance of preconditioning in U-Nets as well as their self-similarity structure. Importantly, this definition is not limited to orthogonal wavelet spaces as choices for $\mathcal{V}$ and $\mathcal{W}$, hence enabling the use of U-Nets for a much broader set of domains and tasks. Second, based on this definition, our framework enables the

design of novel architectures given a more thorough understanding of the various components in a U-Net. This is demonstrated on the elliptic PDE problem and data over a triangular domain. Third, we analyse the usefulness of the inductive bias of U-Nets in diffusion models, a novel contribution. Fourth, our experiments with Multi-ResNets and multi-resolution training and sampling, as well as on triangular data are novel. Fifth, [66] focusses particularly on the application of U-Nets in hierarchical VAEs, which this work is not interested in in particular. In summary, while [66] provided crucial components and ideas in U-Nets, our work is focused on a framework for designing and analysing general U-Nets, which enables a wide set of applications across various domains.

Miscellaneous. We further briefly discuss various unrelated works which use similar concepts as those in this paper. Preconditioning, initialising a problem with prior information so to facilitate solving it, is a widely used concept in optimisation across various domains [5, 6, 16, 30, 219, 233]. In particular, we refer to its use in multi-level preconditioning and Garlekin methods [8, 9, 45]. Preconditioning is also used in the context of score-based diffusion models [253]. Most notably, preconditioning is a key motivation in the seminal paper on ResNets [87]. While preconditioning is a loosely defined term, we use it in the context of this literature and its usage there.

The concept of 'dimensionality reduction' is widely popular in diffusion models beyond its use in U-Nets. For instance, Stable Diffusion and Simple Diffusion both perform a diffusion in a lower-dimensional latent space and experience performance gains [95, 196]. Stable Diffusion in particular found that their model learns a "low-dimensional latent space in which high-frequency, imperceptible details are abstracted away" [196]. It is this intuition that the U-Net formalises. Simple Diffusion also features a multi-scale loss resembling the staged training in Algorithm 1. Another paper worth pointing out learns score-based diffusion models over wavelet coefficients, demonstrating that wavelets and their analysis can be highly useful in diffusion models [80].

C.3.2 Hilbert spaces

A Hilbert space is a vector space W endowed with an inner-product $\langle \cdot, \cdot \rangle$ that also induces a complete norm $\| \cdot \|$ via $\|w\|^2 = \langle w, w \rangle$. Due to the inner-product, we have a notion of two vectors $w_1, w_2 \in W$ being orthogonal if $\langle w_1, w_2 \rangle = 0$.

In Section 4.3 we have paid special attention to two specific Hilbert spaces: $L^2([0,1])$ and $\mathcal{H}_0^1([0,1])$.

1. the space $L^2([0,1])$ has as elements square-integrable functions and has an inner-product given by

$$\langle f, g \rangle_{L^2} = \int_0^1 f(x)g(x)dx; \text{ and,} \tag{C.31}$$

2. the space $\mathcal{H}_0^1([0,1])$ has as elements once weakly differentiable functions which vanish at both zero and one, with inner-product given by

$$\langle f, g \rangle_{\mathcal{H}_0^1} = \int_0^1 f'(x)g'(x)dx, \tag{C.32}$$

   where $f', g' \in L^2([0,1])$ are the weak derivatives of f and g respectively.

If we have a sequence $\{\phi_k\}_{k=0}^{\infty}$ of elements of W which are pairwise orthogonal and span W, then we call this an orthogonal basis for W. Examples of orthogonal bases for both $L^2([0,1])$ and $\mathcal{H}_0^1([0,1])$ are given in Section C.3.3.

C.3.3 Introduction to Wavelets

Wavelets are refinable basis functions for $L^2([0,1])$ which obey a self-similarity property in their construction. There is a 'mother' and 'father' wavelet ϕ and ψ, of which the children wavelets are derivative of the mother wavelet ϕ. For

instance, in the case of Haar wavelets with domain $[0,1]$ we have that $\psi(x) = 1$ and the mother and children wavelets given by

$$\phi_{k,j}(x) = \phi(2^k x + j/2^k), \qquad \phi(x) = 1_{[0,1/2)}(x) - 1_{[1/2,1]}(x).$$

Under the L^2-inner product, the family $\{\phi_{k,j}\}_{j=0,k=1}^{2^j,\infty}$ forms an orthogonal basis of $L^2([0,1])$. Further, if we define the functions

$$\tilde{\phi}_{k,j}(x) = \int_0^x \phi_{k,j}(y)dy, \tag{C.33}$$

then each of these functions is in $\mathcal{H}_0^1([0,1])$, and is further orthogonal with respect to the $\mathcal{H}_0^1$-inner-product, bootstrapped from the orthogonality of the Haar wavelets.

The discrete encoding of the Haar basis can be given by the kronecker product

$$H_i = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes H_{i-1}, \qquad H_0 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{C.34}$$

To move between the pixel basis and the Haar basis natural to average pooling, we use the mapping defined on resolution i by $T_i : V_i \mapsto V_i$ through

$$T_i(v_i) = \Lambda_i H_i v_i, \tag{C.35}$$

where H_i is the Haar matrix above and Λ_i is a diagonal scaling matrix with entries

$$(\Lambda_i)_{k,k} = 2^{-i+j-1}, \quad \text{if } k \in \{2^{j-1}, \ldots, 2^j\}. \tag{C.36}$$

To get this, we simply identify how to represent a piecewise constant function from its pixel by pixel function values to be the coefficients of the Haar basis functions. For example, in one dimension for an image with four pixels we can describe the function as the weighted sum of the four basis functions [Right], that take values ± 1 or zero. The initial father wavelet is set to be the average of the four pixel values, and the coefficients of the mother and children wavelets are chosen to be the local deviance's of the averages from these function values, as seen in Figure C.13.

For further details on wavelets, we refer to textbooks on this topic [47, 153].

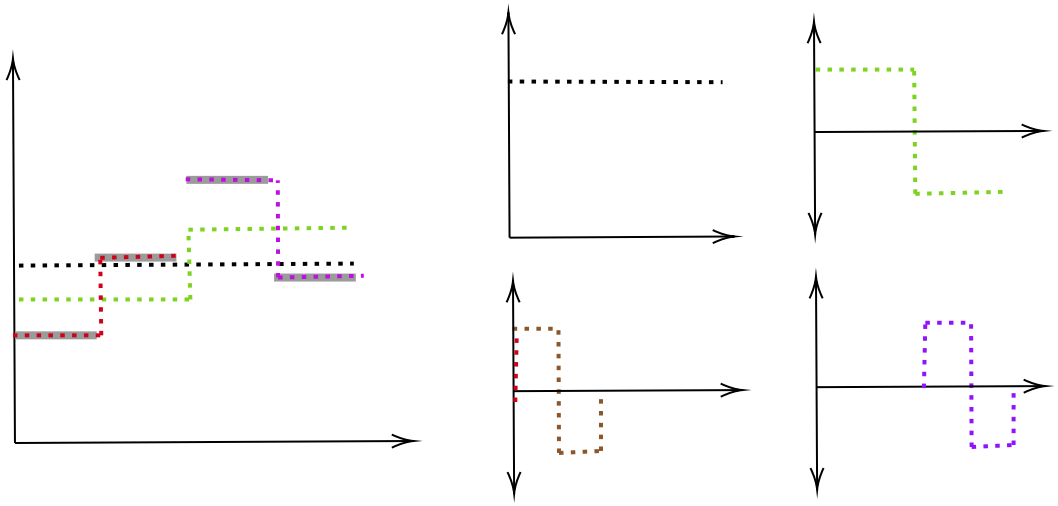


Figure C.13: Modelling a 1D image with four pixel values as the weighted sum of Haar wavelet frequencies. The coefficients are such that the local averages of pixel values give the Haar wavelet at a lower frequency, hence average pooling is conjugate to basis truncation here.

C.3.4 Images are functions

An image, here a gray-scale image with squared support, can be viewed as the graph of a function over the unit square $[0,1]^2$ [66]. We visualise this idea in C.14, referring to [66, Section 2] and [62] for a more detailed introduction. Many other signals can likewise be viewed as It is hence natural to construct our U-Net framework on function spaces, and have E_i and D_i be operators on functions.

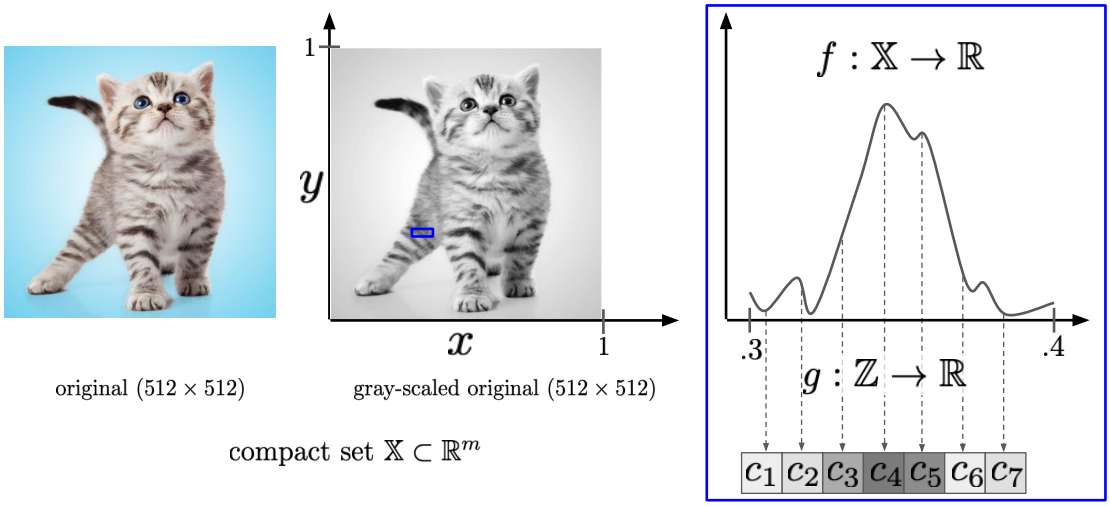


Figure C.14: Images modelled as functions.

C.4 Code, computational resources, datasets, existing assets used

Code. We provide our code base as well as instructions to reproduce the key results in this paper under MIT License at `https://github.com/FabianFalck/unet-design`. Our code base uses code from four Github repositories which are subfolders. For our diffusion experiments on `MNIST` and `MNIST-Triangular`, we directly build on top of the repository `https://github.com/JTT94/torch_ddpm`. For our diffusion experiments on `CIFAR10`, we directly build on top of the repository `https://github.com/w86763777/pytorch-ddpm`. For our PDE experiments on `Navier-Stokes` and `Shallow water`, we use the repository `https://github.com/microsoft/pdearena` [79]. For our image semgentation experiments on `WMH`, we are inspired by the repository `https://github.com/hongweilibran/wmh_ibbmTum` [137], but write the majority of code ourselves. We note that the MIT License only pertains to the original code in our repository, and we refer to these four repositories for their respective licenses.

We extended each of these repositories in various ways. We list key contributions below:

- We implemented several Residual U-Net architectures in the different repositories.
- We implemented Multi-ResNets in each repository.
- We implemented the staged training Algorithm (see Algorithm 1), as well as its strict version which freezes parameters.
- We implemented logging with weights & biases, as well miscellaneous adjustments for conveniently running code, for instance with different hyperparameters from the command line.

Datasets. In our experiments, we use the following five datasets: `MNIST` [133], `MNIST-Triangular`, a custom version of MNIST where data is supported on a triangle rather than a square, `CIFAR10` [127], `Navier-stokes`, `Shallow water` [79], and the MICCAI 2017 White Matter Hyperintensity (`WMH`) segmentation challenge dataset [130, 137]. These five datasets—with the exception of `MNIST-Triangular`—have in common that data is presented over a square or rectangular domain, possibly with several channels, and of varying resolutions. We refer to the respective repositories above where these datasets have already been implemented. For `MNIST-Triangular`, we provide our custom implementation and dataset class as part of our code base, referring to Appendix C.2.3 on how it is constructed. It is also worth noting that `Navier-Stokes` and `Shallow Water` require considerable storage. On our hardware, the two datasets take up approximately 120 GB and 150 GB unzipped, respectively.

Computational resources. This work made use of two computational resources. First, we used two local machines with latest CPU hardware and one with an onboard GPU for development and debugging purposes. Second, we had access to a large compute cluster with A100 GPU nodes and appropriate CPU and RAM hardware. This cluster was shared with a large number of other users.

To reproduce a single run (without error bars) in any of the experiments, we provide approximate run times for each dataset using the GPU resources: On `MNIST` and `MNIST-Triangular`, a single run takes about 30 mins. On `CIFAR10`, a single run takes several days. On `Navier-Stokes` and `Shallow water`, a single run takes approximately 1.5 days and 1 day, respectively.

