



UNIVERSITY OF OXFORD

COMPUTER SCIENCE DEPARTMENT

DPhil THESIS

Improving Representation Learning through Variational Autoencoding

AUTHOR: SHUYU LIN, BALLIOL COLLEGE

SUPERVISORS: PROFESSOR NIKI TRIGONI

PROFESSOR STEPHEN ROBERTS

Acknowledgements

“路漫漫其修远兮，吾将上下而求索”

Translation: The path (to truth) is long and far, but I will not stop searching for it.

- Qu Yuan (c.340 BC – 278 BC), a Chinese poet and politician

I consider the five years in my PhD as a journey to pursuing truth - some very small pieces of unknown truth that keep me up at nights. As the Chinese poet wisely said 2 thousand years ago, this journey is not easy. The biggest challenges in this journey is doubt. I doubt myself. I doubt every decision I have made during my PhD: "Am I working on an interesting topic?" "Am I asking the right question?" "Will this idea solve the problem?" "Ah it does not work very well, should I stick to it, or should I leave it and try something else?" "Ah something interesting seems happening, but what exactly is that?" The second challenge is loneliness. Inspiring talks and engaging discussion with supervisors and colleagues can only help so far. Nobody shares the exact same objectives. Nobody wants to answer the exact same questions. To get to the truth I am after, I need to find my own way.

Luckily after five years of learning, asking questions, proposing ideas, submitting and re-submitting, I am here today - writing this last piece in my thesis. I want to thank many people - my parents, my supervisors, my friends, my cat and my boyfriend. It is their trust that I am able to complete this journey and to answer the questions that I want to ask that helps me to overcome doubt and loneliness. So I would like to dedicate this thesis to them and to this unforgettable journey of my life.

Shuyu Lin, London, May 2022

Abstract

Representation learning aims to distill useful knowledge from raw data and apply this knowledge to a wide range of applications. This ability to extract information that is useful not only for selected tasks but also generalizes to new settings is a key step towards artificial intelligence.

In this thesis, we focus on representations derived through a specific type of generative model, i.e. variational autoencoders (VAEs). VAEs have several desirable properties. Thanks to the use of variational inference and the convenient model assumptions of Gaussian posteriors and a simple prior, VAEs are often easy to train and exhibit fast convergence. The probabilistic modelling formulation allows VAEs to derive a smooth latent representation of the raw data (i.e. semantically similar data samples are likely to be projected to nearby regions in the latent space). VAEs compress the raw data to a much lower dimension latent space. Working with the low-dimensional representations rather than the raw data can significantly reduce costs in memory and computation. With these advantages, VAEs have been widely applied to many applications, including robotics [1], drug discovery [2] and digital content creation [3].

Despite the widespread application of VAEs, improving the generative modelling of VAEs further remains an active research topic. In this thesis, we focus on two challenges in the VAE training: 1) over-regularized posterior distributions are often encountered in VAEs with Gaussian decoders and simple prior models; 2) the auto-encoding function may cause severe information drift and alter the information in the raw data in successive encodings. We propose solutions to both phenomena. Specifically, we optimize a variance parameter in the Gaussian decoder to balance competing loss terms in the ELBO objective. We adopt a flexible prior model that is implemented as a VAE in the latent space to mitigate the over-regularization effects. To reduce the information drift, we propose to modify the ELBO objective with a consistency loss that penalizes such drift. We show that these proposals can effectively address the challenges identified previously and improve the likelihood score of the VAEs.

In addition to the contributions related to improving VAEs, we also demonstrate the power of VAEs' representation learning in two important machine learning applications. Firstly, we show that a VAE's ability to compress complicated, high-dimensional data is key to achieving good performance in anomaly detection. We design a VAE-LSTM anomaly detection system that can accurately identify anomalous effects in a time series. Secondly, we show that a classifier which incorporates a VAE module can give better calibrated predictions. This is the result of a VAEs' capability of expressing the uncertainty between similar data samples in the spread of the posterior distribution and of identifying out-of-distribution samples.

Keywords: Representation learning, unsupervised learning, variational autoencoders, anomaly detection, uncertainty estimation.

Acronyms

VAEs	variational autoencoders	2
ELBO	evidence lower bound	2
OOD	out of distribution	6
LaDDer	latent data distribution modelling with a generative prior	7
LSTM	long short-term memory	8
pPCA	probabilistic principal component analysis	10
GAN	generative adversarial network	2
i.i.d.	independent and identically distribute	17
LVMs	latent variable models	13
FID	Frechet inception distance	50
LL	log likelihood	45
IWAE	importance weighted autoencoder	72
GMM	Gaussian mixture model	76
KL	Kullback-Leibler	25
CNNs	convolutional neural networks	100
RNNs	recurrent neural networks	100

Contents

1	Introduction	1
1.1	Research questions	4
1.2	Research contributions	7
1.3	Thesis structure	10
1.4	Datasets	11
2	Background	13
2.1	Important distributions	13
2.1.1	Definition of a probability distribution	14
2.1.2	Discrete distributions	14
2.1.3	Gaussian distributions	16
2.1.4	Maximum likelihood estimation	17
2.2	Graphical models	19
2.2.1	Sampling in a directed graphical model	19
2.2.2	Conditional independence	20
2.3	Latent variable models	22
2.4	Variational inference	24
2.4.1	Evidence lower bound objective	25
2.4.2	Optimizing ELBO	26
2.4.3	The mean-field assumption	27
2.5	An example: probabilistic PCA	28
2.5.1	Model definition	29
2.5.2	Deriving ELBO objective	30
2.5.3	Exact update equations for the model parameters	32
2.5.4	Update procedures	33
2.5.5	Estimating parameters in an example	34
2.5.6	Optimization by gradient descent	37
2.6	Summary	39

3	Variational Autoencoders	41
3.1	Derivation of VAEs	41
3.1.1	Deriving ELBO objective	44
3.1.2	Learning model parameters	46
3.2	Evaluation metrics for VAEs	48
3.2.1	Likelihood metrics	48
3.2.2	Reconstruction error	49
3.2.3	Cross-entropy between aggregate posterior and prior	50
3.2.4	Generation quality metrics	50
3.3	Connection to other representation learning approaches	51
3.3.1	Generative adversarial networks (GANs)	52
3.3.2	Normalizing flows	54
3.4	Efforts to improve the VAE modelling	56
3.5	Summary	58
4	Challenges in the VAE modelling	59
4.1	Over-regularized encoding	59
4.1.1	Gaussian decoders with an arbitrary variance parameter	61
4.1.2	Overly restrictive prior	62
4.2	Encoding drift	64
4.3	Summary	67
5	Addressing the over-regularization issue	69
5.1	Inferring σ^2 from data	69
5.1.1	Optimizing σ^2 under ELBO	71
5.1.2	Extending to sample-specific, pixel-wise variances	71
5.1.3	Performance gain resulted from optimizing σ^2	72
5.1.4	Uncertainty indicated by the pixel-wise variances	73
5.2	Estimating the prior distribution	75
5.2.1	Introducing a generative prior	76
5.2.2	A Gaussian mixture model hyper prior	77
5.2.3	Optimizing model parameters via block coordinate ascent algorithm	78
5.2.4	Experimental results	79
5.2.5	Better latent space interpolation	83
5.3	Summary	85
6	Mitigating the encoding drift in VAEs	89

6.1	Introducing a consistency loss	89
6.2	Compute the consistency term $D_{\text{KL}}(\mathbf{z}' \mathbf{z})$	91
6.3	Reducing drift in the posterior distributions	92
6.4	Impact on other metrics	93
6.5	The weight parameter α	95
6.6	An interesting observation	95
6.7	Summary	97
7	Anomaly Detection for Time Series	99
7.1	Background and related work	99
7.2	Our method	101
7.2.1	Training VAE-LSTM Models	101
7.2.2	Anomaly Detection using the VAE-LSTM Model	103
7.3	Experiments and results	104
7.4	Summary	106
8	Uncertainty estimation for classification	109
8.1	Background and related work	110
8.2	Our Method	111
8.2.1	Classification with a hybrid model	111
8.2.2	Optimizing model parameters	112
8.2.3	Considering out-of-distribution data	112
8.2.4	System overview	114
8.3	Experiments and Results	115
8.4	Summary	119
9	Conclusion	121
9.1	Contributions	121
9.2	Future work	123
	Bibliography	127
A	Appendix A	I
A.1	Matrix derivatives	I
A.2	Expectation for vectors and matrices	I
A.3	Variational inference derivations	II
A.4	KL divergence between two Gaussians	IV
A.5	Likelihood for a VAE model	VI

B	Appendix B	VII
B.1	Additional results for Section 5.1	VII
B.1.1	More Visualisation of MNIST 2D Aggregate Posteriors	VII
B.2	Uncertainty Estimation	VIII
B.3	Additional results for Section 5.2	XI
B.3.1	Model Architectures and Training Parameters	XI
B.3.1.1	MNIST and Fashion MNIST	XI
B.3.1.2	CelebA	XIV
B.3.2	More Experimental Results	XV
B.3.2.1	Generated Samples from Inferred Prior Distribution .	XV
B.3.2.2	FID Scores on MNIST and fashion-MNIST	XVII
B.3.2.3	Ablation Study: the Impact of Hyper Latent Dimen- sion on Performance	XVIII
B.3.2.4	Latent Space Interpolation Through Our Shortest Likelihood Path	XVIII

1

Introduction

Over the last decade, great strides have been made in advancing the capabilities of machine learning algorithms. In many specific tasks, such as image classification and speech recognition, we are able to achieve (super) human performance with a well designed and calibrated algorithm. Many of these algorithms are trained in a supervised manner, i.e. the parameters of these algorithms are determined by labelled pairs of input $\mathbf{x}_i \in \mathbb{R}^n$ and output $\mathbf{y}_i \in \mathbb{R}^m$ data. However, the attention of the machine learning community has turned to the possibility of learning directly from unlabelled data, i.e. without supervision. This type of learning is often considered to be more challenging, as it is hard to define a universal learning objective that suits all types of data and all kinds of tasks.

This type of learning is important, as most real-world data are collected in a unsupervised fashion. Providing the supervision labels is usually a labour-intensive and time-consuming activity. Hence, any progress toward reducing the amount of labelled data needed is appealing. Furthermore, most biological intelligent systems appear to be capable of learning from unsupervised or very sparsely supervised data. This indicates that such a learning system exists. We just need to discover the right design principles and implement it with the right hardware and software to make it happen. This, of course, is not easy. There has been decades of research efforts in this direction and we are likely still in the early part of the journey.

One of the most exciting advances in the direction of unsupervised learning is the understanding that features, often derived as an intermediate by-product in a learning task, are an important asset themselves. Multiple studies [4], [5] have shown that features derived from a large-scale dataset can be useful for tasks different from the original task that these features are derived from. For example, the features derived from ImageNet [6] which contains 1.4 million natural images for an image classification task are shown to be useful for other tasks, including object detection [7], activity recognition [8], human pose estimation [9] and image captioning [10]. If finetuning is allowed, then these features can deliver an accuracy that surpasses

the accuracy from models only trained for the target task alone. These results show that the features derived from a dataset and a task can contain knowledge that is able to transfer to other datasets or tasks [11]. Since this realisation, the task of deriving more powerful and generalisable features has become an important research topic in machine learning and we refer to it as "*representation learning*".

Formally speaking, the objective of representation learning is to derive a mapping function that converts data observations, which can come in various forms, such as text (a sequence of discrete variables), speech signals (a sequence of continuous variables), images (regular grid of discrete variables) and graphs (nodes and edges), to a vector or a tensor in the Euclidean space. Representation learning is an unsupervised learning task, as no observations of "correct" representations are available. As a result, different approaches have been devised with different learning strategies to obtain the properties that a good representation is believed to possess. For example, auto-encoder based approaches [12] aim to learn a representation of the input data that maximally preserves the information in the original input. Thus, these approaches commonly minimize a reconstruction error, which is computed as the difference between the original input and a reconstructed version of it by decoding from the representation. The idea behind generative adversarial network (GAN) [13] based approaches is that a good representation can be learned by training two networks in tandem - one trying to generate realistic samples from the learned representation and the other trying to distinguish between real and fake samples. Likelihood based approaches [12] assume that the likelihood of a sample can be computed and by maximizing the likelihood over all training samples, a good representation can be learned. Some examples of likelihood based approaches include normalizing flows [14], auto-regressive models [15] and variational autoencoders (VAEs) [16], [17].

In this thesis, we focus primarily on one important class of representation learning approaches, that is *variational autoencoders* (VAEs) [16], [17]. VAEs are designed to capture the underlying data distribution that is most likely to generate a given set of data samples. To do so, VAEs introduce a low dimensional latent variable, which serves as a representation of the observed data variable. Utilizing the statistical technique of variational inference [18], [19], VAEs optimize an evidence lower bound (ELBO) learning objective. With some simplifying assumptions, such as the latent variable following a zero-mean unit-variance Gaussian distribution and the posterior and the conditional distribution between the data and the latent variable are both Gaussian distributions, the ELBO objective can be computed analytically. Coupled with deep neural networks' capability of approximating any complex func-

tions [20], [21], VAEs are able to obtain a mapping that encode any data samples to the corresponding latent representations and that decode any latent representations to the corresponding data samples by maximizing the ELBO objective using a stochastic gradient ascent procedure [22], [23]. Once the optimization of ELBO converges, these mapping functions can be used to convert any new data observations into the representations for use in downstream tasks.

There are various desirable properties that make VAEs one of the widely used representation learning approaches. Firstly, the vast majority of VAE models adopt a latent variable that is of (significantly) lower dimensionality compared to the data variable. This modelling choice ensures that the latent representation is a compressed version of the original data samples. The compression is obtained primarily via the following: 1) correlations across data dimensions are discovered and encoded in the encoding and decoding functions that VAEs infer and 2) random dispersion, beyond the modelling capacity of the given neural networks, is suppressed (and potentially removed). With this compressed representation, we can transmit the majority of information from the original data with less bandwidth, power and memory. Furthermore, with redundancy and noise decreased, we can make more reliable decisions based on the encoded data.

Secondly, VAEs are designed to construct a disentangled representation of the original data [24], [25]. Disentanglement leads to a representation such that each latent variable encodes for a subset of the information in the data. The change in one latent dimension therefore only affects that subset of information, leaving other information in the data unchanged. Disentanglement is obtained via a modelling requirement that the latent variables are as statistically independent as possible. Correlated information in the data will hence tend to be organized into a single latent dimension, allowing us to achieve a more compact version of the original data, with information in the original data represented well using a smaller set of independent latent basis variables.

Thirdly, VAEs are often used to make the decision produced by a black-box system more interpretable [26], [27]. This interpretability comes from two sources: 1) the disentangled representation allows us to discover the underlying relations between the original data variables and 2) the one-to-one mapping between the latent representation and the data samples allows us to translate abstract representation in the latent space to (interpretable) visualization in the data space. The latter property allows to carry out counterfactual reasoning [28], i.e. if certain changes are made in the latent representation, we can visualize the corresponding changes in the data space which will also be related to the impact on the final decision. As more

and more machine learning systems, which are hard to interpret and understand, are applied to important tasks in the domains including healthcare, finance and jurisdiction, any attempt to make these systems more interpretable is valuable, as it helps to build trust in the machine learning systems and avoid bad decisions in critical situations where loss of life, distress and injustice might be the consequences.

Finally, we can use the VAE’s ELBO objective, which is a lower bound to the log likelihood of the data distribution, as a metric to approximate the likelihood of a new data sample belonging to the data distribution. This property is valuable, as it grants us the ability to indicate when we encounter situations that we have not seen in the training data. Many deep learning based systems are often criticized for making over-confident decisions [29], [30]. A large part of this behaviour is because such systems cannot identify when they move into unfamiliar data regions. In contrast, VAEs’ ability to identify such situations allow us to build machine learning systems that can identify when they are faced with data that they have not seen before. It is worth mentioning that the approximation to the likelihood of the data distribution from the VAE’s ELBO objective is not perfect. In the cases where the data samples are completely different from training samples, the gap between ELBO and the true likelihood can be large. Therefore, it is still an active research area to estimate the gap between the two quantities and hence reduce the gap so we can have a better estimate of the likelihood.

1.1 Research questions

Despite the great promise of VAEs as discussed above, there are still major challenges that need to be overcome to facilitate their wide spread adoption in important domains such as healthcare, legislation, robotics and autonomous driving. In this thesis, we focus on answering the following research questions related to VAEs.

How do we improve the generative modelling performance of VAEs?
Many applications of VAEs rely on the assumption that the derived generative model is faithful and representative of the true data distribution. However, we often observe that the generated samples from VAEs are blurry and less diverse compared to the real data samples [31]. This indicates that a gap exists between the generative model inferred from VAEs and the true data distribution.

In this thesis, we are interested in improving the VAE models’ generative modelling capability in terms of reaching a higher ELBO and reducing information loss between the derived representation and the raw data. To do so, we focus on two

aspects. Firstly, we notice that VAEs can produce uninformative posterior distributions (i.e. the latent representations of the input data samples) and, hence, fail to offer a useful low-dimensional representation of the raw data. Such effects are primarily caused by an overly strong regularization being imposed from the prior distribution in a VAE model to the posterior distributions. We notice such effects in two scenarios: 1) VAEs with Gaussian decoders and 2) VAEs with overly simplistic prior model. We are interested in mitigating such effects for each of the scenarios and making sure meaningful posterior distributions can be derived from VAEs.

Secondly, we notice that the auto-encoding function inferred from a VAE model can cause severe modification of the information in the raw data. We notice this effect in a study where we repeatedly auto-encode (or reconstruct) an input sample. The posterior distribution of the reconstructed sample has shift away from the posterior distribution of the input sample. After several iterations of auto-encoding, the data sample has been transformed to a shape that is completely different from the original input. We refer to this behaviour of a VAE model as "*information drift*", where the auto-encoding function inferred in a VAE model alters the information in the input data and the latent representation of the input might not represent the input data perfectly. In this thesis, we are interested in finding out how severe this information drift is and find out ways to reduce such drift.

How can we apply VAEs to address the challenging task of anomaly detection for a time series? The ability to accurately detect anomalies in a sequential data is core to many applications, such as predictive maintenance, cybersecurity and financial applications. However, designing an accurate anomaly detection algorithm is very challenging, because we often have limited data of anomalies (as anomalies are rare events) and very likely the past instances of the anomalies do not represent the anomalous behaviours in the future (as anomalies can appear in a large number of forms).

We consider VAEs to be a good candidate for this challenging task, as we can use VAEs to estimate the distribution of normal behaviours in the given dataset, which we have plenty of observations of. With the ELBO objective - a lower bound to the data likelihood, we can easily indicate anomalous events when the ELBO of a given section of the time series drops to the low range. Due to these benefits, VAEs have been applied to anomaly detection in the context of non-sequential data and shown promising results [32]–[34]. However, there is another layer of complexity when the data considered is a time series: the anomalies can span over multiple scales of time. Directly applying VAEs on the sections of the time series with a fixed window length is unlikely to handle the anomalies that span beyond the window length. We are

interested in designing a novel anomaly detection algorithm that can address this problem.

How can we utilize the probabilistic modelling from VAEs to design a classifier that gives better calibrated uncertainty estimates? Uncertainty estimation is an important topic in machine learning research, as many of the machine learning models and algorithms are applied to applications where the stakes are high. These include autonomous driving, screening job applications and assisting doctors to make diagnosis decisions to name a few. We are particularly interested in uncertainty estimation for classification tasks. In regression tasks, the level of uncertainty is often described by a single parameter of a parametric distribution, such as the variance parameter in a Gaussian distribution, and can be easily calibrated. However uncertainty for classification tasks cannot be specified by a single parameter. Therefore, calibrating the uncertainty estimation of a classifier requires a radical change of the entire classifier design [35]. We measure the quality of the uncertainty estimation by a metric called calibration error. This metric measures the difference between the predicted confidence from a classifier and the empirical accuracy of these predictions. A perfectly calibrated classifier will have a calibration error of zero, meaning a prediction with 80% confidence made by this classifier has a chance of being correct 80% of the time. Most uncertainty estimation approaches operate by extending a deterministic classifier to (an approximation of) its Bayesian equivalent utilizing techniques such as Monte Carlo dropout or ensemble [30], [36]. We have seen promising results from these approaches on reducing the amount of over-confident predictions made by the original deterministic classifier without a Bayesian extension. However, these approaches still struggle with predicting sensible uncertainties when faced with input samples which are greatly different from the training examples, i.e. out of distribution (OOD) samples.

We believe that a classifier designed with a VAE module can be a good alternative to the MC dropout approach or the deep ensemble approach and can potentially overcome the problem with OOD samples. Firstly, VAEs offer a great framework for probabilistic modelling. With the VAE encoder, we can conveniently convert a data sample to a probability distribution in the latent space, giving the same effect as the MC dropout and ensemble. Secondly, VAEs are good at identifying OOD samples. We can utilize this capability and find ways to convert such detection to sensible distribution in the final class prediction. In this thesis, we are interested in exploring the possibilities in this direction and investigate how we can devise a classifier that gives more calibrated predictions.

1.2 Research contributions

In this thesis, we seek to answer each of the above questions. Our contributions are summarized as follows.

Firstly, we improve the generative modelling capacity of VAEs by tackling the two problems identified in the previous section: 1) an over-regularization issue and 2) an information drift effect.

To mitigate the over-regularizing behaviours, we first focus on the VAE models with a Gaussian decoder. We notice that the over-regularization is often caused by the difficulty of finding a good variance parameter for the decoder. We propose to infer the optimal value of this variance parameter by learning it directly from training data. We achieve this by optimizing this parameter under the ELBO objective together with the other model parameters. We show that VAEs trained with this variance parameter being inferred can generate better quality data samples and reach higher likelihood score. We also propose to extend the variance parameter from a single value shared by all the data samples in the given dataset to be variable across different input samples and data dimensions. The variance map inferred directly correlate to the reconstruction error and, hence, can be used to indicate uncertain regions in the data samples generated or reconstructed from a VAE.

In parallel to this line of research, we also address the over-regularization effect caused by an inflexible prior model in a VAE. We propose latent data distribution modelling with a generative prior (LaDDer). As the name suggests, we propose to adopt a flexible prior model implemented as a VAE in the latent space. To prevent the over-regularization to occur in the prior VAE, we adopt a Gaussian mixture prior for the prior VAE. We show that all the model parameters can be optimized in a unified variational inference framework under the ELBO objective. To efficiently update the model parameters, we can iterate across different modules, e.g. the outer data VAE, inner prior VAE and the hyper Gaussian mixture prior, and only update the parameters of a module at a time while keeping the rest unchanged. From extensive experiments, we show that our LaDDer model is able to accurately estimate complex latent data distribution and lead to VAE models that can reach higher likelihood score.

To mitigate the information drift phenomenon, we propose to modify the ELBO objective by adding a consistency loss term. This consistency loss penalizes the difference between the latent encoding of the original input data and the latent encoding of the reconstructed data, hence reducing the drifts caused by the auto-encoding function in the latent space. Taking advantage of the modelling assumption

of the Gaussian posteriors in the VAE models, we can compute the consistency loss term analytically. Hence, the gradients of all the model parameters are easy to be computed and training of the VAE models with the modified loss remains stable and efficient. Furthermore, the modified learning objective is a lower bound to the ELBO objective. The gap between the two objectives will be closed to zero when the two encodings of the original sample and the reconstructed sample are the same, i.e. the encoding of the original sample and the reconstructed sample contain exactly the same information. We show that with our consistency loss, the information loss associated with the auto-encoding mapping can be reduced and in some cases, the inferred VAE models reach a higher ELBO value, indicating a better fit to the given dataset.

Secondly, we design a VAE-LSTM hybrid model to detect anomalies in a time series. Our solution is crafted based on the assumption that normal behaviours in a time series are predictable. Hence, we train a model to predict the future readings of a time series based on the most recent readings. By comparing the difference in the prediction and the actual readings coming in, we can decide whether the current input has shown any sign of anomalies. To achieve this, we utilize the VAE’s capability to distill the most important features from the original data and use a VAE as a feature extractor for short-term windows of a given time series. These compact representations of the original time series allow us to ignore the noises commonly existing in the time series data. Coupling with a long short-term memory (LSTM) unit [37], [38] that makes prediction on the VAE latent features, we are able to capture the patterns that span over a much longer window than directly applying the LSTM on the raw data. Our experiments show that this anomaly detection scheme can effectively detect anomalies in the real world situations and outperform other baseline approaches.