Existing assets used. Our code base uses the following main existing assets: Weights&Biases [19] (MIT License), PyTorch [180] (custom license), in particular the `torchvision` package, `pytorch_wavelets` [43] (MIT License), `PyWavelets` [135] (MIT License), `pytorch_lightning` [68] (Apache License 2.0), `matplotlib` [99] (PSF License), `numpy` [85] (BSD 3-Clause License), `tensorboard` [157] (Apache

License 2.0), `PyYaml` [40] (MIT License), `tqdm` [41] (MPLv2.0 MIT License), `scikit-learn` and `sklearn` [181] (BSD 3-Clause License), and `pickle` [225] (License N/A). We further use Github Copilot for the development of code, and in few cases use ChatGPT [176] as a writing aid.

D

Appendix of *Is In-Context Learning in Large Language Models Bayesian? A Martingale Perspective*

Contents

D.1 Proofs of Theoretical Statements in the Main Text

Fact D.1. Any exchangeable random sequence $\{Z_i\}$ must be conditionally identically distributed.

Proof. See, e.g., [17, p. 2030]. □

Proposition 5.1. A sequence $\{Z_{n+1:n+m}\} \sim p_M(\cdot|Z_{1:n})$ satisfies the martingale property if and only if the following holds: for all $n', k \in \mathbb{N}$ and integrable functions g, h:

$$\mathbb{E}((g(Z_{n'+k}) - g(Z_{n'+1}))h(Z_{n+1:n'})|Z_{1:n}) = 0. \tag{5.8}$$

Proof. It suffices to show the equivalence between the following three statements:

(i) $Z_{n+1:n+m}|Z_{1:n}$ satisfies Eq. (5.1)

(ii) for all $n' \geq n, k \geq 1$ and integrable function g we have $\mathbb{E}(g(Z_{n'+k}) - g(Z_{n'+1})|Z_{1:n}, Z_{n+1:n'}) = 0$

(iii) for all $n' \geq n, k \geq 1$ and integrable (g, h) we have $0 = \mathbb{E}((g(Z_{n'+k}) - g(Z_{n'+1}))h(Z_{n+1:n'})|Z_{1:n})$.

The equivalence between (i) and (ii) is trivial. We have (ii) $\Rightarrow$ (iii) because $\mathbb{E}((g(Z_{n'+k})-g(Z_{n'+1}))h(Z_{n+1:n'}) \mid Z_{1:n}) = \mathbb{E}(\mathbb{E}(g(Z_{n'+k})-g(Z_{n'+1}) \mid Z_{1:n'})h(Z_{n+1:n'}) \mid Z_{1:n}) \overset{(ii)}{=} 0$. To show (iii) $\Rightarrow$ (ii), for any $\sigma(Z_{n+1:n'})$-measurable set A let $h := \mathbb{1}_A$ be the respective indicator function, so that $\mathbb{E}((g(Z_{n'+k}) - g(Z_{n'+1}))\mathbb{1}_A \mid Z_{1:n}) \overset{(iii)}{=} 0 = \mathbb{E}(0 \cdot \mathbb{1}_A \mid Z_{1:n})$. Since this holds for all A, it follows by the definition of conditional expectation [114] that $\mathbb{E}(g(Z_{n'+k}) - g(Z_{n'+1}) \mid Z_{1:n'}) = 0$ a.s.. □

Corollary 5.1. Let $\{Z_i : i \in \mathbb{N}\}$ be a sequence of random variables satisfying the martingale property. Then for all integers $n, n', k > 0$ and $n' > n$ it holds that:

(i) $\mathbb{E}(g(Z_{n+1})|Z_{1:n}) = \mathbb{E}(g(Z_{n+k})|Z_{1:n})$ for all integrable functions g, and

(ii) $\mathbb{E}((Z_{n'+k+1} - Z_{n'+1})Z_{n'}^\top|Z_{1:n}) = 0$.

Proof. (i) follows by setting $h(z_{n+1:n'}) \equiv \mathbb{1}$ in (5.8). (ii) follows by setting $g(z) = z, h(z_{n+1:n'}) = z_{n'}$. □

Fact 5.1. Let $\pi(\theta)$ and $p_M(Z|\theta)$ be the prior and likelihood of a Bayesian model, $\bar{\theta}_n := \mathbb{E}_{\theta\sim\pi(\theta|z_{1:n})}\theta$ the posterior mean given data $z_{1:n}$, and $\|\cdot\|$ be any vector norm. Then,

$$\mathbb{E}_{\theta_0\sim\pi, z_{1:n}\sim\pi(z|\theta_0)}\mathbb{E}_{\theta\sim\pi(\theta|z_{1:n})}\|\theta - \bar{\theta}_n\|^2 = \mathbb{E}_{\theta_0\sim\pi, z_{1:n}\sim\pi(z|\theta_0)}\|\theta_0 - \bar{\theta}_n\|^2. \tag{5.9}$$

Proof. This holds because θ and θ_0 are conditionally independent and identically distributed given $z_{1:n}$, and $\bar{\theta}_n$ equals the conditional expectation of both random variables. □

D.2 Further Discussion of Theory and Methodology

D.2.1 Additional Background on Martingale Posteriors

In §5.2.3 we discussed the construction of martingale posteriors in the finite-support case. Here, we can construct the martingale posterior by sampling $Z_{n+1:n+m}|Z_{1:n}$, which will determine a sample $\theta_{n+m}|Z_{1:n}$ as the parameter that indexes the predictive distribution $p(Z_{n+m+1} = \cdot|Z_{1:n+m}) = p_{\theta_{n+m}}(\cdot)$; and since $\theta_{n+m} \to \theta_\infty$ as $m \to \infty$, we can truncate the process at a large $m \gg n$ to obtain a good approximation for θ_∞.

The restriction to finite support is largely for expository simplicity as it allows us to avoid measure-theoretic considerations. More generally, it is always possible to view the distribution $p(Z_{n+1} = \cdot|Z_{1:n}) =: \theta_n$ as a random element in a suitable Banach space of measures and the condition in Eq. (5.1) as requiring $\{p(Z_{n+1} =$

$\cdot|Z_{1:n}) : n \in \mathbb{N}\}$ to define a martingale in that space. When Doob's theorem applies, the above construction provides a distribution over predictive distributions that quantifies the epistemic uncertainty.

Nonetheless, for tractability and comparability to Bayesian parametric posteriors, it is useful to consider the following alternative, 'model-based' procedure:

1. Sample $Z_{n+1:n+m} \sim p_M(\cdot|Z_{1:n})$.
2. Compute $\hat{\theta}_m := \arg\max_{\theta\in\Theta} \sum_{j=1}^{m} \log p(Z_{n+j}|\theta)$.
3. Return $\hat{\theta}_m$ as an approximate sample from the martingale posterior, defined as the conditional distribution of the pointwise limit $\lim_{m\to\infty} \hat{\theta}_m$ given $Z_{1:n}$.

We repeat this procedure to obtain multiple samples $\hat{\theta}_m$ from the martingale posterior in order to approximate its distribution (see Fig. 5.1 [Centre]). In the above, $p(Z_i|\theta)$ is the likelihood in the Bayesian parametric model. If $\{p_M(Z_{n+j}|Z_{1:n+j-1})\}_{j=1}^{\infty}$ corresponds to a certain posterior predictive defined by the same likelihood, and the model is such that maximum likelihood estimation is consistent, it follows from de Finetti's theorem (applied to $Z_{n+1:}|Z_{1:n}$) and consistency that as $m \to \infty$, $\hat{\theta}_m$ will converge to a random variable $\hat{\theta}_\infty$ (w.r.t. the norm and notion of convergence in consistency), and the distribution $\hat{\theta}_\infty|Z_{1:n}$ must equal the Bayesian posterior. Applying the same procedure to a more general p_M that satisfies Eq. (5.1) leads to the methodology in [70].

We adopted this 'model-based' approach in §5.3.3 and for computing the approximate martingale posterior in Fig. 5.1 [centre]. Compared with the former approach, it is easier to implement on ICL tasks where each sample Z_i is represented with multiple tokens and a correctly specified likelihood for the true observations is available; the latter is always true in our synthetic experiments. More importantly, when m is finite (and not $\gg n$), only with this approach can we compare the sampling distribution of $\hat{\theta}_m|Z_{1:n}$ across different p_M, as we explain in the following.

This is important in our experiments where we find the LLMs (at best) follow the martingale property within a horizon of $m = \Theta(n)$.

D.2.2 Approximate Martingale Posteriors with Finite Paths

We have claimed that with a finite m, the spread of the approximate martingale posterior $\hat{\theta}_m$ defined as the MLE on m samples (see §5.3.3, or above) is comparable between different choices of p_M. We now substantiate on this claim.

Let us first restrict to exchangeable (i.e., Bayesian) choices of p_M. Consider de Finetti's representation for the posterior predictive measure: $Z_{n+1,\dots}|Z_{1:n}$ can be represented through

$$\theta_\infty \sim \pi(\cdot|Z_{1:n}), \quad Z_{n+1,\dots} \overset{iid}{\sim} p(\cdot|\theta_\infty)$$

where the measure $\pi(\cdot|Z_{1:n})$ equals the Bayesian posterior, which as discussed in §D.2.1 equals the exact martingale posterior. Combining the above representation and the fact that $\hat{\theta}_m$ is a function of $Z_{n+1:n+m}$ leads to $\hat{\theta}_m \perp Z_{1:n}|\theta_\infty$, and

$$\begin{aligned}
&\mathrm{Cov}(\hat{\theta}_m|Z_{1:n}) \\
&= \mathbb{E}(\mathrm{Cov}(\hat{\theta}_m|\theta_\infty)|Z_{1:n}) + \mathrm{Cov}(\mathbb{E}(\hat{\theta}_m|\theta_\infty)|Z_{1:n}) \\
&\approx \mathbb{E}(\mathrm{Cov}(\hat{\theta}_m|\theta_\infty)|Z_{1:n}) + \mathrm{Cov}(\theta_\infty|Z_{1:n}),
\end{aligned}$$

where we dropped the term $\mathbb{E}(\hat{\theta}_m|\theta_\infty) - \theta_\infty$ which is the bias of MLE and thus a higher-order term for regular models. Therefore, the (co)variance overhead $\mathrm{Cov}(\hat{\theta}_m|Z_{1:n}) - \mathrm{Cov}(\theta_\infty|Z_{1:n})$ is, up to the first order, the average-case error of MLE on m i.i.d. samples when the true parameter is sampled from the posterior $\pi(\cdot|Z_{1:n})$. For regular models this is always $\Theta(d/m)$, where the coefficient hidden in the Θ notation is also comparable across different p_M as long as the Fisher information matrix evaluated at $\theta \sim \pi(\cdot|Z_{1:n})$ has a comparable value (e.g., across all choices of p_M that satisfy *consistency*). As the martingale posterior covariance $\mathrm{Cov}(\theta_\infty|Z_{1:n})$ has the same $\Theta(d/n)$ scaling across all regular Bayesian models to

which the Bernstein von-Mises theorem applies, with a choice of $m = \Theta(n)$, any deviation in the scaling of $\mathrm{Cov}(\hat{\theta}_m)$—from that of any regular Bayesian model—must be attributable to a different scaling of the exact MP covariance, and thus a deviation from all regular Bayesian models.

Lastly, we note that while we focus on ICL models that are approximately Bayesian, the above discussion may also apply to general models that only satisfy the martingale property, since for those models $Z_{n+1,\ldots}|Z_{1:n}$ remains asymptotically exchangeable [17]. Moreover, the above discussion applies to inter-quartile range (IQR) as well, because for asymptotically normal posteriors the IQR is proportional to the posterior standard deviation; and even for non-normal posteriors, the IQR should still have the same order as the posterior contraction rate by definition.

D.2.3 Acceptable Approximation Errors of Properties (i) and (ii) in Corollary 5.1

Even when we restrict to a finite horizon m, there can still be expected deviations from Eq. (5.1), and thus those in Corollary 5.1, simply because Eq. (5.1) represents invariance conditions that are not "hard-wired" in the LLM's architecture. Yet, small violations of these equalities should not have practical consequences. We now derive the order of what is an acceptable violation in the setting of Example 5.2.

As discussed in this example, the equalities in Corollary 5.1 guarantee the expressions for posterior mean and covariance for the parameter θ to have consistently defined values, regardless of the choices of (n', k). The posterior mean has the order of $\Theta(1)$ and requires the violation of Corollary 5.1 (i) to be $o(1)$. The posterior covariance is generally $\Omega(1/n)$ and can be expressed through Example 5.2 as

$$\mathrm{Cov}(\theta|Z_{1:n}) = \mathbb{E}(Z_{n+1}Z_{n+k}|Z_{1:n}) - \mathbb{E}(Z_{n+k}|Z_{1:n})^2.$$

Therefore, it can have an approximately consistent value if the equalities in Corollary 5.1 hold approximately *up to an error of* $o(1/n)$. Posterior mean and

covariance are key quantities in the interpretation of predictive uncertainty, which in turn is a major benefit of the martingale property. Thus, we consider the above deviation to be acceptable as it already guarantees the approximately consistent interpretation of predictive uncertainty through the martingale property.