Finally, we present a method for obtaining calibrated uncertainties using a hybrid classifier. This hybrid classifier combines a VAE module and a light-weight deterministic classification head. Given an input sample, our hybrid classifier first transforms the data sample to a posterior distribution in the latent space through the VAE’s encoder. To make a class prediction, we draw multiple samples from the posterior distribution and pass them through the deterministic classification head. The final predicted class distribution is obtained by averaging the predictions from all the samples. As the VAE’s encoder intrinsically captures the data noise and groups similar data samples in nearby regions in the latent space, such sampling will easily capture uncertainties that are caused by data samples residing close to the classification boundaries.

To capture the uncertainties associated with OOD samples, we need to extend the VAE’s encoder by considering two scenarios. If a data sample scores high ELBO (i.e. the data sample is likely generated from the data distribution), then the latent distribution is computed using the VAE’s encoder. If a data sample gives a low ELBO (i.e. this sample is likely to be out of data distribution), then we assign the latent distribution to be a wide, uninformative Gaussian. With data samples represented by such distributions, the prediction will be made by drawing multiple samples from the representation distribution and the uncertainty is encoded in the variance in the predictions coming from these samples. Our experiments show that the uncertainty estimated in our algorithm is better calibrated than other baseline approaches.

We list the peer-reviewed publications which are based on the work and analysis from this thesis as below:

1. LaDDer: Latent Data Distribution Modelling with a Generative Prior. **Shuyu Lin**, Ronald Clark. *The British Machine Vision Conference (BMVC)*. 2020.

Contributions: I proposed the main idea behind this publication, i.e. identifying discrepancy between the unit Gaussian prior and the aggregate posterior in a vanilla VAE model and proposing to replace the unit Gaussian prior with a generative prior to mitigate such effects. I implemented the proposed system and set up the experiments and evaluations. Ronald engaged in the discussion to better formulate the idea and helped with proofreading. The content related to this publication is included in Chapter 5.2 in this thesis and it has not been included in any other report that constitutes any co-authors’ degree.

2. Towards Consistent Variational Auto-Encoding. Yijing Liu*, **Shuyu Lin***, Ronald Clark. *Proceedings of the AAAI Conference on Artificial Intelligence (Student Abstract)*. 2020. *Equal contribution.

Contributions: I proposed the main idea behind this publication, i.e. identifying the drift between the posterior distributions if the auto-encoding function is repeatedly applied to the input sample and its reconstruction and proposing an additional consistency loss to mitigate such effects. Yijing and I set up the experiments included in the paper together. Ronald engaged in the discussion to better formulate the idea and helped with proofreading. The content related to the publication is included in Chapter 6 in this thesis and it has not been included in any other report that constitutes any co-authors’ degree.

3. Anomaly Detection for Time Series Using VAE-LSTM Hybrid Model. **Shuyu Lin**, Ronald Clark, Robert Birke, Sandro Schoonborn, Niki Trigoni, Stephen

Roberts. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2020.

Contributions: I proposed the main idea behind this publication, i.e. designing a VAE-LSTM hybrid system to detect anomalous events in a time series, implemented the proposed system and set up the experiments and evaluations. Robert and Sandro helped to identify anomaly detection as an important application domain. Ronald engaged in the discussion to better formulate the idea. All co-authors helped with proofreading. The content related to this publication is included in Chapter 7 in this thesis and it has not been included in any other report that constitutes any co-authors' degree.

4. Uncertainty Estimation with a VAE-Classifer Hybrid Model. **Shuyu Lin**, Ronald Clark, Niki Trigoni, Stephen Roberts. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2022.

Contributions: I proposed the main idea behind this publication, i.e. designing a hybrid system that contains a generative model to derive a probabilistic encoding distribution and a discriminative classifier to obtain better calibrated uncertainty estimation, implemented the proposed system and set up the experiments and evaluations. Ronald engaged in the discussion to better formulate the idea. All co-authors helped with proofreading. The content related to this publication is included in Chapter 8 in this thesis and it has not been included in any other report that constitutes any co-authors' degree.

The code that is related to the projects in this thesis is available at the following sites:

1. <https://github.com/lin-shuyu/encoder-decoder-mismatch>,
2. <https://github.com/lin-shuyu/ladder-latent-data-distribution-modelling>,
3. <https://github.com/lin-shuyu/VAE-LSTM-for-anomaly-detection>.

1.3 Thesis structure

We organize the rest of the thesis as follows.

In Chapter 2, we describe the variational inference technique, which is central to understanding the derivation of VAEs. As an illustrative example, we provide a step-by-step solution to the probabilistic principal component analysis (pPCA) problem using variational inference. In Chapter 3, we give a thorough introduction to VAEs. We show how VAEs are derived and trained given a dataset. We compare the difference between VAEs and other representation learning approaches, such as

GANs and normalizing flows. In Chapter 4, we give illustrative examples of the over-regularized posterior and the information drift effects. We explain what may cause such effects and why it is important to mitigate these behaviours. We present our solution to the over-regularization issue in Chapter 5. We present our solution to the information drift behaviours in Chapter 6. In Chapter 7, we focus on an important machine learning task - anomaly detection for a time series. We design a novel solution utilizing VAEs' capability to derive compact representations of the original data. We show our solution gives good performance on this task and outperforms baseline approaches. In Chapter 8, we address another important machine learning task - uncertainty estimation. We propose a classification framework that incorporates a VAE module to derive a probabilistic encoding function. Finally, we summarize the main contributions of this thesis in Chapter 9 and point out future research directions that can be extended from the work in thesis.

Prior works that introduce VAEs and relevant techniques to improve the generative modelling performance of VAEs are introduced and discussed in Chapter 2 and 3. These literatures form the reference foundation for the entire thesis. In Chapter 7 and 8, we introduce additional related works which are specific to the problem domains of anomaly detection and uncertainty estimation in Section 7.1 and Section 8.1.

1.4 Datasets

In this thesis, we carry out experiments on the following datasets:

- MNIST [39], a dataset that includes 70,000 handwritten digit images,
- Fashion MNIST [40], a dataset that includes 70,000 black-and-white clothes images,
- CelebA [41], a dataset that includes 200,000 RGB images of celebrity faces.
- Numenta anomaly benchmark [42], a dataset that includes 5 real-world time series which have the labels of anomaly events,
- HAM10k [43], a medical dataset that includes 10,015 dermatoscopic images and corresponding skin lesion labels.

In Chapter 4-6, we discuss challenges that are commonly encountered by VAE models regardless of downstream applications. Following the literatures in the related works, we carry out experiments with MNIST, Fashion MNIST and CelebA datasets. In Chapter 7, we design a system that detect anomalies in time series. Therefore, we evaluate our method's performance together with baseline approaches on Numenta

anomaly benchmark dataset. In Chapter 8, we propose a classification system that provides better calibrated uncertainty estimation. We carry out experiments on both MNIST and HAM10k dataset. We decide to include HAM10k dataset as part of the evaluation in this work, because to make a prediction with sensible uncertainty estimation is important in medical applications and HAM10k helps to demonstrate the message.

2

Background

Given a set of observations of the data variable $\mathbf{x}$, we are interested in discovering the distribution $p(\mathbf{x})$ from which the data samples are generated. This task is known as *density estimation*. For simple distributions, such as Gaussian distributions for continuous variables or multinomial distributions for discrete variables, we can approach this task by directly estimating the unknown parameters by maximizing the data likelihood. For more complex distributions, we often need to introduce a set of latent variables $\mathbf{z}$, which are not directly observable, but will allow us to define the joint distribution $p(\mathbf{x}, \mathbf{z})$. We can then obtain the data distribution $p(\mathbf{x})$ by integrating out $\mathbf{z}$. We refer to this type of models as latent variable models (LVMs). The learning task is to estimate the unknown parameters that define the joint distribution $p(\mathbf{x}, \mathbf{z})$. In addition, we are often interested in inferring the distribution of $\mathbf{z}$ given a data sample $\mathbf{x}$, i.e. $p(\mathbf{z}|\mathbf{x})$. This is referred to as the *inference* task in statistics.

In this chapter, we will cover the essential concepts needed for solving a density estimation task. Specifically, we will introduce important distributions, such as Gaussian and multinomial distributions, which will be used throughout this thesis. We will show how to estimate parameters in these distributions under the maximum likelihood objective. We will then show how to estimate parameters for latent variable models using *variational inference*. We will go through an example and show that variational inference can be used to solve a probabilistic principal component analysis (PCA) problem.

2.1 Important distributions

To approach the probability density estimation task, we first need to identify the random variable $\mathbf{x}$ - the probability distribution is estimated at different values of this variable. Depending on the nature of the random variable - discrete or continuous, unimodal or multimodal, we then define a model of the probability

distribution that we believe will give a good approximation of the target distribution. These models of the distributions are governed by a set of parameters, whose values will uniquely determine the distribution. To estimate these unknown model parameters, we will define an objective function that measures how well the current model given a particular set of parameters fits a set of observations which are the samples drawn from the underlying distribution. The set of model parameters that maximizes the objective function is our best estimate and determines our estimate of the target distribution.

In this section, we introduce two widely used probability distributions - the Gaussian distribution and the multinomial distribution. We demonstrate how to estimate the model parameters for these distributions.

2.1.1 Definition of a probability distribution

We first define the probability or probability distribution $p(\mathbf{x})$ of a random variable $\mathbf{x}$. We refer to the set of numbers that the random variable $\mathbf{x}$ is allowed to take as the input space or sample space Ω of the probability distribution. A subset X of Ω , i.e. $X \subseteq \Omega$, is often referred to as an event or an element in the sample space. The probability distribution $p(\mathbf{x})$ is a function that maps any element of the random variable $\mathbf{x}$ to a real number, i.e. $p(\mathbf{x}): X \mapsto \mathbb{R}$, and this function needs to satisfy the following three axioms [44]:

1. $p(X) \geq 0$ for every X , i.e. the probability is non-negative.
2. $p(\Omega) = 1$, i.e. the probabilities of all of the possible outcomes sum up to 1.
3. If two events X_1 and X_2 are mutually exclusive, i.e. $X_1 \cap X_2 = \emptyset$, then $p(X_1 \cup X_2) = p(X_1) + p(X_2)$.

Any function that satisfies the above definition is a valid probability distribution.

2.1.2 Discrete distributions

Now we introduce a few important probability distributions that we will use throughout this thesis. Firstly, we consider the case when the random variable $\mathbf{x}$ is binary, i.e. $\mathbf{x} \in \{0, 1\}$. There are plenty of occasions that we use such a random variable to describe the outcomes of the real-world events. For example, clinicians need to predict whether the diagnosis for a patient is positive or negative; juries make a decision whether a suspect is guilty or not for the charges raised against him/her; central banks decide whether they will raise the interest rate or not in response to the high inflation. We define the probability distribution of this binary

random variable as

$$\text{Bern}(\mathbf{x}|\mu) = \mu^{\mathbf{x}}(1 - \mu)^{1-\mathbf{x}}, \quad (2.1)$$

which is known as the Bernoulli distribution and the parameter μ describes the probability for the positive outcome, i.e.

$$p(x = 1|\mu) = \mu, \quad (2.2)$$

and $0 \leq \mu \leq 1$.

Given the probability distribution of a random variable, we can evaluate some basic statistics, such as the expectation and variance, of this random variable. Concretely, the expectation of a random variable $\mathbf{x}$ is defined as the average value of $\mathbf{x}$ under its probability distribution $p(\mathbf{x})$ and it is denoted as $\mathbb{E}[\mathbf{x}]$. For discrete variables,

$$\mathbb{E}[\mathbf{x}] = \sum_x p(\mathbf{x})\mathbf{x}. \quad (2.3)$$

For continuous variables,

$$\mathbb{E}[\mathbf{x}] = \int_x p(\mathbf{x})\mathbf{x} \, d\mathbf{x}. \quad (2.4)$$

The variance of $\mathbf{x}$ gives a measure of variability in the random variable $\mathbf{x}$ and is defined as

$$\text{var}[\mathbf{x}] = \mathbb{E}[(\mathbf{x} - \mathbb{E}[\mathbf{x}])^2] = \mathbb{E}[\mathbf{x}^2] - \mathbb{E}[\mathbf{x}]^2. \quad (2.5)$$

According to the definition in Equation (2.3) and (2.5), we can compute the mean and variance of the binary variable that follows the Bernoulli distribution in Equation (2.1) are

$$\mathbb{E}[\mathbf{x}] = \mu, \quad (2.6)$$

$$\text{var}[\mathbf{x}] = \mu(1 - \mu). \quad (2.7)$$

We can extend the Bernoulli distribution to a discrete variable that has more than two outcomes. We assume that there are K possible outcomes in total and each outcome is mutually exclusive with the other outcomes. We can express such variables using a K -dimensional vector $\mathbf{x}$ where only one of the elements $x_k = 1$ while the remaining elements are 0. Such representation is often referred to as one-hot encoding or 1-of- K encoding. We denote the probability of $x_k = 1$ by the parameter μ_k . Therefore, the overall distribution of the $\mathbf{x}$ with more than two outcomes is defined as

$$p(\mathbf{x}|\boldsymbol{\mu}) = \prod_{k=1}^K \mu_k^{x_k}, \quad (2.8)$$

where $\boldsymbol{\mu} = [\mu_1, \dots, \mu_K]$ contains K parameters that govern this distribution and μ_k is required to satisfy $\mu_k \geq 0$ and $\sum_k \mu_k = 1$. Similar to the binary case, we can compute the expectation according to Equation (2.3) that

$$\mathbb{E}[\mathbf{x}|\boldsymbol{\mu}] = \sum_{\mathbf{x}} p(\mathbf{x}|\boldsymbol{\mu})\mathbf{x} = \boldsymbol{\mu}. \quad (2.9)$$

2.1.3 Gaussian distributions

We need to consider a different set of probability distributions, when estimating the probability distribution for a continuous random variable $\mathbf{x}$. There are also abundant examples of continuous random variables in our lives. To name a few examples, we have the distribution of Apple's share price, the distribution of today's temperature and the distribution of the number of Covid infections in London tomorrow. A widely used distribution for modelling continuous random variables is the Gaussian distribution, which is also known as the normal distribution.