D.3 Additional Experimental Details and Results

D.3.1 Additional Experimental Details

Test statistics of properties implied by the martingale property. We summarise and empirically measure properties (i) and (ii) in Corollary 5.1 using the aggregated statistics

$$T_{1,g} := \frac{2}{Jm} \sum_{j=1}^{J} \sum_{i=1}^{m/2} (g(z_{n+i}^{(j)}) - g(z_{n+i+m/2}^{(j)})), \tag{D.1}$$

$$T_{2,k} := \frac{1}{Jm} \sum_{j=1}^{J} \sum_{i=1}^{m-k-1} (z_{n+i+1}^{(j)} - z_{n+i+k}^{(j)}) z_{n+i}^{(j)}. \tag{D.2}$$

The statistics $T_{1,g}$ and $T_{2,k}$ are defined using samples $\{z_{n+i}^{(j)}\}$ from J paths generated by an LLM via ICL and correspond to Monte-Carlo estimates of the expectations in properties (i) and (ii). To be robust against the possible outlier paths (§5.3.1), we remove sample paths with anomalous mean absolute values using the standard 1.5×IQR rule.

We compare the observed value of the statistics above evaluated on LLMs with bootstrap confidence intervals computed using a reference Bayesian model (§5.4.1). For the latter, we draw $K = 300$ sets of completions $\{\{z_{bs,n+i}^{(j,k)} : 1 \leq i \leq m, 1 \leq j \leq J\} : 1 \leq k \leq K\}$ from the predictive distribution of the reference Bayesian model, which provides K samples for the test statistics, and compute two-sided confidence intervals using the respective quantiles.

Experimental setup. For the first two experiments we vary $n \in \{20, 50, 100\}$, $m \in \{n/2, 2n\}$ and sample $J = 200$ paths from the LLMs. For the natural language experiments we fix $n = 100, m = 50, J = 80$. As non-exchangeable models may demonstrate different behaviour on different permutations of the same dataset, for the experiments in §5.4.2 we permute the observations when generating each sample path, so that we can produce a single test statistic that summarises each experiment configuration. For the experiments in §5.4.3, however, we use a fixed ordering for the observations for all path samples within each run, and report the median inter-quartile range across 9 runs for each configuration. This change is made to avoid (possibly small) deviations from exchangeability from inflating the estimated spread of the posterior.

For a proper test of the martingale property, it is vital that the model cannot distinguish between the ICL training data $Z_{1:n}$ and its own generations $\{Z_{n+i}\}$. This is trivially true if the LLM takes free-form text as inputs without additional annotation, as with `llama-2-7b`, `mistral-7b`, and `gpt-3.5` accessed through the `Completion` API from OpenAI. However, the `gpt-4` model is only accessible through a different API (`ChatCompletion`) which includes annotation for user input and model generation in the prompt. To ensure a proper implementation of the checks, we hence call the API m times in generating each path sample. In each iteration we sample a single data point, and then append it to the user input part of the prompt. This is far less cost-efficient than our use of `gpt-3.5`. Therefore, we only include `gpt-4` for the Bernoulli experiment with $n \leq 50$, and the natural language experiment.

We discuss prompt design and format in detail below. Here we emphasise that across all tasks, the prompt always includes sufficient information about the true likelihood.

Prompt design and format. We use the following prompt format `<instruction> <observed data> <sampled data>`. `<instruction>` describes the distribution (i.e.

true likelihood) of the observed data and importantly states that the observed samples were drawn i.i.d., i.e. from exchangeable random variables. `<observed data>` and `<sampled data>` lists the observed $z_{1:n}$, and sampled data $\hat{z}_{n+k}$ (if there exists any), respectively. Samples are represented depending on the experiment: as `int` values as 1-digit characters (e.g. '1'), `float` values with 1-digit of precision (e.g. '2.2') or words for synthetic natural language. As a sanity check, we also consider replacing integers with random words (e.g. 'tiger' for '1', 'hedgehog' for '0'), but did not notice important differences in the LLMs' behaviour. Each sample is delineated by a separator (e.g. ';').

We present exemplary prompts for each dataset below:

- A Bernoulli experiment with $n = 5$ and $m = 2$: *"Provided are independent, identically distributed tosses of a coin, which flips 1 with probability p where p is unknown: 1;0,0,1,0,0;1".*
- A Gaussian experiment with $n = 2$ and $m = 3$: *"Provided are independent, identically distributed draws from a Gaussian, with fixed but unknown mean and unit variance: 1.1,0.8,1.3,1.0,0.9".*
- The the natural language experiment: *"You will make predictions for a novel disease. The observed dataset contains records for multiple subjects which are assumed to be independent and identically distributed. For each subject there are two binary variables, indicating fever and disease diagnosis, respectively. Output your prediction for the disease diagnosis of the next subject.\n Id: 0\n Fever: Y\n Diagnosis: N ... "*

Other work represents both `int` and `float` numbers as a space-separated string of digits with fixed precision, where each number is separated by a semi-colon. This guarantees a per-digit tokenisation that was observed to be beneficial in the context of time series forecasting and further minimises the required number of

tokens per number as the decimal point is redundant [78]. We did not opt for this representation and corresponding tokenisation for two reasons: First, initial experiments with GPT-2 showed deteriorating sampling performance, where the model often hallucinated unrelated content. Second, and related to the first point, this representation is somewhat 'out-of-distribution' and probably unseen in the training distribution, which could limit and constrain any conclusions made in our experiments. Note that because of the tokenisation, in §5.4, the Gaussian experiment is more difficult than the Bernoulli experiment (or any dataset with single-token samples) as the LLM is required to learn the correlation structure between consecutive tokens representing a real-valued number.

Additional details for the natural language experiment. For the natural language experiment, we modify the scheme as follows: we split the ICL dataset and the imputations into two sequences $(\{Y_{i_{0,k}}\}_{k=1}^{n_1+m_1}, \{Y_{i_{1,k}}\}_{k=1}^{n_0+m_0})$ based on the value of X_i. Subsequently, either sequence contains i.i.d. Bernoulli random variables with a different mean, and any Bayesian ICL model with a correctly specified likelihood must produce imputations following a separate Bayesian posterior for Bernoulli data. Thus, we can apply our Bernoulli diagnostics separately to both sequence. This modification allows us to focus on LLMs' conditional predictive distributions of the form $p_M(Y_{i+1}|X_{i+1}, Z_{1:i})$, which is more relevant in practice.

D.3.2 Further Experimental Results and Discussion

Full results: Bernoulli experiments. Figs. D.1 and D.2 report the full results for the Bernoulli experiment in the setting of Fig. 5.3 ($m \in \{n/2, 2n\}$), where we also visualise the $o(1/n)$ 'acceptable deviation' (§D.2.3) using a light shade with width $0.1/n$. Consistent with the results in Fig. 5.3, for all models except the least capable `gpt-3-2.7b`, the martingale property is generally satisfied in the short-horizon scheme ($m = n/2$), but increasingly violated as we move to $m = 2n$.

The results for `gpt-3-2.7b` provide a sanity check of our experiment setup: Its unsatisfactory performance shows that our tasks require nontrivial ICL capabilities which are known to be absent in `gpt-3-2.7b` [27]. As another sanity check we provide the results for $m = 10n$ in Fig. D.5, where we drop all GPT models due to limitations with its API. As we can see, in this setting where the sampling horizon becomes even longer, deviation from the martingale property also becomes more severe. The consistently large negative value of $T_{1,g}$ indicates a continual upward bias towards 1, which demonstrates the 'creation of new knowledge' phenomenon discussed in §5.2.2.

Full results: Gaussian experiments. We report additional results for the Gaussian experiment in Fig. D.3 ($\theta = 0$) and Fig. D.4 ($\theta = -1$). As we can see, all models generally demonstrate a deviation from the martingale property when $\theta = -1$, but with $\theta = 0$ they may often appear to satisfy the property within a shorter horizon ($m = n/2$). Results for $\theta = 1$ are similar to the $\theta = -1$ case and thus omitted. We note that in many cases the predictive distribution cannot be matched to any Bayesian posterior with the correct likelihood: for the latter the sample variance should be greater than 1, the likelihood variance, but this is often not true for the LLMs. For example, for `gpt-3.5` in the setting of Fig. 5.4 we find the sample variance to be $0.711 < 1$ (95% CI: $[0.680, 0.742]$).

Additional results: Scaling of epistemic uncertainty. To avoid clutter, in Fig. 5.6 we have plotted the sample median of the test statistic T_3 from various models, and in that aspect `gpt-3-170b` appears to be close to the reference Bayesian model when n is smaller. Here we note that a deviation becomes more evident if we compare the individual samples of T_3 (obtained from independent runs) against bootstrap CIs from the reference Bayesian model, as shown in Fig. D.6.

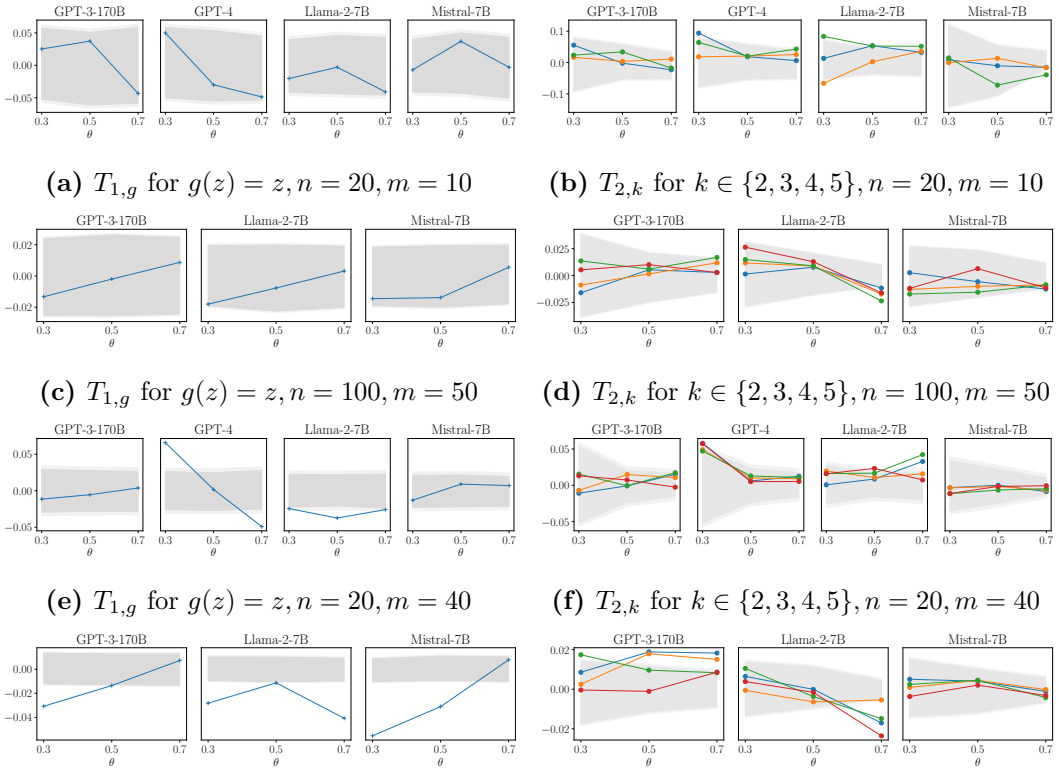
Additional results with fine-tuned models. Some previous works [107, 256] studied ICL under the assumption that the LLM has been perfectly pretrained on the ICL test distributions. While such assumptions are somewhat unrealistic, it may still be interesting to investigate whether finetuning on datasets that are similar to the ICL test distribution could lead to a closer-to-Bayesian behaviour for ICL. To this end we finetune `gpt-3-2.7b` models on Bernoulli and synthetic NLP datasets with randomly sampled parameters, and repeat the checks in §5.3.2 on the finetuned models.[1] The results are visualised in Fig. D.7. We can see that the finetuned models may indeed demonstrate a better adherence to the martingale property, but they do not always pass the checks.

Comparison of LLMs with and without instruction tuning. Among all LLMs evaluated, `gpt-3.5` and `gpt-4` generally demonstrate the worst performance in our evaluations, and incidentally they are the only LLMs that have undergone instruction tuning. The comparison between `gpt-3-170b` and `gpt-3.5` (see Fig. 5.6 and Fig. D.2) is particularly interesting since the two models are generally similar, with a main difference being the presence of instruction tuning. These observations seem to suggest that instruction tuning may have exacerbated the non-Bayesian ICL behaviour. Such an explanation would be broadly consistent with the previous findings that instruction tuning generally causes the LLM to produce less calibrated uncertainty estimates [78, 113, 176].

Could the LLMs correspond to 'unreasonable Bayesian models'? As discussed in the main text, our findings suggest that the behaviours of `gpt-3.5` and `llama-2-7b` are highly unlikely to correspond to any 'reasonable' Bayesian models in the Bernstein von-Mises sense. Generally speaking, to conduct any

[1] We use OpenAI's finetuning service which determines optimisation hyperparameters by validation loss. For the Bernoulli dataset, we sampled 10^4 sequence for training, each with an expected length of 75; the true parameter θ is sampled from the uniform distribution on $[0, 1]$. For the NLP dataset, we sampled 5000 sequences for training, each with an expected length of 85; the two Bernoulli parameters that determine the prompt distribution are sampled from a $\text{Beta}(0.5, 0.5)$ distribution.

statistical test with a reasonable level of power it is necessary to impose some regularity restrictions on the null hypothesis to be tested. Moreover, it could be similarly concerning if the LLMs correspond to any 'unreasonable' Bayesian model that does not satisfy the regularity conditions in the Bernstein von-Mises theorem. Nonetheless, in the setting above we can also provide some informal discussion on why the LLMs are unlikely to be 'unreasonable' Bayesian models (e.g., one with an approximately degenerate prior), by comparing the results across different choices of n. Specifically, for `gpt-3.5`, its small-sample behaviour in Fig. 5.6 can only be explained as a Bayesian model with a very strong prior that has the bulk of its mass near the true parameter; yet this would contradict its larger-than-regular posterior spread when n is large. For `llama-2-7b`, its large-sample behaviour could only be explained with the exact opposite (e.g., a Beta$(100, 100)$ prior); yet that should have led to a much larger IQR when n is small.



(a) $T_{1,g}$ for $g(z) = z, n = 20, m = 10$

(b) $T_{2,k}$ for $k \in \{2, 3, 4, 5\}, n = 20, m = 10$

(c) $T_{1,g}$ for $g(z) = z, n = 100, m = 50$

(d) $T_{2,k}$ for $k \in \{2, 3, 4, 5\}, n = 100, m = 50$

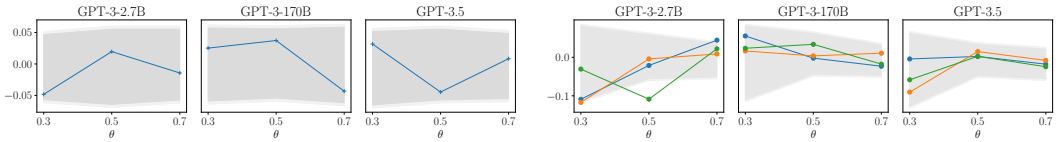
(e) $T_{1,g}$ for $g(z) = z, n = 20, m = 40$

(f) $T_{2,k}$ for $k \in \{2, 3, 4, 5\}, n = 20, m = 40$

(g) $T_{1,g}$ for $g(z) = z, n = 100, m = 200$

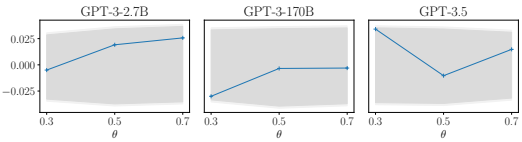
(h) $T_{2,k}$ for $k \in \{2, 3, 4, 5\}, n = 100, m = 200$

Figure D.1: Checking the martingale property: results for the Bernoulli experiments for all choices of (n, m) in the setting of Fig. 5.3. Note that we drop `gpt-4` for $n = 100$ due to API limitations (as discussed in App. D.3.1).

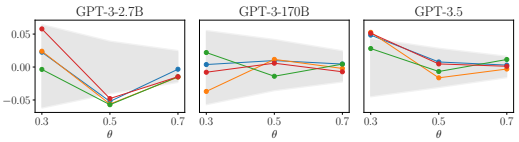


(a) $T_{1,g}$ for $g(z) = z, n = 20, m = 10$

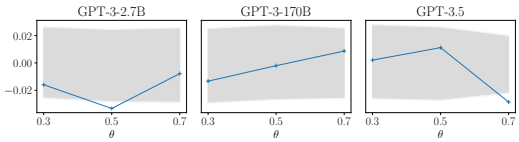
(b) $T_{2,k}$ for $k \in \{2, 3, 4, 5\}, n = 20, m = 10$



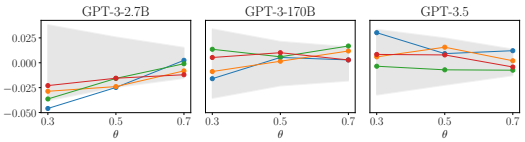
(c) $T_{1,g}$ for $g(z) = z, n = 20, m = 10$



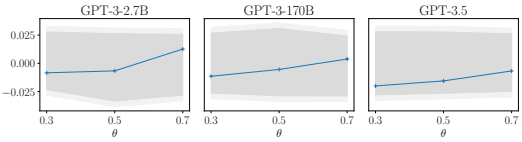
(d) $T_{2,k}$ for $k \in \{2, 3, 4, 5\}, n = 20, m = 10$



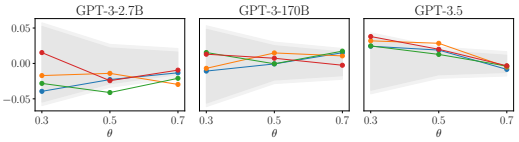
(e) $T_{1,g}$ for $g(z) = z, n = 100, m = 50$



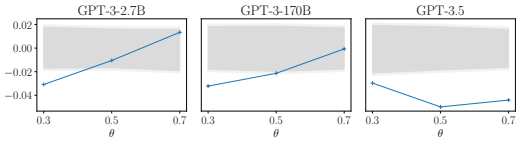
(f) $T_{2,k}$ for $k \in \{2, 3, 4, 5\}, n = 100, m = 50$



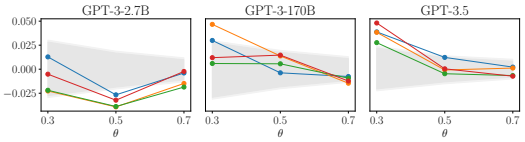
(g) $T_{1,g}$ for $g(z) = z, n = 20, m = 40$



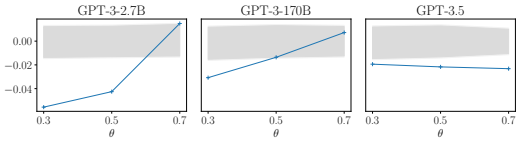
(h) $T_{2,k}$ for $k \in \{2, 3, 4, 5\}, n = 20, m = 40$



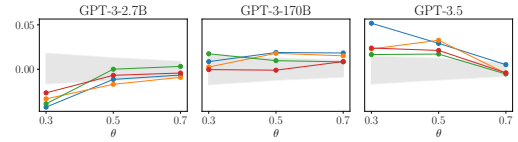
(i) $T_{1,g}$ for $g(z) = z, n = 50, m = 100$



(j) $T_{2,k}$ for $k \in \{2, 3, 4, 5\}, n = 50, m = 100$

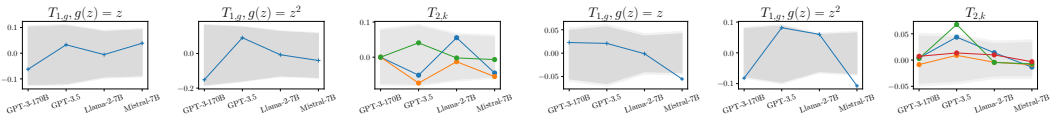


(k) $T_{1,g}$ for $g(z) = z, n = 100, m = 200$

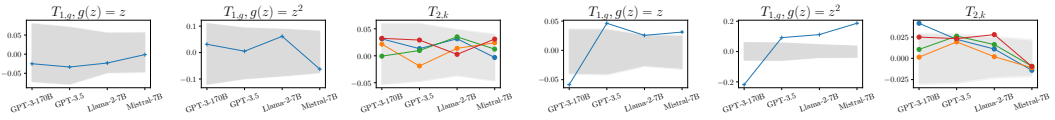


(l) $T_{2,k}$ for $k \in \{2, 3, 4, 5\}, n = 100, m = 200$

Figure D.2: Checking the martingale property: results for `gpt-3-2.7b` and `gpt-3.5` in the setting of Fig. 5.3.

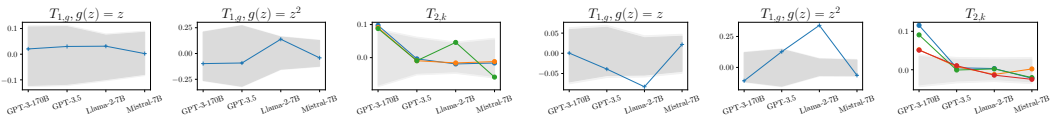


(a) $n = 20, m = 10$ **(b)** $n = 20, m = 40$

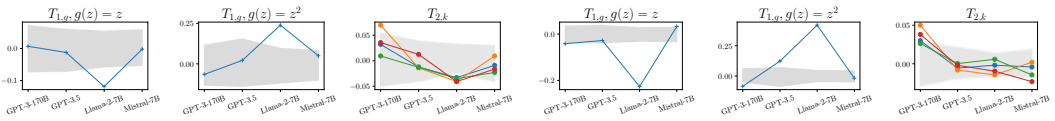


(c) $n = 50, m = 25$ **(d)** $n = 50, m = 100$

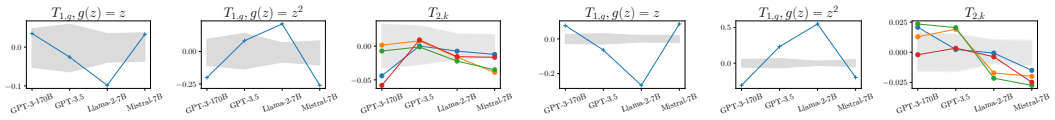
Figure D.3: Checking the martingale property: results for the Gaussian experiments with $\theta = 0$. See Fig. 5.4 for details.



(a) $n = 20, m = 10$ **(b)** $n = 20, m = 40$

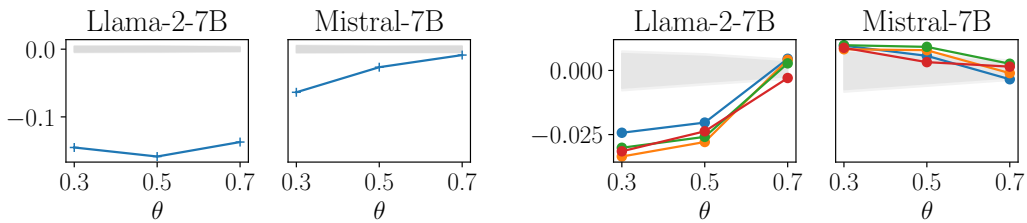


(c) $n = 50, m = 25$ **(d)** $n = 50, m = 100$



(e) $n = 100, m = 50$ **(f)** $n = 100, m = 200$

Figure D.4: Checking the martingale property: results for the Gaussian experiments with $\theta = -1$. See Fig. 5.4 for details.



(a) $T_{1,g}, g(z) = z$ **(b)** $T_{2,k}, k \in \{2, 3, 4, 5\}$

Figure D.5: Checking the martingale property on Bernoulli experiments: additional result with $n = 100, m = 10n$. See Fig. 5.3 for details.

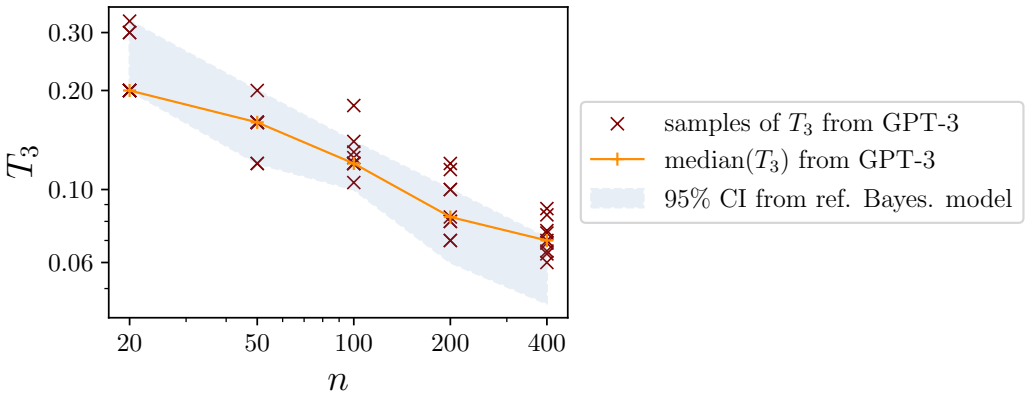
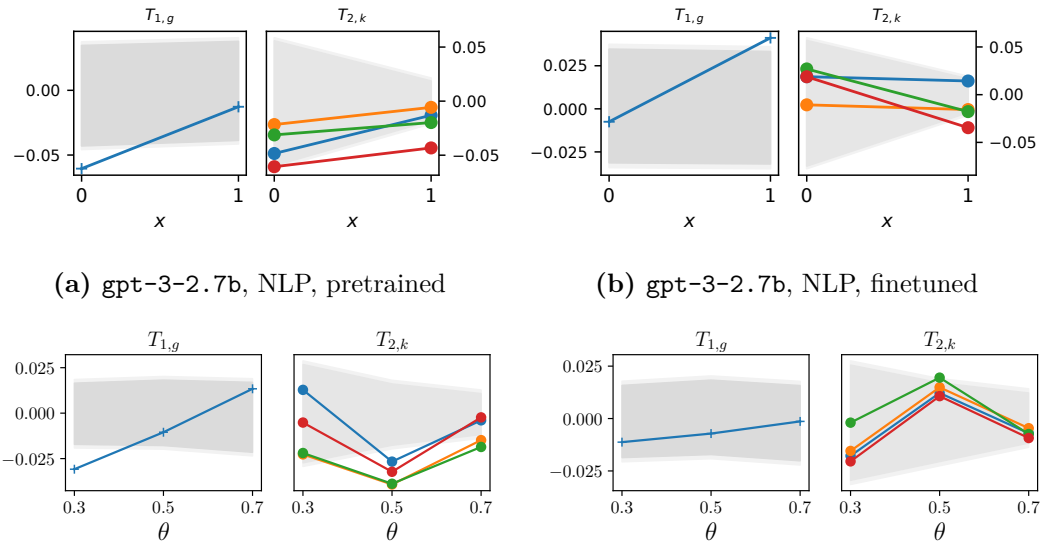


Figure D.6: Scaling of epistemic uncertainty: samples of the test statistic T_3 evaluated on `gpt-3-170b`, compared with the 95% CI from the reference Bayesian model.