We give the definition of the Gaussian distribution below. When the random variable is a scalar, then Gaussian distribution is defined as

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}} \exp\left(-\frac{1}{2\sigma^2}(x - \mu)^2\right), \quad (2.10)$$

where μ is the mean of x and σ^2 is the variance of x . When the random variable is a D -dimensional vector, i.e. $\mathbf{x} \in \mathbb{R}^D$, we have the multivariate Gaussian distribution defined as below:

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{\frac{D}{2}}} \frac{1}{|\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right), \quad (2.11)$$

where $\boldsymbol{\mu}$ is a D -dimensional mean vector for $\mathbf{x}$, $\boldsymbol{\Sigma}$ is a $D \times D$ covariance matrix for $\mathbf{x}$ and $|\boldsymbol{\Sigma}|$ indicates the determinant of $\boldsymbol{\Sigma}$.

From the above definitions, we can see that the Gaussian distribution is governed by the two parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. For a general D -dimensional Gaussian distribution, there are $\frac{D(D+1)}{2}$ free parameters to be determined for the covariance matrix $\boldsymbol{\Sigma}$ and D free parameters for the mean vector $\boldsymbol{\mu}$. A common limitation of the Gaussian distribution is its inability to provide a good fit to multimodal distributions, as the Gaussian distribution is intrinsically unimodal. However, this limitation can be overcome by considering a (finite) mixture of Gaussians distribution instead. Furthermore, the Gaussian mixture model offers more flexibility in modelling heavy- and light-tailed distributions, which can be particularly useful for

real-world applications.

2.1.4 Maximum likelihood estimation

The majority of probability distributions we use in modelling are parametric, and hence defined by unknown parameters - in the case of the Gaussian, the mean and the (co)variance. Inference using these probability distributions, is therefore a matter of estimating the values of the parameters that govern the distribution. To do so, we need to define an objective function that can indicate how well a given distribution with a specific set of parameters approximates the true distribution.

Assuming we have a data set $\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ of N observed values of the random variable $\mathbf{x}$, we can compute the likelihood function under the assumption that the observations are drawn independently from the target distribution $p(\mathbf{x})$, as follows:

$$p(\mathcal{D}) = \prod_{i=1}^N p(\mathbf{x} = \mathbf{x}_i). \quad (2.12)$$

The independent assumption is very important, as it allows us to break the joint distribution of the entire dataset to a product of distributions of individual samples. We refer to the data points that are drawn independently from the same distribution as independent and identically distribute (i.i.d.) [12]. This is a common assumption we make in many machine learning tasks, as it significantly simplifies the computation of many objective functions.

In the example of the Bernoulli distribution, we have $p(\mathbf{x})$ defined in Equation (2.1). Therefore, the likelihood is:

$$p(\mathcal{D}) = \prod_{i=1}^N \mu^{x_i} (1 - \mu)^{1-x_i}, \quad (2.13)$$

which is function of the unknown parameter μ . Now we take the natural logarithm of the above likelihood function, we derive the log likelihood function as below:

$$\mathcal{L}(\mu) = \log p(\mathcal{D}) = \sum_{i=1}^N \{x_i \log \mu + (1 - x_i) \log(1 - \mu)\}. \quad (2.14)$$

The (log) likelihood function can serve as a good objective function, as it indicates how well the current choice of model parameters fits the given dataset $\mathcal{D}$. The set of model parameters that maximizes the (log) likelihood function is the maximum likelihood estimator of the model parameters. We obtain this set of parameters by

2. Background

computing the derivative of $\log \mathcal{L}(\mu)$ with respect to μ :

$$\frac{\partial \mathcal{L}(\mu)}{\partial \mu} = \frac{1}{\mu} \sum_{i=1}^N x_i - \frac{1}{1-\mu} \sum_{i=1}^N (1-x_i), \quad (2.15)$$

and set this derivative to zero. This gives us the maximum likelihood estimator of μ as:

$$\mu_{\text{ML}} = \frac{1}{N} \sum_{i=1}^N x_i. \quad (2.16)$$

In the example of estimating the model parameters in a multivariate Gaussian distribution, we again consider the log likelihood function, which now becomes:

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \log p(\mathcal{D}) = -\frac{ND}{2} \log 2\pi - \frac{N}{2} \log |\boldsymbol{\Sigma}| - \frac{1}{2} \sum_{i=1}^N (\mathbf{x}_i - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_i - \boldsymbol{\mu}), \quad (2.17)$$

after substituting the Gaussian distribution defined in Equation (2.11) in $p(\mathbf{x})$.

To find the maximum likelihood estimator of the Gaussian parameters, we need to compute the derivative of the log likelihood function with respect to each of the model parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. Using Equation (A.1) from Appendix, we can compute the derivative of the log likelihood function with respect to $\boldsymbol{\mu}$ as:

$$\frac{\partial \mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \boldsymbol{\mu}} = \sum_{i=1}^N \boldsymbol{\Sigma}^{-1} (\mathbf{x}_i - \boldsymbol{\mu}). \quad (2.18)$$

Setting this derivative to zero, we obtain the maximum likelihood estimate of the mean vector as:

$$\boldsymbol{\mu}_{\text{ML}} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i. \quad (2.19)$$

To compute the derivative of the log likelihood function with respect to $\boldsymbol{\Sigma}$, we use the matrix derivative identity (A.3) and (A.4) and obtain:

$$\frac{\partial \mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \boldsymbol{\Sigma}} = -\frac{N}{2} \boldsymbol{\Sigma}^{-T} + \frac{1}{2} \sum_{i=1}^N \boldsymbol{\Sigma}^{-T} (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-T}. \quad (2.20)$$

Again, we set this derivative to zero and obtain the maximum likelihood estimate of the covariance parameter as:

$$\boldsymbol{\Sigma}_{\text{ML}} = \frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T. \quad (2.21)$$

Note that $\boldsymbol{\Sigma}_{\text{ML}}$ is dependent on the mean parameter $\boldsymbol{\mu}$. We will substitute $\boldsymbol{\mu}_{\text{ML}}$ to

compute Σ_{ML} .

2.2 Graphical models

In the previous section, we have shown that parametric probability models can be used to estimate the probability distribution of a random variable. When we consider a scenario where multiple random variables co-exist and dependencies between these variables emerge, we might find it beneficial to have a diagrammatic representation of the joint probability of all the random variables concerned. Such a representation helps to visualize the dependencies between various variables and, hence, simplifies the computation of the joint probability. The probabilistic graphical models, derived from the well-known data structure - graphs, are invented for this purpose.

In a probabilistic graphical model, nodes represent random variables and links (or edges) represent probabilistic relationships between the connected variables [12]. In this thesis, we particularly focus on directed graphical models (also known as Bayesian networks), where the links of the graphs have directionality and indicate the particular dependencies of the conditional probabilities needed to define the joint probability. For example, in the directed graphical model shown in Figure 2.1, we see a link that goes from $\mathbf{x}_1$ to $\mathbf{x}_3$. We call the node at the start of the link the parent node of the node at the end of the link, i.e. $\mathbf{x}_1$ is the parent node of $\mathbf{x}_3$. Correspondingly, $\mathbf{x}_3$ is the child node of $\mathbf{x}_1$. The joint probability of all the variables represented in a graph can be written as the product of the conditional distribution for each node conditioned on all of its parent variables. This is known as the *factorization* property of the joint probability for a directed graphical model [12]. Therefore, in this example, we have:

$$p(\mathbf{x}_1, \dots, \mathbf{x}_5) = p(\mathbf{x}_1)p(\mathbf{x}_2)p(\mathbf{x}_3|\mathbf{x}_1, \mathbf{x}_2)p(\mathbf{x}_4|\mathbf{x}_1)p(\mathbf{x}_5|\mathbf{x}_2). \quad (2.22)$$

In general, for a graph with K nodes, the joint probability of all the K variables is defined as:

$$p(\mathbf{x}_1, \dots, \mathbf{x}_K) = \prod_{k=1}^K p(\mathbf{x}_k|\text{pa}_k), \quad (2.23)$$

where pa_k represents all the parent variables of node $\mathbf{x}_k$.

2.2.1 Sampling in a directed graphical model

Very often we need to draw samples of the data according to the data distribution. The sampling can be easily done if we obtain the graphical model of the

corresponding data distribution $p(\mathbf{x})$. Here $\mathbf{x} = \{\mathbf{x}_1, \dots, \mathbf{x}_K\}$ denotes a set of K data variables. According to the factorization rule in Equation (2.23), $p(\mathbf{x})$ can be factorized into K conditional distributions $p(\mathbf{x}_k|\mathbf{pa}_k)$.

To draw a sample $\hat{\mathbf{x}} = [\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_K]^T$, we start with the lowest-indexed node $\mathbf{x}_1$ and draw a sample from its distribution $p(\mathbf{x}_1)$. We denote this sample as $\hat{\mathbf{x}}_1$. We then go through each node in order and draw samples according to the conditional distribution $p(\mathbf{x}_k|\mathbf{pa}_k)$, where the parent nodes are taken the sampled value. For example, when we draw a sample from the distribution represented in Figure 2.1, we first draw samples of $\mathbf{x}_1$ and $\mathbf{x}_2$ from their distributions, i.e. $\hat{\mathbf{x}}_1 \sim p(\mathbf{x}_1)$ and $\hat{\mathbf{x}}_2 \sim p(\mathbf{x}_2)$, as they are root nodes (nodes without any parent nodes). Then we move to $\mathbf{x}_3$, which has one parent node $\mathbf{x}_1$. We now draw sample of $\mathbf{x}_3$ according to the conditional distribution $p(\mathbf{x}_3|\mathbf{x}_1 = \hat{\mathbf{x}}_1)$, giving the sampled value $\hat{\mathbf{x}}_3$. The same applies to $\mathbf{x}_4$ and $\mathbf{x}_5$, i.e. $\hat{\mathbf{x}}_4 \sim p(\mathbf{x}_4|\mathbf{x}_1 = \hat{\mathbf{x}}_1, \mathbf{x}_2 = \hat{\mathbf{x}}_2)$ and $\hat{\mathbf{x}}_5 \sim p(\mathbf{x}_5|\mathbf{x}_2 = \hat{\mathbf{x}}_2)$. Now we collect all the sampled values of each node and we obtain the sampled data point $\hat{\mathbf{x}} = [\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_5]^T$.

2.2.2 Conditional independence

Conditional independence can provide great convenience when dealing with probability distributions that involve multiple random variables. If two variables are conditionally independent given some other variables, then we can factorize the joint probability of these two variables into a product of the distributions of each variable conditioned on the related other variables. This breaks a high-dimensional distribution into two lower-dimensional distributions, making the modelling easier.