(a) `gpt-3-2.7b`, NLP, pretrained

(b) `gpt-3-2.7b`, NLP, finetuned

(c) `gpt-3-2.7b`, Bernoulli ($n = 50, m = 100$), pretrained

(d) `gpt-3-2.7b`, Bernoulli ($n = 50, m = 100$), finetuned

Figure D.7: Checking the martingale property: comparison of `gpt-3` models before and after fine-tuning on the NLP (Fig. 5.5) and Bernoulli (Fig. 5.3) datasets.

D.4 Related Work

In-context learning as Bayesian inference. Numerous papers have explained ICL as performing some form of Bayesian Inference. The hypothesis is likewise studied in [107, 231] and the concurrent work of [249]. It is also covered by [256] if we restrict to exchangeable demonstrations. Closely related are the works of [3, 178] which demonstrate that high-capacity transformers pretrained with square loss may recover the Bayes predictor.

[243] studied ICL in a setting where the the LLM is perfectly trained on a pretraining distribution defined by a Hidden Markov Model (HMM). Under this and further assumptions, they prove that the LLM must implicitly perform Bayesian inference to infer a latent concept of the prompt. Strictly speaking, their assumptions do not exactly match our hypothesis, because their Bayesian model employs a likelihood that is misspecified for ICL: it does not assume $\{Z_i\}$ is conditionally i.i.d. or exchangeable. However, their additional assumptions render the ICL behaviour similar to that of another Bayesian model that assumes conditionally i.i.d. observations: when considering Eq. (8-10) in their work, which imply that the log likelihood of their Bayesian model is well approximated by a Bayesian model assuming conditional i.i.d. observations. In this regard their analysis is connected to our hypothesis, as it applies to the Bayesian model we study. It is important to note that their assumptions have been crucial in their proof for sample efficiency. More broadly, for any ICL predictor to be sample efficient on exchangeable $\{Z_i\}$, it is perhaps reasonable to expect the predictor to (approximately) recognise the exchangeable nature of $\{Z_i\}$, where our hypothesis would apply.

We review the other works in brief in the following. [100] dicussed high-level connections between [243] and various notions of exchangeability. [82][Section 1.4] relates to [243] as it can similarly be understood in terms of Bayesian inference, with the difference that they view the training tasks to be open-ended and compositional, in contrast to the finite nature of an HMM. [231] likewise takes a Bayesian viewpoint,

which they utilise to select the ICL dataset optimally. [107] explains various phenomena of the 'emergent abilities' of LLMs, such as in-context learning and chain-of-thought prompting, through Bayesian inference on the common distribution underlying natural languages. [256] show that ICL implicitly uses a Bayesian model averaging. [77] recover the prior distributions in LLMs for everyday observations, such as the time of movies.

Theories for in-context learning. Numerous theoretical models and frameworks beyond Bayesian inference exist which aim at understanding and formalising ICL. We refer to [59] for a detailed survey on in-context learning. [3] prove that transformer-based architectures can implement classical learning algorithms such as linear models and ridge regression. [10] extend this work by demonstrating that ICL via transformers can implement and even braoder set of algorithms, including convex risk minimisation algorithms and gradient descent, where the model intrinsically selects a different learning algorithm based on the task at hand. [205] shows that the ability of performing ICL algorithms such as Bayesian inference may be a transient phenomenon which produces highest accuracy during certain stages of pretraining an LLM. [192] show that the ability of in-context learning to tasks unseen during training by picking the right learning algorithm depends on the task diversity during training.

Input order dependence of Large language models. Previous work has found a dependence of LLMs on the order in which an input sequence is presented. [149] demonstrate that input order can significantly change the performance of an LLM in text classification tasks from "state-of-the-art" to "random guess". In the context of few-shot learning, [262] show the prediction of an LLM can depend on many seemingly irrelevant items, such as the prompt format or the order in which input examples are presented in a prompt, again with a sensititivity of performance to these factors. [254] note that the topic structure of a document may be exchangeable,

which motivates them to use Bayesian models, namely Latent Dirichlet Allocation, to analyse the representations of an LLM. Our discussion on exchangeability relates to this line of work, but has a novel perspective on it through our focus on the martingale property, a necessary condition for exchangeability, among other implications of the martingale property which we study (e.g. the decomposition of uncertainty and the resulting identification of epistemic uncertainty). Furthermore, in contrast to the related work, which shuffles the input data $Z_{1:n}$, we analyse the effect of shuffling the imputed, generated sequence $Z_{n+1}, \ldots$, where we find non-exchangeable behaviour which deviates from any reasonable Bayesian model.

Miscellaneous. Our work also relates a number of applications of LLMs. As we are generating samples from an LLM with ICL, which as we demonstrate deviate from the distribution of the ICL dataset, this work relates to and has implications for a line of work on LLMs for synthetic data generation [23, 83, 140, 214, 227]. Furthermore, we show that the martingale property is violated for long sampling paths, which may have implications for time series prediction with LLMs [78, 109], particularly over long horizons. We also demonstrate a dependence on the order in which missing values are imputed, which has direct implications for the machine learning task of missing value imputations with LLMs [160]. [203] demonstrate that models (including LLMs) which are recursively trained on data which they have previously generated shift in their distribution, where long tails disappear. While this work 'conditions' on synthetic data by retraining, our work analyses the conditioning via ICL. Lastly, as LLMs violate the martingale property in certain empirical regimes, they hence do not allow for a decomposed interpretation of their predictive uncertainty, which has important implications for uncertainty quantification with LLMs [241].

D.5 Negative Societal Impact

This paper analyses and characterises the behaviour of LLMs. We try to understand whether ICL in LLMs follows Bayesian principles. As we outlined in §5.2.3 this has important consequences for their potential use as trustworthy systems, which can be deployed in safety-critical, high-stakes applications such as healthcare. These systems often crucially rely on a principled notion of uncertainty. The evidence presented in this work cautions against the use of LLMs in such settings without further checks as they—under certain experimental settings—do not possess such a principled interpretation of uncertainty, rendering their uncertainty 'black-box'. Furthermore, while LLMs have typically been trained in non-exchangeable scenarios (e.g. natural language where the order of words or tokens changes meaning), as we showed in §5.2.2, we caution against their use in exchangeable settings (e.g. i.i.d. in-context data) as their predictions can be rendered inconsistent.

The points noted above are potential negative societal impacts if Bayesian behaviour cannot be guaranteed by a model, as we argue in this work. While we do not see any direct negative consequences from our analysis, we believe this work provides ample pointers and reason for further investigation of these concerns, and shall point out and warn against (potentially intended) misuse of LLMs.

D.6 Code, Computational Resources, Datasets, Existing Assets Used

Code. We provide our code base on `https://github.com/meta-inf/bayes_icl` under MIT License, together with a `README.md` containing instructions on reproducing the key results in this paper.

Datasets. We used three synthetic datasets for our experiments: a coin flip experiment, sampling from univariate Bernoulli distributions, a Gaussian experiment,

sampling from univariate Gaussian distributions, and a synthetic natural language experiment, sampling (conditionally) from Bernoulli distributions. We refer to §5.4 and App. D.3 where they are introduced and discussed.

Computational resources and APIs used. Referring to §5.4, we implemented `llama-2-7B` and `mistral-7B` with the Huggingface Transformer library [240], and implemented `gpt-3`, `gpt-3.5` and `gpt-4` using the OpenAI API [176]. For all Huggingface models, we generated the sampling paths by performing inference on a single A100 Nvidia GPU for each run.

Existing assets used. Our work uses the following main software libraries and corresponding licenses: PyTorch [180] (custom license), `numpy` [85] (BSD 3-Clause License), Weights&Biases [19] (MIT License), Huggingface transformers library [240] (Apache License 2.0; model licenses see below), `matplotlib` [99] (PSF License), `tqdm` [41] (MPLv2.0 MIT License), `scikit-learn` and `sklearn` [181] (BSD 3-Clause License), `pandas` [159] (BSD 3-Clause License), `openai` (Apache 2.0 License), `tiktoken` (MIT License), and `pickle` [225] (License N/A). We use Github Copilot and ChatGPT [176] for code development and occasionally as a writing aid.

The five pretrained large language models we used (see §5.4) have the following licenses: `llama-2-7B` [218] (custom license); `mistral-7B` [106] (Apache 2.0 License); `gpt-3` [27], `gpt-3.5`, and `gpt-4` [176] (API; no code license).

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. "GPT-4 Technical Report". In: *arXiv preprint arXiv:2303.08774* (2023).

[2] Shmuel Agmon. *Lectures on elliptic boundary value problems*. Vol. 369. American Mathematical Soc., 2010.

[3] Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. "What learning algorithm is in-context learning? investigations with linear models". In: *arXiv preprint arXiv:2211.15661* (2022).

[4] Elie Aljalbout, Vladimir Golkov, Yawar Siddiqui, Maximilian Strobel, and Daniel Cremers. "Clustering with deep learning: Taxonomy and new methods". In: *arXiv preprint arXiv:1801.07648* (2018).

[5] Shun-Ichi Amari. "Natural gradient works efficiently in learning". In: *Neural computation* 10.2 (1998), pp. 251–276.

[6] Shun-ichi Amari, Jimmy Ba, Roger Grosse, Xuechen Li, Atsushi Nitanda, Taiji Suzuki, Denny Wu, and Ji Xu. "When does preconditioning help or hurt generalization?" In: *arXiv preprint arXiv:2006.10732* (2020).

[7] Aryan Verma. *Triangulations Using Matplotlib*. `https://www.scaler.com/topics/matplotlib/matplotlib-triangulation/`.

[8] O Axclsson and Panayot S Vassilevski. "Algebraic multilevel preconditioning methods. I". In: *Numerische Mathematik* 56.2-3 (1989), pp. 157–177.

[9] Owe Axelsson and Panayot S Vassilevski. "Algebraic multilevel preconditioning methods, II". In: *SIAM Journal on Numerical Analysis* 27.6 (1990), pp. 1569–1590.

[10] Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. "Transformers as Statisticians: Provable In-Context Learning with In-Context Algorithm Selection". In: *arXiv preprint arXiv:2306.04637* (2023).

[11] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. "Training a helpful and harmless assistant with reinforcement learning from human feedback". In: *arXiv preprint arXiv:2204.05862* (2022).

[12] Christoph Bandt, Nguyen Hung, and Hui Rao. "On the open set condition for self-similar fractals". In: *Proceedings of the american mathematical society* 134.5 (2006), pp. 1369–1374.

[13] Shruthi Bannur, Kenza Bouzid, Daniel C Castro, Anton Schwaighofer, Sam Bond-Taylor, Maximilian Ilse, Fernando Pérez-García, Valentina Salvatelli, Harshita Sharma, Felix Meissen, et al. "MAIRA-2: Grounded Radiology Report Generation". In: *arXiv preprint arXiv:2406.04449* (2024).

[14] Michael F Barnsley. *Fractals everywhere*. Academic press, 2014.

[15] Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. "Invertible residual networks". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 573–582.

[16] Michele Benzi. "Preconditioning techniques for large linear systems: a survey". In: *Journal of computational Physics* 182.2 (2002), pp. 418–477.

[17] Patrizia Berti, Luca Pratelli, and Pietro Rigo. "Limit theorems for a class of identically distributed random variables". en. In: *The Annals of Probability* 32.3 (July 2004). (Visited on 01/08/2024).

[18] Anirban Bhattacharya, Debdeep Pati, and YUN Yang. "Bayesian fractional posteriors". In: *Annals of Statistics* 47.1 (2019), pp. 39–66.

[19] Lukas Biewald. *Experiment Tracking with Weights and Biases*. Software available from wandb.com. 2020. URL: `https://www.wandb.com/`.

[20] Christopher M Bishop. "Pattern recognition". In: *Machine learning* 128.9 (2006).

[21] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. "On the opportunities and risks of foundation models". In: *arXiv preprint arXiv:2108.07258* (2021).

[22] Barry Boots, Kokichi Sugihara, Sung Nok Chiu, and Atsuyuki Okabe. "Spatial tessellations: concepts and applications of Voronoi diagrams". In: (2009).

[23] Vadim Borisov, Kathrin Seßler, Tobias Leemann, Martin Pawelczyk, and Gjergji Kasneci. "Language models are realistic tabular data generators". In: *arXiv preprint arXiv:2210.06280* (2022).

[24] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. "Generating Sentences from a Continuous Space". In: *arXiv preprint arXiv:1511.06349* (2015).