Conditional independence comes in two forms. Let's consider three random variables $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$. If we show that the conditional distribution of $\mathbf{a}$ given $\mathbf{b}$ and $\mathbf{c}$ is equivalent to the conditional distribution of $\mathbf{a}$ given $\mathbf{c}$, i.e.

$$p(\mathbf{a}|\mathbf{b}, \mathbf{c}) = p(\mathbf{a}|\mathbf{c}), \quad (2.24)$$

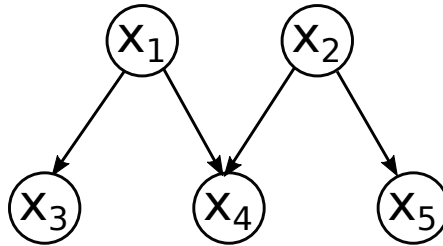


Figure 2.1: A directed graphical model representing the joint probability over 5 random variables $\mathbf{x}_1 - \mathbf{x}_5$.

then $\mathbf{a}$ is conditionally independent of $\mathbf{b}$ given $\mathbf{c}$. The second form of conditional independence is expressed by the factorization rule. If we can show the joint distribution of $\mathbf{a}$ and $\mathbf{b}$ given $\mathbf{c}$ factorizes into the conditional distributions of $\mathbf{a}$ and $\mathbf{b}$ on its own given $\mathbf{c}$, i.e.

$$p(\mathbf{a}, \mathbf{b}|\mathbf{c}) = p(\mathbf{a}|\mathbf{c})p(\mathbf{b}|\mathbf{c}), \quad (2.25)$$

then $\mathbf{a}$ is conditionally independent of $\mathbf{b}$ given $\mathbf{c}$. We use the symbol $\perp$ to indicate the conditional independence relationship between different random variables. For example, we can express $\mathbf{a}$ and $\mathbf{b}$'s conditional independence as $\mathbf{a} \perp \mathbf{b}|\mathbf{c}$.

Graphical models are a convenient tool to derive conditional independence between random variables. Figure 2.2(a) defines a graphical model over three random variables $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$, where the joint distribution can be factorized as:

$$p(\mathbf{a}, \mathbf{b}, \mathbf{c}) = p(\mathbf{a}|\mathbf{c})p(\mathbf{b}|\mathbf{c})p(\mathbf{c}). \quad (2.26)$$

We also know that

$$p(\mathbf{a}, \mathbf{b}, \mathbf{c}) = p(\mathbf{a}, \mathbf{b}|\mathbf{c})p(\mathbf{c}). \quad (2.27)$$

Therefore,

$$p(\mathbf{a}, \mathbf{b}|\mathbf{c}) = p(\mathbf{a}|\mathbf{c})p(\mathbf{b}|\mathbf{c}). \quad (2.28)$$

So in this graphical model, $\mathbf{a}$ is conditionally independent of $\mathbf{b}$ given $\mathbf{c}$, i.e. $\mathbf{a} \perp \mathbf{b}|\mathbf{c}$.

It is worth noting that $\mathbf{a}$ and $\mathbf{b}$ are not independent without the conditioning on $\mathbf{c}$, i.e. $\mathbf{a} \not\perp \mathbf{b}$. We can see this by examining the joint probability $p(\mathbf{a}, \mathbf{b})$, which can be obtained by integrating out $\mathbf{c}$ in the joint probability $p(\mathbf{a}, \mathbf{b}, \mathbf{c})$ from Equation (2.26), i.e.

$$p(\mathbf{a}, \mathbf{b}) = \int_{\mathbf{c}} p(\mathbf{a}|\mathbf{c})p(\mathbf{b}|\mathbf{c})p(\mathbf{c})d\mathbf{c}. \quad (2.29)$$

The above joint distribution $p(\mathbf{a}, \mathbf{b})$ does not factorize into the product $p(\mathbf{a})p(\mathbf{b})$. Hence, $\mathbf{a}$ and $\mathbf{b}$ are not independent in general. After conditioning on a common factor $\mathbf{c}$, $\mathbf{a}$ and $\mathbf{b}$ become independent and their joint probability given the common factor now factorizes.

The conditional independence demonstrated in Figure 2.2(a) can be found in many scenarios. We give a few examples in the list below [45]:

- Amount of speeding fine $\perp$ Make of the car | Speed
- Lung cancer $\perp$ Yellow teeth | Smoking (corresponding to Figure 2.2(b))
- Is the car's motor working? $\perp$ Is the car's air conditioner working? | Is the car's battery flat?

In all of the above examples, the two variables without being conditioned on the

common factor are not independent. For example, whether a person is likely to get lung cancer may be highly correlated to whether the person has yellow teeth. This is because people who smoke a lot often have yellow teeth. Given we know about whether the person's smoking preference, then the two variables - the chance of the person getting lung cancer and the chance of the person has yellow teeth - are no longer correlated. We will see in the next section that the conditional independence derived between two variables which share a common factor can be very helpful in a latent variable model and it can greatly simplify the probability model to be estimated.

2.3 Latent variable models

An important type of graphical models is latent variable models (LVMs) [46]. We often build a graphical model over a set of random variables that we can take observations of. Latent variables refer to the random variables that are not available for observations. Hence, latent variables are also referred to as hidden variables. As we do not have access to the actual values of the latent variables, there is often a great level of flexibility on how to integrate the latent variables into the graphical model, i.e. where we insert the latent variables in a graph of other variables and what connections we link the latent variables to. All these decisions can be made by the designer of the latent variable model and are often chosen so the corresponding models can be easily estimated.

Adding additional random variables to the already challenging probability density estimation task might be counter-intuitive. There are some great benefits of adopting the latent variable models. We will discuss two such benefits.

Firstly, introducing the latent variables into an existing graphical model might simplify the estimation task. We show an example of this effect in Figure 2.3. In

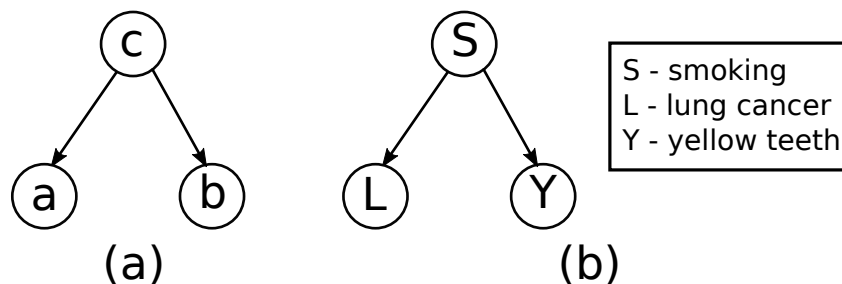


Figure 2.2: Conditional independence over three random variables. (a) A directed graphical model over three random variables a , b and c . (b) An example of conditional independence in a real-world scenario.

the subplot (a), we have a directed graphical model that contains 6 data variables $\mathbf{x}_1, \dots, \mathbf{x}_6$. In this graphical model, the overall joint probability is defined as:

$$p(\mathbf{x}_1, \dots, \mathbf{x}_6) = p(\mathbf{x}_1)p(\mathbf{x}_2)p(\mathbf{x}_3)p(\mathbf{x}_4|\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)p(\mathbf{x}_5|\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)p(\mathbf{x}_6|\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3). \quad (2.30)$$

Assuming all the random variables are binary and we express all the conditional probabilities in probability distribution tables, we have 24 unknown parameters to solve to determining this probability distribution.

Now we introduce a latent variable $\mathbf{h}$ into the model. As we have the flexibility to decide where and how to add the latent variable, we choose to have $\mathbf{h}$ being a common descendent of the three parent nodes - $\mathbf{x}_1, \mathbf{x}_2$ and $\mathbf{x}_3$ - and a common parent to the three leaf nodes - $\mathbf{x}_4, \mathbf{x}_5$ and $\mathbf{x}_6$, as shown in the subfigure (b). Such a design is sensible, because the original graph indicates that $\mathbf{x}_4, \mathbf{x}_5$ and $\mathbf{x}_6$ are caused by $\mathbf{x}_1, \mathbf{x}_2$ and $\mathbf{x}_3$ and the new latent variable model respects this nature of the original graph and allow the influence of $\mathbf{x}_1, \mathbf{x}_2$ and $\mathbf{x}_3$ to pass onto $\mathbf{x}_4, \mathbf{x}_5$ and $\mathbf{x}_6$ via an unobservable latent factor. In this new latent variable model, the overall joint probability is defined as:

$$p(\mathbf{x}_1, \dots, \mathbf{x}_6, \mathbf{h}) = p(\mathbf{x}_1)p(\mathbf{x}_2)p(\mathbf{x}_3)p(\mathbf{h}|\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)p(\mathbf{x}_4|\mathbf{h})p(\mathbf{x}_5|\mathbf{h})p(\mathbf{x}_6|\mathbf{h}). \quad (2.31)$$

With the additional $\mathbf{h}$, we end up with a joint probability that decomposes into more conditional probabilities compared to the original model. However, the total number of the unknown parameters is reduced. This is because the conditional distributions $p(\mathbf{x}_i|\mathbf{h})$ for $i = 4, 5, 6$ are much simpler than the original conditional distributions for these random variables. For example $p(\mathbf{x}_4|\mathbf{h})$ only needs 1 parameter, whereas $p(\mathbf{x}_4|\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ needs 7 parameters to specify. Therefore, the total number of model parameters reduces to 14, which is almost halved from the original model.

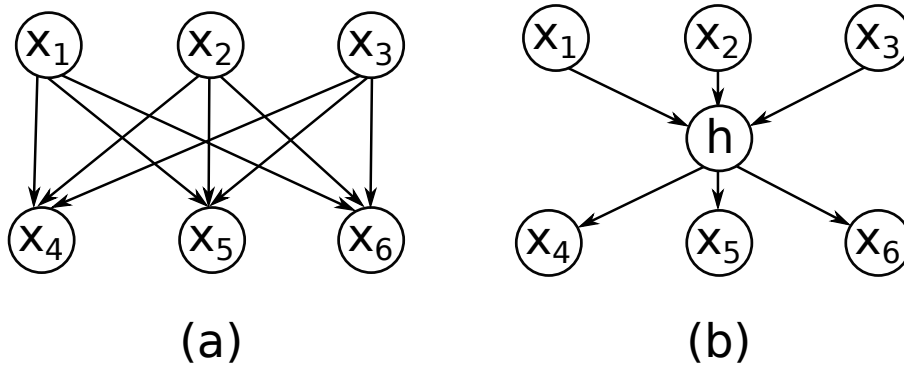


Figure 2.3: Converting a directed graphical model in (a) into a latent variable model in (b). The random variable $\mathbf{h}$ denotes the latent variable.

Secondly, if we choose to adopt a latent variable $\mathbf{h}$ that is of lower dimension than the data variables $\mathbf{x}$ which $\mathbf{h}$ is dependent on, then we can see this latent variable as an information *bottleneck*, which serves as a compressed representation of $\mathbf{x}$. Having such a bottleneck is favoured in many machine learning models and algorithms, as it requires the models to focus on the important factors rather than paying attention to nuances in the data, which are likely to correspond to noise. Hence, the corresponding models are more likely to generalize to new observations.

2.4 Variational inference

Now let's consider a general latent variable model with a D_z dimensional latent variable $\mathbf{z}$ and a D_x dimensional data variable $\mathbf{x}$. If we define the joint distribution as below:

$$p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z}), \quad (2.32)$$

where $p_\theta(\mathbf{z})$ denotes the prior distribution for the latent variable $\mathbf{z}$, $p_\theta(\mathbf{x}|\mathbf{z})$ denotes the conditional distribution between the data and the latent variable and θ denotes the set of model parameters that define the two distributions - $p_\theta(\mathbf{z})$ and $p_\theta(\mathbf{x}|\mathbf{z})$, the corresponding directed graphical model of this latent variable model is shown in Figure 2.4(a). Our task here is to estimate the values of the model parameters θ and to estimate the posterior distribution $p(\mathbf{z}|\mathbf{x})$ so we can infer the values of $\mathbf{z}$ given an observation of the data.

Why is this task so difficult? We can see this by deriving an expression for the posterior distribution as below:

$$p(\mathbf{z}|\mathbf{x}) = \frac{p_\theta(\mathbf{z}, \mathbf{x})}{p(\mathbf{x})} = \frac{p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})}{p(\mathbf{x})} = \frac{p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})}{\int_{\mathbf{z}} p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})d\mathbf{z}}. \quad (2.33)$$

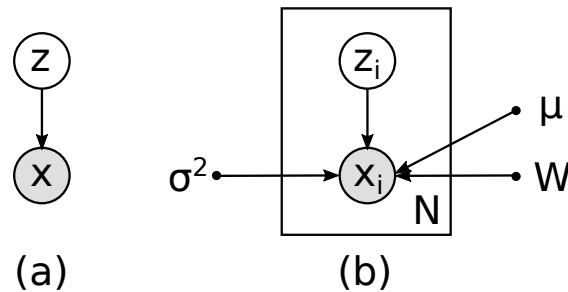


Figure 2.4: The graphical model for a probabilistic PCA model. (a) The graphical model for the latent variable model expressed in Equation (2.32). (b) The graphical model for a pPCA model where N data samples are i.i.d. and $\{\mathbf{W}, \boldsymbol{\mu}, \sigma\}$ are the model parameters.