[25] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[26] Susanne C Brenner, L Ridgway Scott, and L Ridgway Scott. *The mathematical theory of finite element methods*. Vol. 3. Springer, 2008.

[27] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[28] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. "Importance weighted autoencoders". In: *arXiv preprint arXiv:1509.00519* (2015).

[29] Chris Burgess and Hyunjik Kim. *3D Shapes Dataset*. https://github.com/deepmind/3dshapes-dataset/. 2018.

[30] Ke Chen. *Matrix preconditioning techniques and applications*. Vol. 19. Cambridge University Press, 2005.

[31] Ricky T. Q. Chen, Xuechen Li, Roger Grosse, and David Duvenaud. "Isolating sources of disentanglement in variational autoencoders". In: *arXiv preprint arXiv:1802.04942* (2018).

[32] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. "Neural ordinary differential equations". In: *Advances in neural information processing systems* 31 (2018).

[33] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. "Training deep nets with sublinear memory cost". In: *arXiv preprint arXiv:1604.06174* (2016).

[34] Rewon Child. "Very Deep VAEs Generalize Autoregressive Models and Can Outperform Them on Images". In: *International Conference on Learning Representations*. 2021.

[35] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. "Generating long sequences with sparse transformers". In: *arXiv preprint arXiv:1904.10509* (2019).

[36] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. "A downsampled variant of imagenet as an alternative to the CIFAR datasets". In: *arXiv preprint arXiv:1707.08819* (2017).

[37] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. "3D U-Net: learning dense volumetric segmentation from sparse annotation". In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2016: 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II 19*. Springer. 2016, pp. 424–432.

[38] Bernardo Cockburn, George E Karniadakis, and Chi-Wang Shu. *The development of discontinuous Galerkin methods*. Springer, 2000.

[39] Andrew Collette. *Python and HDF5*. O'Reilly, 2013.

[40] pyyaml contributors. *pyyaml*. `https://github.com/yaml/pyyaml`. 2006.

[41] tqdm contributors. *ImageIO*. `https://github.com/tqdm/tqdm`. 2022.

[42] Rob Cornish, Anthony Caterini, George Deligiannidis, and Arnaud Doucet. "Relaxing bijectivity constraints with continuously indexed normalising flows". In: *International conference on machine learning*. PMLR. 2020, pp. 2133–2143.

[43] Fergal Cotter. "Uses of complex wavelets in deep convolutional neural networks". PhD thesis. University of Cambridge, 2020.

[44] Ying Cui, Xiaoli Z. Fern, and Jennifer G. Dy. "Non-redundant multi-view clustering via orthogonalization". In: *Proceedings - IEEE International Conference on Data Mining, ICDM* 3 (2007), pp. 133–142.

[45] Wolfgang Dahmen and Angela Kunoth. "Multilevel preconditioning". In: *Numerische Mathematik* 63.1 (1992), pp. 315–344.

[46] Lisandro Dalcin and Yao-Lung L Fang. "Mpi4py: Status update after 12 years of development". In: *Computing in Science & Engineering* 23.4 (2021), pp. 47–54.

[47] Ingrid Daubechies. *Ten Lectures on Wavelets*. SIAM, 1992.

[48] Sam Dauncey, Christopher C Holmes, Christopher Williams, and Fabian Falck. "On Gradients of Deep Generative Models for Representation-Invariant Anomaly Detection". In: *ICLR 2023 Workshop on Pitfalls of limited data and computation for Trustworthy ML*.

[49] Ian Davidson and Zijie Qi. "Finding alternative clusterings using constraints". In: *Proceedings - IEEE International Conference on Data Mining, ICDM* (2008), pp. 773–778.

[50] Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel. "The helmholtz machine". In: *Neural computation* 7.5 (1995), pp. 889–904.

[51] Valentin De Bortoli, Emile Mathieu, Michael Hutchinson, James Thornton, Yee Whye Teh, and Arnaud Doucet. "Riemannian score-based generative modeling". In: *arXiv preprint arXiv:2202.02763* (2022).

[52] Valentin De Bortoli, James Thornton, Jeremy Heng, and Arnaud Doucet. "Diffusion Schrödinger bridge with applications to score-based generative modeling". In: *Advances in Neural Information Processing Systems*. Vol. 34. 2021.

[53] Bruno De Finetti. "Funzione caratteristica di un fenomeno aleatorio". In: *Atti del Congresso Internazionale dei Matematici: Bologna del 3 al 10 de settembre di 1928*. 1929, pp. 179–190.

[54] Boris Delaunay. "Sur la sphère vide". In: *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk* 7.793-800 (1934), pp. 1–2.

[55] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database". In: *IEEE Conference on Computer Vision and Pattern Recognition*. Ieee. 2009, pp. 248–255.

[56] Prafulla Dhariwal and Alexander Nichol. "Diffusion Models Beat GANs on Image Synthesis". In: *Advances in Neural Information Processing Systems*. Vol. 34. 2021.

[57] Lee R Dice. "Measures of the amount of ecologic association between species". In: *Ecology* 26.3 (1945), pp. 297–302.

[58] Nat Dilokthanakul, Pedro A. M. Mediano, Marta Garnelo, Matthew C. H. Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. "Deep Unsupervised Clustering with Gaussian Mixture VAE". In: *arXiv preprint arXiv:1611.02648* (2017). arXiv: `1611.02648`.

[59] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. "A survey for in-context learning". In: *arXiv preprint arXiv:2301.00234* (2022).

[60] David L Donoho. "De-noising by soft-thresholding". In: *IEEE transactions on information theory* 41.3 (1995), pp. 613–627.

[61] Joseph L Doob. "Application of the theory of martingales". In: *Le calcul des probabilites et ses applications* (1949), pp. 23–27.

[62] Emilien Dupont, Yee Whye Teh, and Arnaud Doucet. "Generative models as distributions of functions". In: *arXiv preprint arXiv:2102.04776* (2021).

[63] S. M. Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S. Morcos, Marta Garnelo, Avraham Ruderman, Andrei A. Rusu, Ivo Danihelka, Karol Gregor, et al. "Neural scene representation and rendering". In: *Science* 360.6394 (2018), pp. 1204–1210.

[64] Patrick Esser, Ekaterina Sutter, and Björn Ommer. "A variational u-net for conditional appearance and shape generation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8857–8866.

[65] Fabian Falck, Ziyu Wang, and Christopher C. Holmes. "Is In-Context Learning in Large Language Models Bayesian? A Martingale Perspective". In: *Forty-first International Conference on Machine Learning*. 2024.

[66] Fabian Falck, Christopher Williams, Dominic Danks, George Deligiannidis, Christopher Yau, Chris C Holmes, Arnaud Doucet, and Matthew Willetts. "A multi-resolution framework for U-Nets with applications to hierarchical VAEs". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 15529–15544.

[67] Fabian Falck, Haoting Zhang, Matthew Willetts, George Nicholson, Christopher Yau, and Chris C Holmes. "Multi-facet clustering variational autoencoders". In: *Advances in Neural Information Processing Systems*. Vol. 34. 2021.

[68] William Falcon and The PyTorch Lightning team. *PyTorch Lightning*. Version 1.4. Mar. 2019. URL: `https://github.com/Lightning-AI/lightning`.

[69] Thorsten Falk, Dominic Mai, Robert Bensch, Özgün Çiçek, Ahmed Abdulkadir, Yassine Marrakchi, Anton Böhm, Jan Deubner, Zoe Jäckel, Katharina Seiwald, et al. "U-Net: deep learning for cell counting, detection, and morphometry". In: *Nature methods* 16.1 (2019), pp. 67–70.

[70] Edwin Fong, Chris Holmes, and Stephen G Walker. "Martingale posterior distributions". In: *arXiv preprint arXiv:2103.15671* (2021).

[71] Hongyang Gao and Shuiwang Ji. "Graph u-nets". In: *international conference on machine learning*. PMLR. 2019, pp. 2083–2092.

[72] Subhashis Ghosal and Aad Van der Vaart. *Fundamentals of nonparametric Bayesian inference*. Vol. 44. Cambridge University Press, 2017.

[73] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010, pp. 249–256.

[74] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial networks". In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[75] Prasoon Goyal, Zhiting Hu, Xiaodan Liang, Chenyu Wang, and Eric P. Xing. "Nonparametric variational auto-encoders for hierarchical representation learning". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 5094–5102.

[76] Will Grathwohl, Dami Choi, Yuhuai Wu, Geoffrey Roeder, and David Duvenaud. "Backpropagation through the void: Optimizing control variates for black-box gradient estimation". In: *International Conference on Learning Representations*. 2018.

[77] Thomas L Griffiths and Joshua B Tenenbaum. "Optimal predictions in everyday cognition". In: *Psychological science* 17.9 (2006), pp. 767–773.

[78] Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. "Large language models are zero-shot time series forecasters". In: *arXiv preprint arXiv:2310.07820* (2023).

[79] Jayesh K Gupta and Johannes Brandstetter. "Towards Multi-spatiotemporal-scale Generalized PDE Modeling". In: *arXiv preprint arXiv:2209.15616* (2022).

[80] Florentin Guth, Simon Coste, Valentin De Bortoli, and Stephane Mallat. "Wavelet Score-Based Generative Modeling". In: *arXiv preprint arXiv:2208.05003* (2022).

[81] Alfred Haar. *Zur theorie der orthogonalen funktionensysteme.* Georg-August-Universitat, Gottingen., 1909.

[82] Michael Hahn and Navin Goyal. "A theory of emergent in-context learning as implicit structure induction". In: *arXiv preprint arXiv:2303.07971* (2023).

[83] Perttu Hämäläinen, Mikke Tavast, and Anton Kunnari. "Evaluating large language models in generating synthetic hci research data: a case study". In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 2023, pp. 1–19.

[84] Pierre Hansen and Brigitte Jaumard. "Cluster analysis and mathematical programming". In: *Mathematical programming* 79.1-3 (1997), pp. 191–215.

[85] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, et al. "Array programming with NumPy". In: *Nature* 585.7825 (2020), pp. 357–362.

[86] Louay Hazami, Rayhane Mama, and Ragavan Thurairatnam. "Efficient-VDVAE: Less is more". In: *arXiv preprint arXiv:2203.13751* (2022).

[87] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[88] Wenchong He, Zhe Jiang, Chengming Zhang, and Arpan Man Sainju. "CurvaNet: Geometric deep learning based on directional curvature for 3D shape analysis". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 2214–2224.

[89] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. "Gans trained by a two time-scale update rule converge to a local nash equilibrium". In: *Advances in neural information processing systems* 30 (2017).

[90] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. "beta-VAE: Learning basic visual concepts with a constrained variational framework". In: *International Conference on Learning Representations*. 2017.

[91] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. "Flow++: Improving flow-based generative models with variational dequantization and architecture design". In: *International Conference on Machine Learning*. 2019.

[92] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising Diffusion Probabilistic Models". In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020.

[93] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[94] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, DDL Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. "Training compute-optimal large language models". In: *arXiv preprint arXiv:2203.15556* 10 (2022).

[95] Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. "simple diffusion: End-to-end diffusion for high resolution images". In: *arXiv preprint arXiv:2301.11093* (2023).

[96] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural networks* 2.5 (1989), pp. 359–366.

[97] Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. "Learning discrete representations via information maximizing self-augmented training". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1558–1567.

[98] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. "Deep Networks with Stochastic Depth". In: *European Conference on Computer Vision*. Springer. 2016, pp. 646–661.

[99] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95.

[100] Ferenc Huszár. *Implicit Bayesian Inference in Large Language Models*. https://www.inference.vc/implicit-bayesian-inference-in-sequence-models/. 2022.

[101] Nabil Ibtehaz and M Sohel Rahman. "MultiResUNet: Rethinking the U-Net architecture for multimodal biomedical image segmentation". In: *Neural networks* 121 (2020), pp. 74–87.

[102] Fabian Isensee, Paul F Jaeger, Simon AA Kohl, Jens Petersen, and Klaus H Maier-Hein. "nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation". In: *Nature methods* 18.2 (2021), pp. 203–211.

[103] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. "Image-To-Image Translation With Conditional Adversarial Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[104] Allan Jabri, David Fleet, and Ting Chen. "Scalable Adaptive Computation for Iterative Generation". In: *arXiv preprint arXiv:2212.11972* (2022).

[105] Eric Jang, Shixiang Gu, and Ben Poole. "Categorical reparameterization with gumbel-softmax". In: *International Conference on Learning Representations*. 2017.

[106] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. "Mistral 7B". In: *arXiv preprint arXiv:2310.06825* (2023).

[107] Hui Jiang. "A latent space theory for emergent abilities in large language models". In: *arXiv preprint arXiv:2304.09960* (2023).

[108] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. "Variational Deep Embedding: An Unsupervised and Generative Approach to Clustering". In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*. 2017, pp. 1965–1972. URL: https://doi.org/10.24963/ijcai.2017/273.

[109] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, et al. "Time-llm: Time series forecasting by reprogramming large language models". In: *arXiv preprint arXiv:2310.01728* (2023).

[110] Matthew James Johnson, David Duvenaud, Alexander B. Wiltschko, Sandeep R. Datta, and Ryan P. Adams. "Composing Graphical Models with Neural Networks for Structured Representations and Fast Inference". In: *Advances in Neural Information Processing Systems*. 2016. arXiv: 1603.06277.

[111] Tom Joy, Sebastian M Schmon, Philip HS Torr, N Siddharth, and Tom Rainforth. "Capturing label characteristics in vaes". In: *arXiv preprint arXiv:2006.10102* (2020).

[112] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin **Z**ídek, Anna Potapenko, et al. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596.7873 (2021), pp. 583–589.

[113] Adam Tauman Kalai and Santosh S Vempala. "Calibrated language models must hallucinate". In: *arXiv preprint arXiv:2311.14648* (2023).

[114] Olav Kallenberg. *Foundations of modern probability*. Vol. 2. Springer, 1997.

[115] Ozsel Kilinc and Ismail Uysal. "Learning Latent Representations in Neural Networks for Clustering Through Pseudo Supervision and Graph-based activity Regularization". In: *International Conference on Learning Representations*. 2018. arXiv: arXiv:1810.05749v1.

[116] Hyunjik Kim and Andriy Mnih. "Disentangling by factorising". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 2649–2658.

[117] Diederik Kingma and Ruiqi Gao. "Understanding diffusion objectives as the elbo with simple data augmentation". In: *Advances in Neural Information Processing Systems* 36 (2024).

[118] Diederik P. Kingma and Prafulla Dhariwal. "Glow: Generative flow with invertible 1x1 convolutions". In: *Advances in Neural Information Processing Systems*. 2018.

[119] Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. "Improved variational inference with inverse autoregressive flow". In: *Advances in Neural Information Processing Systems*. 2016.

[120] Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. "Variational Diffusion Models". In: *Advances in Neural Information Processing Systems*. Vol. 34. 2021.

[121] Diederik P. Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[122] Bohdan Kivva, Goutham Rajendran, Pradeep Ravikumar, and Bryon Aragam. "Identifiability of deep generative models without auxiliary information". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 15687–15701.

[123] Almar et al. Klein. *ImageIO*. https://zenodo.org/record/6551868#.Yolo_5PMIhg. 2022.

[124] Simon Kohl, Bernardino Romera-Paredes, Clemens Meyer, Jeffrey De Fauw, Joseph R Ledsam, Klaus Maier-Hein, SM Eslami, Danilo Jimenez Rezende, and Olaf Ronneberger. "A probabilistic u-net for segmentation of ambiguous images". In: *Advances in neural information processing systems* 31 (2018).

[125] Simon AA Kohl, Bernardino Romera-Paredes, Klaus H Maier-Hein, Danilo Jimenez Rezende, SM Eslami, Pushmeet Kohli, Andrew Zisserman, and Olaf Ronneberger. "A hierarchical probabilistic u-net for modeling multi-scale ambiguities". In: *arXiv preprint arXiv:1905.13077* (2019).

[126] Philipp Krähenbühl, Carl Doersch, Jeff Donahue, and Trevor Darrell. "Data-dependent initializations of convolutional neural networks". In: *International Conference on Learning Representations*. 2016.

[127] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).

[128] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. *80 million tiny images*. `http://groups.csail.mit.edu/vision/TinyImages/`. 2020.

[129] Harold W. Kuhn. "The Hungarian method for the assignment problem". In: *Naval research logistics quarterly* 2.1-2 (1955), pp. 83–97.

[130] Hugo J Kuijf, J Matthijs Biesbroek, Jeroen De Bresser, Rutger Heinen, Simon Andermatt, Mariana Bento, Matt Berseth, Mikhail Belyaev, M Jorge Cardoso, Adria Casamitjana, et al. "Standardized assessment of automatic segmentation of white matter hyperintensities and results of the WMH segmentation challenge". In: *IEEE transactions on medical imaging* 38.11 (2019), pp. 2556–2568.

[131] D La Torre and F Mendivil. "The Monge–Kantorovich metric on multimeasures and self–similar multimeasures". In: *Set-Valued and Variational Analysis* 23.2 (2015), pp. 319–331.

[132] Zoe Landgraf, Fabian Falck, Michael Bloesch, Stefan Leutenegger, and Andrew J Davison. "Comparing view-based and map-based semantic labelling in real-time SLAM". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 6884–6890.

[133] Yann LeCun, Corinna Cortes, and C. J. Burges. *MNIST handwritten digit database*. `http://yann.lecun.com/exdb/mnist/`. 2010.

[134] Der-Tsai Lee and Bruce J Schachter. "Two algorithms for constructing a Delaunay triangulation". In: *International Journal of Computer & Information Sciences* 9.3 (1980), pp. 219–242.

[135] Gregory Lee, Ralf Gommers, Filip Waselewski, Kai Wohlfahrt, and Aaron O'Leary. "PyWavelets: A Python package for wavelet analysis". In: *Journal of Open Source Software* 4.36 (2019), p. 1237.

[136] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. "Visualizing the Loss Landscape of Neural Nets". In: *Advances in Neural Information Processing Systems*. Vol. 31. 2018.

[137] Hongwei Li, Gongfa Jiang, Jianguo Zhang, Ruixuan Wang, Zhaolei Wang, Wei-Shi Zheng, and Bjoern Menze. "Fully convolutional network ensembles for white matter hyperintensities segmentation in MR images". In: *NeuroImage* 183 (2018), pp. 650–665.

[138] Xiaopeng Li, Zhourong Chen, Leonard K. M. Poon, and Nevin L. Zhang. "Learning latent superstructures in variational autoencoders for deep multidimensional clustering". In: *International Conference on Learning Representations*. 2019. arXiv: 1803.05206.

[139] Zhiyuan Li, Jaideep Vitthal Murkute, Prashnna Kumar Gyawali, and Linwei Wang. "Progressive learning and disentanglement of hierarchical representations". In: *International Conference on Learning Representations*. 2020.

[140] Zhuoyan Li, Hangxiao Zhu, Zhuoran Lu, and Ming Yin. "Synthetic Data Generation with Large Language Models for Text Classification: Potential and Limitations". In: *arXiv preprint arXiv:2310.07849* (2023).

[141] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. "Fourier neural operator for parametric partial differential equations". In: *arXiv preprint arXiv:2010.08895* (2020).

[142] Valentin Liévin, Andrea Dittadi, Lars Maaløe, and Ole Winther. "Towards hierarchical discrete variational autoencoders". In: *2nd Symposium on Advances in Approximate Bayesian Inference*. 2019.

[143] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. "Flow matching for generative modeling". In: *arXiv preprint arXiv:2210.02747* (2022).

[144] P. Liu, H. Zhang, K. Zhang, L. Lin, and W. Zuo. "Multi-level wavelet-CNN for image restoration". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2018, pp. 773–782.

[145] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. "Deep Learning Face Attributes in the Wild". In: *Proceedings of International Conference on Computer Vision (ICCV)*. Dec. 2015.

[146] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. "Large-scale CelebFaces Attributes (CelebA) Dataset". In: (2022).

[147] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. "Challenging common assumptions in the unsupervised learning of disentangled representations". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 4114–4124.

[148] Mert Lostar and Islem Rekik. "Deep hypergraph u-net for brain graph embedding and classification". In: *arXiv preprint arXiv:2008.13118* (2020).

[149] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. "Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity". In: *arXiv preprint arXiv:2104.08786* (2021).

[150] Calvin Luo. "Understanding diffusion models: A unified perspective". In: *arXiv preprint arXiv:2208.11970* (2022).

[151] Lars Maaløe, Marco Fraccaro, Valentin Liévin, and Ole Winther. "BIVA: A Very Deep Hierarchy of Latent Variables for Generative Modeling". In: *Advances in Neural Information Processing Systems*. 2019. arXiv: 1902.02102v1.

[152] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. "The concrete distribution: A continuous relaxation of discrete random variables". In: *International Conference on Learning Representations*. 2017.

[153] Stéphane Mallat. *A Wavelet Tour of Signal Processing*. Elsevier, 1999.

[154] Stephane G Mallat. "Multiresolution approximations and wavelet orthonormal bases of L2(R)". In: *Transactions of the American Mathematical Society* 315.1 (1989), pp. 69–87.

[155] Hariharan Manikandan, Yiding Jiang, and J. Zico Kolter. *Language models are weak learners*. arXiv:2306.14101 [cs]. June 2023. URL: http://arxiv.org/abs/2306.14101 (visited on 03/06/2024).

[156] Xiaojiao Mao, Chunhua Shen, and Yu-Bin Yang. "Image Restoration Using Very Deep Convolutional Encoder-Decoder Networks with Symmetric Skip Connections". In: *Advances in Neural Information Processing Systems*. Vol. 29. 2016.

[157] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[158] Emile Mathieu, Tom Rainforth, N. Siddharth, and Yee Whye Teh. "Disentangling disentanglement in variational autoencoders". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 4402–4412.

[159] Wes McKinney. "Data Structures for Statistical Computing in Python". In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. 2010, pp. 56–61.

[160] Yinan Mei, Shaoxu Song, Chenguang Fang, Haifeng Yang, Jingyun Fang, and Jiang Long. "Capturing semantics for imputation with pre-trained language models". In: *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE. 2021, pp. 61–72.

[161] Amil Merchant, Simon Batzner, Samuel S Schoenholz, Muratahan Aykol, Gowoon Cheon, and Ekin Dogus Cubuk. "Scaling deep learning for materials discovery". In: *Nature* 624.7990 (2023), pp. 80–85.

[162] Tomáš Mikolov et al. "Statistical language models based on neural networks". In: *Presentation at Google, Mountain View, 2nd April* 80.26 (2012).

[163] Erxue Min, Xifeng Guo, Qiang Liu, Gen Zhang, Jianjing Cui, and Jun Long. "A survey of clustering with deep learning: From the perspective of network architecture". In: *IEEE Access* 6 (2018), pp. 39501–39514.

[164] Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. "Metaicl: Learning to learn in context". In: *arXiv preprint arXiv:2110.15943* (2021).

[165] Sudipto Mukherjee, Himanshu Asnani, Eugene Lin, and Sreeram Kannan. "ClusterGAN: Latent space clustering in generative adversarial networks". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 2019, pp. 4610–4617.

[166] Emmanuel Muller, Stephan Gunnemann, Ines Farber, and Thomas Seidl. "Discovering multiple clustering solutions: Grouping objects in different views of the data". In: *2012 IEEE 28th International Conference on Data Engineering*. IEEE. 2012, pp. 1207–1210.

[167] Kevin P Murphy. *Probabilistic machine learning: Advanced topics*. MIT press, 2023.

[168] Kevin P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[169] Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. "Reliable fidelity and diversity metrics for generative models". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 7176–7185.

[170] Eric Nalisnick, Lars Hertel, and Padhraic Smyth. "Approximate Inference for Deep Latent Gaussian Mixtures". In: *Workshop on Bayesian Deep Learning Workshop (NIPS 2016)*. 2016.

[171] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. "Reading digits in natural images with unsupervised feature learning". In: *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*. 2011.

[172] Alex Nichol and Prafulla Dhariwal. "Improved denoising diffusion probabilistic models". In: *arXiv preprint arXiv:2102.09672* (2021).

[173] Beatrix MG Nielsen, Anders Christensen, Andrea Dittadi, and Ole Winther. "DiffEnc: Variational diffusion with a learned encoder". In: *arXiv preprint arXiv:2310.19789* (2023).

[174] NVIDIA. *Apex*. `https://github.com/NVIDIA/apex`. 2019.

[175] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, et al. "Attention u-net: Learning where to look for the pancreas". In: *arXiv preprint arXiv:1804.03999* (2018).

[176] OpenAI. *GPT-4 Technical Report*. 2023. arXiv: `2303.08774 [cs.CL]`.

[177] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. "Training language models to follow instructions with human feedback". In: *Advances in neural information processing systems* 35 (2022), pp. 27730–27744.

[178] Madhur Panwar, Kabir Ahuja, and Navin Goyal. "In-Context Learning through the Bayesian Prism". In: *The Twelfth International Conference on Learning Representations*. 2024.

[179] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. "Image transformer". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 4055–4064.

[180] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems*. 2019, pp. 8024–8035.

[181] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[182] William Peebles, John Peebles, Jun-Yan Zhu, Alexei A. Efros, and Antonio Torralba. "The Hessian Penalty: A Weak Prior for Unsupervised Disentanglement". In: *Proceedings of European Conference on Computer Vision (ECCV)*. 2020.

[183] William Peebles and Saining Xie. "Scalable Diffusion Models with Transformers". In: *arXiv preprint arXiv:2212.09748* (2022).

[184] Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. "Instruction tuning with gpt-4". In: *arXiv preprint arXiv:2304.03277* (2023).

[185] Adeel Pervez, Taco Cohen, and Efstratios Gavves. "Low Bias Low Variance Gradient Estimates for Boolean Stochastic Networks". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 7632–7640.

[186] Vinay Uday Prabhu and Abeba Birhane. "Large image datasets: A pyrrhic win for computer vision?" In: *arXiv preprint arXiv:2006.16923* (2020).

[187] Zijie Qi and Ian Davidson. "A principled and flexible framework for finding alternative clusterings". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2009), pp. 717–725.