To compute the posterior distribution, we need to first find out θ . Even if we have θ at hand, the computation of the posterior is not trivial. The difficulty lies in the computation of the integration which appears in the denominator of the furthest right term in Equation (2.33). Analytical solution for this integration is often not available for most distributions $p_\theta(\mathbf{x}|\mathbf{z})$ and $p_\theta(\mathbf{z})$ and the numerical approximation for this integration can be expensive if the dimensions of $\mathbf{x}$ or $\mathbf{z}$ are large. Without this integration, we cannot compute the posterior distribution directly. Not to mention, estimating the values for θ is not a straight-forward task either.

Variational inference [18], [19], [47] converts this task into an optimization problem. To do so, we first introduce an approximate distribution $q_\phi(\mathbf{z})$ which is parameterized by a set of parameters ϕ . Rather than directly solving for $p(\mathbf{z}|\mathbf{x})$, we now aim to find $q^*(\mathbf{z})$ that is closest to the true posterior $p(\mathbf{z}|\mathbf{x})$, i.e.

$$q^*(\mathbf{z}) = \arg \min_{\phi} D_{\text{KL}}(q_\phi(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x})), \quad (2.34)$$

where $D_{\text{KL}}(q \parallel p)$ is defined as:

$$D_{\text{KL}}(q(\mathbf{u}) \parallel p(\mathbf{u})) = \int q(\mathbf{u}) \log \frac{q(\mathbf{u})}{p(\mathbf{u})} d\mathbf{u} = \mathbb{E}_{q(\mathbf{u})} \left[\log \frac{q(\mathbf{u})}{p(\mathbf{u})} \right], \quad (2.35)$$

which measures the distance between the two distributions $q(\mathbf{u})$ and $p(\mathbf{u})$. Then $q^*(\mathbf{z})$ is the best approximation of the posterior $p(\mathbf{z}|\mathbf{x})$ within the distribution family of our modelling choice for $q_\phi(\mathbf{z})$.

2.4.1 Evidence lower bound objective

In practice, it is difficult to directly optimize the $D_{\text{KL}}(q_\phi(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x}))$ in Equation (2.34), as we do not know the posterior distribution $p(\mathbf{z}|\mathbf{x})$. To go around this problem, variational inference [18], [19], [47] resorts to an alternative objective that can be computed given the current parameters θ and ϕ . To derive the alternative objective, we re-arrange the Kullback-Leibler (KL) divergence in Equation (2.34) into the current form:

$$D_{\text{KL}}(q_\phi(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x})) = \mathbb{E}_{q_\phi(\mathbf{z})} [\log q_\phi(\mathbf{z})] - \mathbb{E}_{q_\phi(\mathbf{z})} [\log p_\theta(\mathbf{z}, \mathbf{x})] + \log p(\mathbf{x}). \quad (2.36)$$

Re-arranging the last line in Equation (2.36), we obtain an expression which we refer to as evidence lower bound (ELBO), as shown below:

$$\text{ELBO}(\mathbf{x}; \phi, \theta) = \mathbb{E}_{q_\phi(\mathbf{z})} [\log p_\theta(\mathbf{z}, \mathbf{x})] - \mathbb{E}_{q_\phi(\mathbf{z})} [\log q_\phi(\mathbf{z})] \quad (2.37)$$

$$= \log p(\mathbf{x}) - D_{\text{KL}}(q_\phi(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x})). \quad (2.38)$$

It is clear that ELBO is a function of the model parameters of ϕ and θ . The name of evidence lower bound comes from Equation (2.38). As $D_{\text{KL}}(q_\phi(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x}))$ is non-negative, ELBO is an effective lower bound to the log likelihood of the data distribution $\log p(\mathbf{x})$ (and $p(\mathbf{x})$ is often referred to as an evidence term), i.e.

$$\log p(\mathbf{x}) \geq \text{ELBO}(\mathbf{x}; \phi, \theta). \quad (2.39)$$

The equality of the above equation only takes place when $D_{\text{KL}}(q_\phi(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x})) = 0$, i.e. $q_\phi(\mathbf{z}) = p(\mathbf{z}|\mathbf{x})$. This indicates that maximizing ELBO is equivalent to minimizing the $D_{\text{KL}}(q_\phi(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x}))$ term required by the variational inference objective. As we can evaluate ELBO for any given values of θ and ϕ , we have converted the original objective expressed in Equation (2.34) to finding the set of θ and ϕ that maximizes ELBO. The ELBO objective is widely used to solve a variational inference task.

2.4.2 Optimizing ELBO

To maximize the ELBO defined in Equation (2.37), we take the derivative of ELBO with respect to every model parameters in θ and ϕ while keeping the other parameters at their current values and set the derivative to zero, i.e.

$$\frac{\partial \text{ELBO}}{\partial \theta^{(i)}} = 0, \quad \text{for } i = 1, \dots \quad (2.40)$$

$$\frac{\partial \text{ELBO}}{\partial \phi^{(j)}} = 0, \quad \text{for } j = 1, \dots \quad (2.41)$$

If we can obtain the analytical solutions for the above equations, then we can reach the maximum of ELBO quite efficiently, as we approach to a local maximum of ELBO at every update. When there are multiple solutions for the above equation, care needs to be taken to choose the value that gives the greatest local maxima. This update needs to be carried out at every parameter over multiple iterations until the value of ELBO has converged.

If computing the analytical solution for Equation (2.40) and (2.41) turns out to be too difficult, then we can take an iterative approach, such as gradient ascent [22], [23], to maximize ELBO. To do so, we compute the gradient of ELBO with respect

to all the parameters at the current parameter values, i.e.

$$\Delta\theta^{(i)} = \left. \frac{\partial \text{ELBO}}{\partial \theta^{(i)}} \right|_{\theta, \phi} \quad \text{for } i = 1, \dots \quad (2.42)$$

$$\Delta\phi^{(j)} = \left. \frac{\partial \text{ELBO}}{\partial \phi^{(j)}} \right|_{\theta, \phi} \quad \text{for } j = 1, \dots \quad (2.43)$$

We then update the parameter in turn with a step change along the positive gradient direction, i.e.

$$\theta_{t+1}^{(i)} = \theta_t^{(i)} + \lambda \times \Delta\theta^{(i)}, \quad \text{for } i = 1, \dots \quad (2.44)$$

$$\phi_{t+1}^{(j)} = \phi_t^{(j)} + \lambda \times \Delta\phi^{(j)}, \quad \text{for } j = 1, \dots \quad (2.45)$$

Here λ indicates the step size and is a hyperparameter that needs to be specified and tuned. With this iterative optimization approach, every update only increases ELBO. Compared to the update of the exact solution, this optimization will often take more updates to converge.

2.4.3 The mean-field assumption

When the dimension of the latent variable $\mathbf{z}$ is high, the expectation with respect to its distribution $q(\mathbf{z})$ can be difficult to evaluate. As shown in Equation (2.37), this expectation is needed to compute ELBO and we need to be able to compute this expectation efficiently as the optimization of ELBO needs to compute this expectation repeatedly. One way to achieve this goal is to introduce some additional assumption to the variational distribution $q_\phi(\mathbf{z})$. A commonly used assumption is that the latent variables are mutually independent. Hence, $q_\phi(\mathbf{z})$ can be factorized into the product of distributions of individual latent variables, i.e.

$$q_\phi(\mathbf{z}) = \prod_{j=1}^M q_{\phi_j}(\mathbf{z}^{(j)}), \quad (2.46)$$

where the superscript j indicates the j -th latent variable, which is stored at the j -th dimension of the latent vector; M indicates that the number of latent variables (i.e. $M = D_z$ the dimension of the latent variable) in our model. We refer to the distribution defined in Equation (2.46) as a mean-field distribution.

The mean-field assumption allows us to simplify the ELBO computation signif-

icantly. For example, the entropy term $\mathbb{E}_{q_\phi(\mathbf{z})}[\log q_\phi(\mathbf{z})]$ can be simplified to:

$$\mathbb{E}_{q_\phi(\mathbf{z})}[\log q_\phi(\mathbf{z})] = \sum_{j=1}^M \mathbb{E}_{q_{\phi_j}(\mathbf{z}^{(j)})}[\log q_{\phi_j}(\mathbf{z}^{(j)})]. \quad (2.47)$$

The full derivation of the above equation is given in Appendix A.3.

The expectation of the joint distribution $\mathbb{E}_{q_\phi(\mathbf{z})}[\log p_\theta(\mathbf{z}, \mathbf{x})]$ can also be simplified using the mean-field assumption. However, the exact form of this term depends on the specific model set up for the joint distribution, e.g. the choice of model for the conditional distribution $p_\theta(\mathbf{x}|\mathbf{z})$. Hence, this term differs from one problem to another. We will show an example of how to compute this term efficiently in the case of a probabilistic PCA in the next section.

2.5 An example: probabilistic PCA

In this section, we show how variational inference is used to solve a well-known probabilistic model - a probabilistic principal component analysis (pPCA) problem [48]. In pPCA, the latent variable $\mathbf{z}$ follows a zero-mean unit-covariance Gaussian distribution, i.e.

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z} | \mathbf{0}, \mathbf{I}), \quad (2.48)$$

and the data variable is obtained under a linear mapping from the latent variable $\mathbf{z}$, subject to perturbation from some white noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$, i.e.

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon}, \quad (2.49)$$

where $\mathbf{W}$ is a $D_x \times D_z$ matrix and $\boldsymbol{\mu}$ is a D_x -dimensional column vector, which together govern the linear mapping. $\boldsymbol{\epsilon}$ denotes the random noise that is injected in the mapping. From Equation (2.49), we know that the conditional distribution $p(\mathbf{x}|\mathbf{z})$ is also a Gaussian, i.e.

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x} | \mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \sigma^2 \mathbf{I}). \quad (2.50)$$

Given a dataset of N observations, i.e. $\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, we can use Equation (2.48) and (2.49) to model the generative process of this dataset. To generate a data sample $\mathbf{x}_i$, we will first sample a value of the latent variable $\mathbf{z}_i$ and then obtain the data observation by sampling from the conditional distribution defined by $\mathbf{z}_i$. This generative process is shown in Figure 2.4(b). As the generation of the N data samples can be assumed to be identically independently distributed (i.i.d.), we

encapsulate the nodes for N different data-latent pairs with a plague which indicates the repetition of the same process by N times. Now our learning task is to estimate the model parameters $\mathbf{W}$, $\boldsymbol{\mu}$, and σ that governs this generative process and to infer the posterior distributions of the latent variable given each data sample, i.e. $p(\mathbf{z}_i|\mathbf{x}_i)$.

2.5.1 Model definition

The first step in a variational inference framework is to clearly define the probabilistic models that are required to compute the ELBO objective given in Equation (2.37). Specifically, we need to define the parametric forms of the distributions, including the prior distribution $p(\mathbf{z})$, the conditional distribution $p(\mathbf{x}|\mathbf{z})$ and the variational posterior distribution $q(\mathbf{z}_i)$ (from here onward we use $q(\mathbf{z}_i)$ to refer to the posterior distribution $p(\mathbf{z}_i|\mathbf{x}_i)$).

Firstly, the prior distribution $p(\mathbf{z})$ is fully specified in a pPCA model. According to the definition of Gaussian distributions given in Equation (2.11), we have

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}) = (2\pi)^{-\frac{D_z}{2}} \exp\left(-\frac{1}{2}\mathbf{z}^T\mathbf{z}\right). \quad (2.51)$$

Secondly, the conditional distribution also follows a multivariate Gaussian distribution, which is governed by the model parameter $\mathbf{W}$, $\boldsymbol{\mu}$ and σ , i.e.

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \sigma^2\mathbf{I}) \quad (2.52)$$

$$= (2\pi)^{-\frac{D_x}{2}} \sigma^{-D_x} \exp\left(-\frac{1}{2\sigma^2}(\mathbf{x} - (\mathbf{W}\mathbf{z} + \boldsymbol{\mu}))^T(\mathbf{x} - (\mathbf{W}\mathbf{z} + \boldsymbol{\mu}))\right). \quad (2.53)$$

Finally, we define the variational posterior distribution $q(\mathbf{z})$ also as a multivariate Gaussian distribution. With the mean-field assumption, the overall posterior $q(\mathbf{z})$ can be factorized over the posteriors for each data sample, as follows:

$$q(\mathbf{z}) = \prod_{i=1}^N q(\mathbf{z}_i), \quad (2.54)$$

$$\text{where } q(\mathbf{z}_i) = \mathcal{N}(\mathbf{z}_i|\mathbf{m}_i, s_i^2\mathbf{I}) \quad (2.55)$$

$$= (2\pi)^{-\frac{D_z}{2}} s_i^{-D_z} \exp\left(-\frac{1}{2s_i^2}(\mathbf{z}_i - \mathbf{m}_i)^T(\mathbf{z}_i - \mathbf{m}_i)\right), \quad (2.56)$$

where $\mathbf{m}_i$ and s_i^2 are the mean and variance of the posterior distribution of the i -th data sample $\mathbf{x}_i$. With the above probabilistic models, we have the following parameters to estimate: the generative model parameters $\theta = \{\mathbf{W}, \boldsymbol{\mu}, \sigma\}$ and the variational model parameters $\phi_i = \{\mathbf{m}_i, s_i^2\}$, $\forall i = 1, \dots, N$.

2.5.2 Deriving ELBO objective

We now derive the expression for the ELBO objective in this pPCA model specified in the previous section. Recall that ELBO is defined as:

$$\text{ELBO}(\mathbf{x}; \phi, \theta) = \mathbb{E}_{q_\phi(\mathbf{z})} [\log p_\theta(\mathbf{z}, \mathbf{x})] - \mathbb{E}_{q_\phi(\mathbf{z})} [\log q_\phi(\mathbf{z})]. \quad (2.57)$$

We can further decompose $\log p_\theta(\mathbf{z}, \mathbf{x})$ under the assumption that the N data samples are i.i.d. Then we have:

$$\log p_\theta(\mathbf{z}, \mathbf{x}) = \sum_{i=1}^N \log p_\theta(\mathbf{x}_i | \mathbf{z}_i) + \sum_{i=1}^N \log p(\mathbf{z}_i) \quad (2.58)$$

This allows us to simplify the joint log likelihood term in ELBO as:

$$\mathbb{E}_{q_\phi(\mathbf{z})} [\log p_\theta(\mathbf{z}, \mathbf{x})] = \sum_{i=1}^N \mathbb{E}_{q_\phi(\mathbf{z})} [\log p_\theta(\mathbf{x}_i | \mathbf{z}_i)] + \sum_{i=1}^N \mathbb{E}_{q_\phi(\mathbf{z})} [\log p(\mathbf{z}_i)]. \quad (2.59)$$

We have brought the expectation inside the summation in the above equation, as expectation is a linear operator. We can further decompose the expectation with respect to $q_\phi(\mathbf{z})$ to the expectation with respect to each latent sample distribution $q_\phi(\mathbf{z}_i)$, as shown below:

$$\sum_{i=1}^N \mathbb{E}_{q_\phi(\mathbf{z})} [\log p_\theta(\mathbf{x}_i | \mathbf{z}_i)] = \sum_{i=1}^N \mathbb{E}_{q_\phi(\mathbf{z}_i)} [\log p_\theta(\mathbf{x}_i | \mathbf{z}_i)] \quad (2.60)$$

$$\sum_{i=1}^N \mathbb{E}_{q_\phi(\mathbf{z})} [\log p(\mathbf{z}_i)] = \sum_{i=1}^N \mathbb{E}_{q_\phi(\mathbf{z}_i)} [\log p(\mathbf{z}_i)]. \quad (2.61)$$

More details of this derivation is given in Appendix A.3. We now substitute the definition of $p(\mathbf{x}_i | \mathbf{z}_i)$ given in Equation (2.53) into Equation (2.60). This gives us the full expression for the joint log likelihood term in ELBO as below:

$$\begin{aligned} \sum_{i=1}^N \mathbb{E}_{q_\phi(\mathbf{z}_i)} [\log p_\theta(\mathbf{x}_i | \mathbf{z}_i)] &= -\frac{ND_x}{2} \log(2\pi) - ND_x \log \sigma - \sum_{i=1}^N \frac{1}{2\sigma^2} (\mathbf{x}_i - \boldsymbol{\mu})^T (\mathbf{x}_i - \boldsymbol{\mu}) \\ &\quad + \sum_{i=1}^N \frac{1}{\sigma^2} \mathbb{E}_{q_\phi(\mathbf{z}_i)} [(\mathbf{x}_i - \boldsymbol{\mu})^T \mathbf{W} \mathbf{z}_i] - \sum_{i=1}^N \frac{1}{2\sigma^2} \mathbb{E}_{q_\phi(\mathbf{z}_i)} [\mathbf{z}_i^T \mathbf{W}^T \mathbf{W} \mathbf{z}_i]. \end{aligned} \quad (2.62)$$

To evaluate the above equation, we need to compute the expectation with respect to $q_\phi(\mathbf{z}_i)$ to a linear and a quadratic quantity of $\mathbf{z}_i$. According to Entity (A.5) and

(A.6) from Appendix A.2, we have:

$$\mathbb{E}_{q_\phi(\mathbf{z}_i)}[(\mathbf{x}_i - \boldsymbol{\mu})^T \mathbf{W} \mathbf{z}_i] = (\mathbf{x}_i - \boldsymbol{\mu})^T \mathbf{W} \mathbf{m}_i \quad (2.63)$$

$$\mathbb{E}_{q_\phi(\mathbf{z}_i)}[\mathbf{z}_i^T \mathbf{W}^T \mathbf{W} \mathbf{z}_i] = \text{Tr}(s_i^2 \mathbf{W}^T \mathbf{W}) + \mathbf{m}_i^T \mathbf{W}^T \mathbf{W} \mathbf{m}_i. \quad (2.64)$$

Similarly, we can derive the expression for the cross-entropy term defined in Equation (2.61) as:

$$\sum_{i=1}^N \mathbb{E}_{q_\phi(\mathbf{z}_i)}[\log p_\theta(\mathbf{z}_i)] = -\frac{ND_z}{2} \log(2\pi) - \sum_{i=1}^N \frac{1}{2} \mathbb{E}_{q_\phi(\mathbf{z}_i)}[\mathbf{z}_i^T \mathbf{z}_i], \quad (2.65)$$

$$\text{where } \mathbb{E}_{q_\phi(\mathbf{z}_i)}[\mathbf{z}_i^T \mathbf{z}_i] = \text{Tr}(s_i^2 \mathbf{I}) + \mathbf{m}_i^T \mathbf{m}_i = D_z s_i^2 + \mathbf{m}_i^T \mathbf{m}_i. \quad (2.66)$$

To derive the above expression, we have used Entity (A.7) from Appendix A.2.

Finally, we turn our attention to the entropy term $\mathbb{E}_{q(\mathbf{z})}[\log q(\mathbf{z})]$ in the ELBO and derive an expression for it. We first apply the mean-field assumption, which allows us to simplify the entropy term as:

$$\mathbb{E}_{q(\mathbf{z})}[\log q(\mathbf{z})] = \sum_{i=1}^N \mathbb{E}_{q(\mathbf{z}_i)}[\log q(\mathbf{z}_i)]. \quad (2.67)$$

We then substitute the definition of $q(\mathbf{z}_i)$ given in Equation (2.55). We have:

$$\sum_{i=1}^N \mathbb{E}_{q(\mathbf{z}_i)}[\log q(\mathbf{z}_i)] = -\frac{ND_z}{2} \log(2\pi) - D_z \sum_{i=1}^N \log s_i - \sum_{i=1}^N \frac{1}{2s_i^2} \mathbf{m}_i^T \mathbf{m}_i \quad (2.68)$$

$$- \sum_{i=1}^N \frac{1}{2s_i^2} \mathbb{E}_{q(\mathbf{z}_i)}[\mathbf{z}_i^T \mathbf{z}_i] + \sum_{i=1}^N \frac{1}{s_i^2} \mathbb{E}_{q(\mathbf{z}_i)}[\mathbf{m}_i^T \mathbf{z}_i]. \quad (2.69)$$

$\mathbb{E}_{q(\mathbf{z}_i)}[\mathbf{z}_i^T \mathbf{z}_i]$ has been derived in Equation (2.66). Applying the entity (A.5), we know $\mathbb{E}_{q(\mathbf{z}_i)}[\mathbf{m}_i^T \mathbf{z}_i] = \mathbf{m}_i^T \mathbf{m}_i$. Substituting these in the above equation, we have:

$$\sum_{i=1}^N \mathbb{E}_{q(\mathbf{z}_i)}[\log q(\mathbf{z}_i)] = -\frac{ND_z}{2} \log(2\pi) - D_z \sum_{i=1}^N \log s_i - \frac{ND_z}{2}. \quad (2.70)$$

Putting together the expressions derived in Equations (2.62, 2.65 and 2.70), we have obtained an expression for the ELBO given the current model parameters

$\{\mathbf{W}, \boldsymbol{\mu}, \sigma, \mathbf{m}_i, s_i\}$, as below:

$$\begin{aligned} \text{ELBO} = & -\frac{ND_x}{2} \log(2\pi) - ND_x \log \sigma - \sum_{i=1}^N \frac{1}{2\sigma^2} (\mathbf{x}_i - \boldsymbol{\mu})^T (\mathbf{x}_i - \boldsymbol{\mu}) \\ & + \sum_{i=1}^N \frac{1}{\sigma^2} (\mathbf{x}_i - \boldsymbol{\mu})^T \mathbf{W} \mathbf{m}_i - \sum_{i=1}^N \frac{1}{2\sigma^2} \text{Tr}(s_i^2 \mathbf{W}^T \mathbf{W}) - \sum_{i=1}^N \frac{1}{2\sigma^2} \mathbf{m}_i^T \mathbf{W}^T \mathbf{W} \mathbf{m}_i \\ & - \sum_{i=1}^N \frac{D_z}{2} s_i^2 - \sum_{i=1}^N \frac{1}{2} \mathbf{m}_i^T \mathbf{m}_i + \sum_{i=1}^N D_z \log s_i + \frac{ND_z}{2}. \end{aligned} \quad (2.71)$$

2.5.3 Exact update equations for the model parameters

With the expression of ELBO derived in the previous section, we are ready to derive the update rule for each parameter. To achieve this, we compute the derivative of the ELBO given in Equation (2.71) with respect to each parameter and set this derivative to zero, as the procedure specified in Section 2.4.2.

Firstly, we compute the derivative for the posterior mean $\mathbf{m}_i$, i.e.

$$\frac{\partial \text{ELBO}}{\partial \mathbf{m}_i} = \frac{1}{\sigma^2} (\mathbf{x}_i - \boldsymbol{\mu})^T \mathbf{W} - \frac{1}{\sigma^2} \mathbf{m}_i^T \mathbf{W}^T \mathbf{W} - \mathbf{m}_i^T = 0 \quad (2.72)$$

Re-arranging the above equation, we have:

$$\mathbf{m}_i^T = (\mathbf{x}_i - \boldsymbol{\mu})^T \mathbf{W} (\mathbf{W}^T \mathbf{W} + \sigma^2 \mathbf{I})^{-1}. \quad (2.73)$$

Secondly, we compute the derivative for the posterior standard deviation s_i , i.e.

$$\frac{\partial \text{ELBO}}{\partial s_i} = \frac{D_z}{s_i} - \frac{1}{\sigma^2} \text{Tr}(\mathbf{W}^T \mathbf{W}) s_i - D_z s_i = 0 \quad (2.74)$$

Re-arranging the above equation, we have:

$$s_i = \sqrt{\frac{D_z}{\frac{1}{\sigma^2} \text{Tr}(\mathbf{W}^T \mathbf{W}) + D_z}}. \quad (2.75)$$

Thirdly, we compute the derivative for the mean vector $\boldsymbol{\mu}$ in the conditional distribution $p(\mathbf{x}|\mathbf{z})$, i.e.

$$\frac{\partial \text{ELBO}}{\partial \boldsymbol{\mu}} = \sum_{i=1}^N \left(\frac{1}{\sigma^2} \mathbf{x}_i^T - \frac{1}{\sigma^2} \boldsymbol{\mu}^T - \frac{1}{\sigma^2} (\mathbf{W} \mathbf{m}_i)^T \right) = 0 \quad (2.76)$$