[188] Svetlozar T Rachev. *Probability Metrics and the Stability of Stochastic Models*. Vol. 269. Wiley, 1991.

[189] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. "Language models are unsupervised multitask learners". In: *OpenAI blog* 1.8 (2019), p. 9.

[190] Ali Rahimi and Benjamin Recht. "Random features for large-scale kernel machines". In: *Advances in Neural Information Processing Systems*. Vol. 20. 2007.

[191] Z. Ramzi, K. Michalewicz, JL. Starck, et al. "Wavelets in the Deep Learning Era". In: *J Math Imaging Vis* 65 (2023), pp. 240–251.

[192] Allan Raventós, Mansheej Paul, Feng Chen, and Surya Ganguli. "Pretraining task diversity and the emergence of non-Bayesian in-context learning for regression". In: *arXiv preprint arXiv:2306.15063* (2023).

[193] William H Reed and Thomas R Hill. *Triangular mesh methods for the neutron transport equation*. Tech. rep. Los Alamos Scientific Lab., N. Mex.(USA), 1973.

[194] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. "Stochastic backpropagation and approximate inference in deep generative models". In: *International Conference on Machine Learning*. 2014.

[195] Michal Rolinek, Dominik Zietlow, and Georg Martius. "Variational autoencoders pursue pca directions (by accident)". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 12406–12415.

[196] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. "High-resolution image synthesis with latent diffusion models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10684–10695.

[197] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical Image Computing and Computer-assisted Intervention*. Springer. 2015, pp. 234–241.

[198] Khaled Saab, Tao Tu, Wei-Hung Weng, Ryutaro Tanno, David Stutz, Ellery Wulczyn, Fan Zhang, Tim Strother, Chunjong Park, Elahe Vedadi, et al. "Capabilities of gemini models in medicine". In: *arXiv preprint arXiv:2404.18416* (2024).

[199] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding". In: *arXiv preprint arXiv:2205.11487* (2022).

[200] Mehdi SM Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. "Assessing generative models via precision and recall". In: *Advances in neural information processing systems* 31 (2018).

[201] Uri Shaham, Kelly Stanton, Henry Li, Boaz Nadler, Ronen Basri, and Yuval Kluger. "SpectralNet: Spectral clustering using deep neural networks". In: *International Conference on Learning Representations*. 2018.

[202] Yuyang Shi, Valentin De Bortoli, George Deligiannidis, and Arnaud Doucet. "Conditional Simulation Using Diffusion Schrödinger Bridges". In: *Uncertainty in Artificial Intelligence*. 2022.

[203] Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. "Model dementia: Generated data makes models forget". In: *arXiv e-prints* (2023), arXiv–2305.

[204] Nahian Siddique, Paheding Sidike, Colin Elkin, and Vijay Devabhaktuni. "U-Net and its variants for medical image segmentation: theory and applications". In: *arXiv preprint arXiv:2011.01118* (2020).

[205] Aaditya K Singh, Stephanie CY Chan, Ted Moskovitz, Erin Grant, Andrew M Saxe, and Felix Hill. "The transient nature of emergent in-context learning in transformers". In: *arXiv preprint arXiv:2311.08360* (2023).

[206] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. "Deep unsupervised learning using nonequilibrium thermodynamics". In: *International Conference on Machine Learning*. PMLR. 2015, pp. 2256–2265.

[207] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. "Ladder Variational Autoencoders". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016.

[208] Jiaming Song, Chenlin Meng, and Stefano Ermon. "Denoising Diffusion Implicit Models". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=St1giarCHLP.

[209] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. "Score-based generative modeling through stochastic differential equations". In: *arXiv preprint arXiv:2011.13456* (2020).

[210] Thorvald A Sorensen. "A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish commons". In: *Biol. Skar.* 5 (1948), pp. 1–34.

[211] Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. "PDEBench: An extensive benchmark for scientific machine learning". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 1596–1611.

[212] Kamrul Hasan Talukder and Koichi Harada. "Haar wavelet based approach for image compression and quality assessment of compressed image". In: *arXiv preprint arXiv:1010.4084* (2010).

[213] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. "Fourier features let networks learn high frequency functions in low dimensional domains". In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020.

[214] Ruixiang Tang, Xiaotian Han, Xiaoqian Jiang, and Xia Hu. "Does synthetic data generation of llms help clinical text mining?" In: *arXiv preprint arXiv:2303.04360* (2023).

[215] David Taubman and Michael Marcellin. *JPEG2000 image compression fundamentals, standards and practice: image compression fundamentals, standards and practice*. Vol. 642. Springer Science & Business Media, 2012.

[216] Yonglong Tian, Dilip Krishnan, and Phillip Isola. "Contrastive Multiview Coding". In: *Computer Vision – ECCV 2020*. Springer International Publishing, 2020.

[217] Naftali Tishby, Fernando C. Pereira, and William Bialek. "The Information Bottleneck Method". In: (2000). URL: http://arxiv.org/abs/physics/0004057.

[218] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. "Llama 2: Open foundation and fine-tuned chat models". In: *arXiv preprint arXiv:2307.09288* (2023).

[219] Eli Turkel. "Preconditioning techniques in computational fluid dynamics". In: *Annual Review of Fluid Mechanics* 31.1 (1999), pp. 385–416.

[220] P Umesh. "Image Processing in Python". In: *CSI Communications* 23 (2012).

[221] Arash Vahdat and Jan Kautz. "NVAE: A deep hierarchical variational autoencoder". In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020.

[222] Aaron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu, et al. "Wavenet: A generative model for raw audio". In: *arXiv preprint arXiv:1609.03499* 12 (2016).

[223] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. "Conditional Image Generation with PixelCNN Decoders". In: *Advances in Neural Information Processing Systems* 29 (2016).

[224] Aad W Van der Vaart. *Asymptotic statistics*. Vol. 3. Cambridge university press, 2000.

[225] Guido Van Rossum. *The Python Library Reference, release 3.8.2*. Python Software Foundation, 2020.

[226] A Vaswani. "Attention is all you need". In: *Advances in Neural Information Processing Systems* (2017).

[227] Veniamin Veselovsky, Manoel Horta Ribeiro, Akhil Arora, Martin Josifoski, Ashton Anderson, and Robert West. "Generating Faithful Synthetic Data with Large Language Models: A Case Study in Computational Social Science". In: *arXiv preprint arXiv:2305.15041* (2023).

[228] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." In: *Journal of machine learning research* 11.12 (2010).

[229] Ulrike Von Luxburg, Robert C. Williamson, and Isabelle Guyon. "Clustering: Science or art?" In: *Proceedings of ICML workshop on unsupervised and transfer learning*. JMLR Workshop and Conference Proceedings. 2012, pp. 65–79.

[230] Haohan Wang, Xindi Wu, Zeyi Huang, and Eric P Xing. "High-frequency component helps explain the generalization of convolutional neural networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 8684–8694.

[231] Xinyi Wang, Wanrong Zhu, Michael Saxon, Mark Steyvers, and William Yang Wang. "Large language models are latent variable models: Explaining and finding good demonstrations for in-context learning". In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023.

[232] Michael L. Waskom. "seaborn: statistical data visualization". In: *Journal of Open Source Software* 6.60 (2021), p. 3021.

[233] A. J. Wathen. "Preconditioning". In: *Acta Numerica* 24 (2015), pp. 329–376.

[234] Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. "De novo design of protein structure and function with RFdiffusion". In: *Nature* 620.7976 (2023), pp. 1089–1100.

[235] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. "Chain-of-thought prompting elicits reasoning in large language models". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 24824–24837.

[236] Matthew Willetts, Stephen J. Roberts, and Chris Holmes. "Disentangling to Cluster: Gaussian Mixture Variational Ladder Autoencoders". In: *4th Workshop on Bayesian Deep Learning (NeurIPS 2019)*. 2019.

[237] Matthew Willetts, Stephen J. Roberts, and Christopher C. Holmes. "Semi-Unsupervised Learning using Deep Generative Models". In: *3rd Workshop on Bayesian Deep Learning (NeurIPS 2018)*. 2018.

[238] Matthew Willetts, Stephen J. Roberts, and Christopher C. Holmes. "Semi-Unsupervised Learning: Clustering and Classifying using Ultra-Sparse Labels". In: *IEEE Big Data Workshop*. 2020. arXiv: 1901.08560.

[239] Christopher Williams, Fabian Falck, George Deligiannidis, Chris C Holmes, Arnaud Doucet, and Saifuddin Syed. "A unified framework for U-Net design and analysis". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 27745–27782.

[240] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. "Transformers: State-of-the-Art Natural Language Processing". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45.

[241] Yuxin Xiao, Paul Pu Liang, Umang Bhatt, Willie Neiswanger, Ruslan Salakhutdinov, and Louis-Philippe Morency. "Uncertainty quantification with pre-trained language models: A large-scale empirical analysis". In: *arXiv preprint arXiv:2210.04714* (2022).

[242] Junyuan Xie, Ross Girshick, and Ali Farhadi. "Unsupervised deep embedding for clustering analysis". In: *International Conference on Machine Learning*. PMLR. 2016, pp. 478–487.

[243] Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. "An explanation of in-context learning as implicit bayesian inference". In: *arXiv preprint arXiv:2111.02080* (2021).

[244] Rui Xu and Donald Wunsch. "Survey of clustering algorithms". In: *IEEE Transactions on neural networks* 16.3 (2005), pp. 645–678.

[245] J. Yan, B. Zheng, H. Xu, Y. Zhu, D. Chen, J. Sun, J. Wu, and J. Chen. "Making pre-trained language models great on tabular prediction". In: *International Conference on Learning Representations*. 2024.

[246] Bo Yang, Xiao Fu, Nicholas D. Sidiropoulos, and Mingyi Hong. "Towards k-means-friendly spaces: Simultaneous deep learning and clustering". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 3861–3870.

[247] Xu Yang, Cheng Deng, Feng Zheng, Junchi Yan, and Wei Liu. "Deep spectral clustering using dual autoencoder network". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4066–4075.

[248] Jong Chul Ye, Yoseob Han, and Eunju Cha. "Deep Convolutional Framelets: A General Deep Learning Framework for Inverse Problems". In: *SIAM Journal on Imaging Sciences* 11.2 (2018), pp. 991–1048.

[249] Naimeng Ye, Hanming Yang, Andrew Siah, and Hongseok Namkoong. "Pre-training and In-context Learning IS Bayesian Inference a la De Finetti". In: *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*. 2024.

[250] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. "Multi-Stage Progressive Image Restoration". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021.

[251] Arnold Zellner. "Optimal information processing and Bayes's theorem". In: *The American Statistician* 42.4 (1988), pp. 278–280.

[252] Claudio Zeni, Robert Pinsler, Daniel Zügner, Andrew Fowler, Matthew Horton, Xiang Fu, Sasha Shysheya, Jonathan Crabbé, Lixin Sun, Jake Smith, et al. "Mattergen: a generative model for inorganic materials design". In: *arXiv preprint arXiv:2312.03687* (2023).

[253] Li Zhang, Hengyuan Ma, Xiatian Zhu, and Jianfeng Feng. "Preconditioned Score-based Generative Models". In: *arXiv preprint arXiv:2302.06504* (2023).

[254] Liyi Zhang, R Thomas McCoy, Theodore R Sumers, Jian-Qiao Zhu, and Thomas L Griffiths. "Deep de Finetti: Recovering Topic Distributions from Large Language Models". In: *arXiv preprint arXiv:2312.14226* (2023).

[255] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. "The unreasonable effectiveness of deep features as a perceptual metric". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 586–595.

[256] Yufeng Zhang, Fengzhuo Zhang, Zhuoran Yang, and Zhaoran Wang. "What and How does In-Context Learning Learn? Bayesian Model Averaging, Parameterization, and Generalization". In: *arXiv preprint arXiv:2305.19420* (2023).

[257] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. "Road extraction by deep residual u-net". In: *IEEE Geoscience and Remote Sensing Letters* 15.5 (2018), pp. 749–753.

[258] Fenqiang Zhao, Zhengwang Wu, Li Wang, Weili Lin, John H Gilmore, Shunren Xia, Dinggang Shen, and Gang Li. "Spherical deformable u-net: Application to cortical surface parcellation and development prediction". In: *IEEE transactions on medical imaging* 40.4 (2021), pp. 1217–1228.

[259] Shengjia Zhao, Jiaming Song, and Stefano Ermon. "Learning hierarchical features from deep generative models". In: *International Conference on Learning Representations*. 2017.

[260] Shengjia Zhao, Jiaming Song, and Stefano Ermon. "Learning hierarchical features from deep generative models". In: *International Conference on Machine Learning*. 2017.

[261] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. "A survey of large language models". In: *arXiv preprint arXiv:2303.18223* (2023).

[262] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. "Calibrate before use: Improving few-shot performance of language models". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 12697–12706.

[263] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. "Unet++: A nested u-net architecture for medical image segmentation". In: *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: 4th International Workshop, DLMIA 2018, and 8th International Workshop, ML-CDS 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 20, 2018, Proceedings 4*. Springer. 2018, pp. 3–11.

[264] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. "Fine-tuning language models from human preferences". In: *arXiv preprint arXiv:1909.08593* (2019).