Re-arranging the above equation, we have:

$$\boldsymbol{\mu}^T = \frac{1}{N} \sum_{i=0}^N (\mathbf{x}_i^T - \mathbf{m}_i^T \mathbf{W}^T). \quad (2.77)$$

Fourthly, we compute the derivative for the mapping matrix $\mathbf{W}$ in the conditional distribution $p(\mathbf{x}|\mathbf{z})$, i.e.

$$\frac{\partial \text{ELBO}}{\partial \mathbf{W}} = \sum_{i=0}^N \left(\frac{1}{\sigma^2} (\mathbf{x}_i - \boldsymbol{\mu}) \mathbf{m}_i^T - \frac{1}{\sigma^2} \mathbf{W} \mathbf{m}_i \mathbf{m}_i^T - \frac{s_i^2}{\sigma^2} \mathbf{W} \right) = 0 \quad (2.78)$$

Re-arranging the above equation, we have:

$$\mathbf{W} = \left[\sum_{i=0}^N (\mathbf{m}_i \mathbf{m}_i^T + s_i^2 \mathbf{I}) \right]^{-1} \left(\sum_{i=0}^N (\mathbf{x}_i - \boldsymbol{\mu}) \mathbf{m}_i^T \right). \quad (2.79)$$

Finally, we compute the derivative for noise standard deviation σ in the conditional distribution $p(\mathbf{x}|\mathbf{z})$, i.e.

$$\frac{\partial \text{ELBO}}{\partial \sigma} = \sum_{i=0}^N \left(\frac{D_x}{\sigma} + \sigma^{-3} (\mathbf{x}_i - \boldsymbol{\mu})^T (\mathbf{x}_i - \boldsymbol{\mu}) - 2\sigma^{-3} (\mathbf{x}_i - \boldsymbol{\mu})^T \mathbf{W} \mathbf{m}_i \right) \quad (2.80)$$

$$+ \sigma^{-3} \mathbf{m}_i^T \mathbf{W}^T \mathbf{W} \mathbf{m}_i + \sigma^{-3} \text{Tr}(\mathbf{W}^T \mathbf{W}) s_i^2 = 0 \quad (2.81)$$

Re-arranging the above equation, we have:

$$\sigma = \sqrt{\frac{1}{ND_x} \sum_{i=0}^N \left((\mathbf{x}_i - \boldsymbol{\mu})^T [\mathbf{x}_i - \boldsymbol{\mu} - 2\mathbf{W} \mathbf{m}_i] + \mathbf{m}_i^T \mathbf{W}^T \mathbf{W} \mathbf{m}_i + \text{Tr}(\mathbf{W}^T \mathbf{W}) s_i^2 \right)}. \quad (2.82)$$

2.5.4 Update procedures

As we have derived the update equations for the model parameters in our probabilistic PCA model, we can estimate the model parameters through an iterative procedure, as detailed in Algorithm 1 in the next page.

Notice that we have separated the parameters into two groups: 1) local parameters refer to the posterior means and standard deviations that are associated with individual data samples. The update of these parameters only require local information of a single data sample. 2) global parameters refer to the parameters that govern the linear mapping from the latent variable to the data variable. To update these parameters, we need to iterate through the entire dataset to compute

the corresponding summation operation in Line (11-13). Note that the information about the dataset can be obtained from its sufficient statistics, such as the mean and variance, which only need to be computed once. However, we still need to compute quantities that require the posterior means and standard deviations of all data samples. These quantities change at every iteration. Therefore, the computation of these global parameters can be expensive. We can speed up the update of these global parameters by using a random subset of the dataset at every iteration. This leads to the stochastic variational inference algorithm as introduced in [49].

Algorithm 1: Iterative optimization for a probabilistic PCA

Data: A dataset of N observations $\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$
Result: Optimized model parameters $\{\mathbf{W}, \boldsymbol{\mu}, \sigma, \mathbf{m}_i, s_i\}$, where $i = 1, \dots, N$

```

1 Init
2   Global parameters:  $\mathbf{W} = \mathbf{I}$ ,  $\boldsymbol{\mu} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ,  $\sigma \sim \text{Unit}(0, 1)$ ;
3   Local parameters:  $\mathbf{m}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ,  $s_i \sim \text{Unit}(0, 1)$  for all  $i = 1, \dots, N$ ;
4   Count:  $t = 0$ ;
5   ELBO: Compute  $\text{ELBO}_0$  according to Equation (2.71),  $\Delta \text{ELBO} = 1$ ;
6   Termination condition:  $\epsilon = 0.01$ 
7 while  $\Delta \text{ELBO} > \epsilon$  do
8   /* Update local parameters */
9   for  $i = 1, \dots, N$  do
10     $\mathbf{m}_i^T = (\mathbf{x}_i - \boldsymbol{\mu})^T \mathbf{W} (\mathbf{W}^T \mathbf{W} + \sigma^2 \mathbf{I})^{-1}$ ;
11     $s_i = \sqrt{\frac{D_z}{\frac{1}{\sigma^2} \text{Tr}(\mathbf{W}^T \mathbf{W}) + D_z}}$ 
12   /* Update global parameters */
13    $\boldsymbol{\mu}^T = \frac{1}{N} \sum_{i=0}^N (\mathbf{x}_i^T - \mathbf{m}_i^T \mathbf{W}^T)$ ;
14    $\mathbf{W} = \left[ \sum_{i=0}^N (\mathbf{m}_i \mathbf{m}_i^T + s_i^2 \mathbf{I}) \right]^{-1} \left( \sum_{i=0}^N (\mathbf{x}_i - \boldsymbol{\mu}) \mathbf{m}_i^T \right)$ ;
15    $\sigma = \sqrt{\frac{1}{ND_x} \sum_{i=0}^N \left( (\mathbf{x}_i - \boldsymbol{\mu})^T [\mathbf{x}_i - \boldsymbol{\mu} - 2\mathbf{W}\mathbf{m}_i] + \mathbf{m}_i^T \mathbf{W}^T \mathbf{W} \mathbf{m}_i + \text{Tr}(\mathbf{W}^T \mathbf{W}) s_i^2 \right)}$ ;
16   /* Test whether ELBO has converged */
17    $t = t + 1$ ;
18   Compute  $\text{ELBO}_t$  according to Equation (2.71);
19    $\Delta \text{ELBO} = \text{ELBO}_t - \text{ELBO}_{t-1}$ 

```

2.5.5 Estimating parameters in an example

We now show how well the derived iterative update procedure works in a simple example. Here we generate a dataset of 200 samples from a zero-mean unit-variance Gaussian distribution in a 1D latent space, shown as the black bars in the left plot

in Figure 2.5. We then transform the latent samples into a 2D data space via the following linear mapping:

$$\mathbf{x} = \mathbf{W}_{\text{gt}}\mathbf{z} + \boldsymbol{\mu}_{\text{gt}} + \boldsymbol{\epsilon}, \quad (2.83)$$

$$\text{where } \mathbf{W}_{\text{gt}} = \begin{bmatrix} 1.5 \\ 2 \end{bmatrix}, \boldsymbol{\mu}_{\text{gt}} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma_{\text{gt}}^2 \mathbf{I}), \text{ where } \sigma_{\text{gt}} = 0.2. \quad (2.84)$$

The transformed data samples are shown as blue dots in Figure 2.5(b).

We apply Algorithm 1 to these data points and the goal is to obtain an estimate of the mapping parameters $\{\mathbf{W}_{\text{gt}}, \boldsymbol{\mu}_{\text{gt}}, \sigma_{\text{gt}}\}$, as well as the local posterior parameters $\{\mathbf{m}_i, s_i\}$ for each data sample. As we update the parameters according to Equation (2.73-2.82), we monitor the change of ELBO in Figure 2.6 at every parameter update (right plot) and at a full pass of all parameter updates as we go through the entire dataset (left plot). We can see that the ELBO has converged by around 10 full passes of the entire dataset. By the end of 30 epochs, we have obtained the estimated mapping parameters as the following:

$$\hat{\mathbf{W}} = \begin{bmatrix} 1.35 \\ 1.80 \end{bmatrix}, \hat{\boldsymbol{\mu}} = \begin{bmatrix} 0.91 \\ 1.91 \end{bmatrix}, \hat{\sigma} = 0.21. \quad (2.85)$$

To see how well the estimated parameters fit the observed dataset, we plot the data points that are projected from the estimated latent samples $\mathbf{z}_i \sim \mathcal{N}(\mathbf{m}_i, s_i^2 \mathbf{I})$

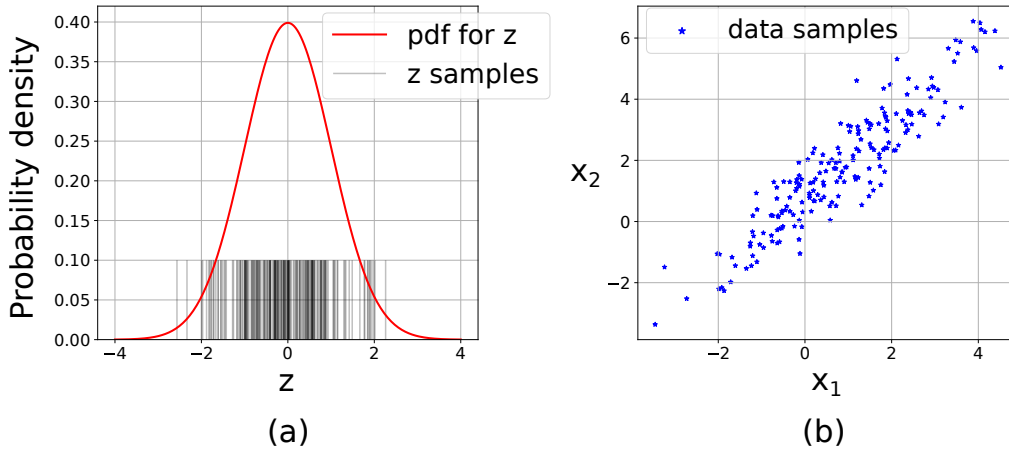


Figure 2.5: Data generated for the probabilistic PCA example. In this example, we have 1D latent variable $\mathbf{z}$ and 2D data variable $\mathbf{x}$. (a) We first generate 200 latent samples according to a zero-mean, unit-variance Gaussian distribution, i.e. $\mathbf{z}_i \sim \mathcal{N}(0, 1)$ for $i = 1, \dots, 200$. (b) Data samples are generated through a linear mapping and an additional white noise, according to Equation (2.83) and (2.84). The blue dots are the data samples allowed to use to estimate the model parameters.

under the estimated linear mapping with the random noise corruption from ϵ from above. The comparison of the estimated data samples and the actual observed data samples are shown in the left plot in Figure (2.7). We can see that the blue and red clusters exhibit very similar distributions. In addition, we also plot 1000 generated data samples from the ground truth mapping parameters given in Equation (2.84) and those from the estimated parameters given in Equation (2.85) in the right plot in Figure (2.7). Again, we can see that the two clusters exhibit very similar distributions.

Of course, the estimation of the parameters is not perfect, as we can see a noticeable difference between the estimated parameters and the ground truth parameters. This is because the current update procedure can be stuck at a local

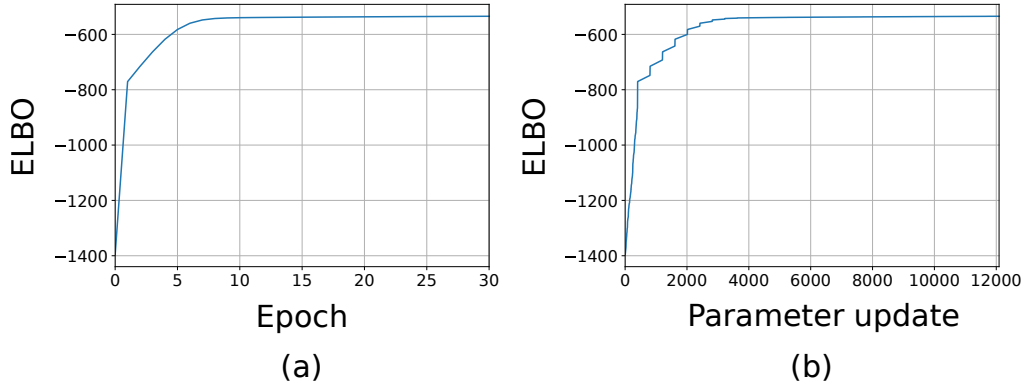


Figure 2.6: The change of ELBO over a round of optimization. One epoch is equivalent to 4000 updates over individual parameters.

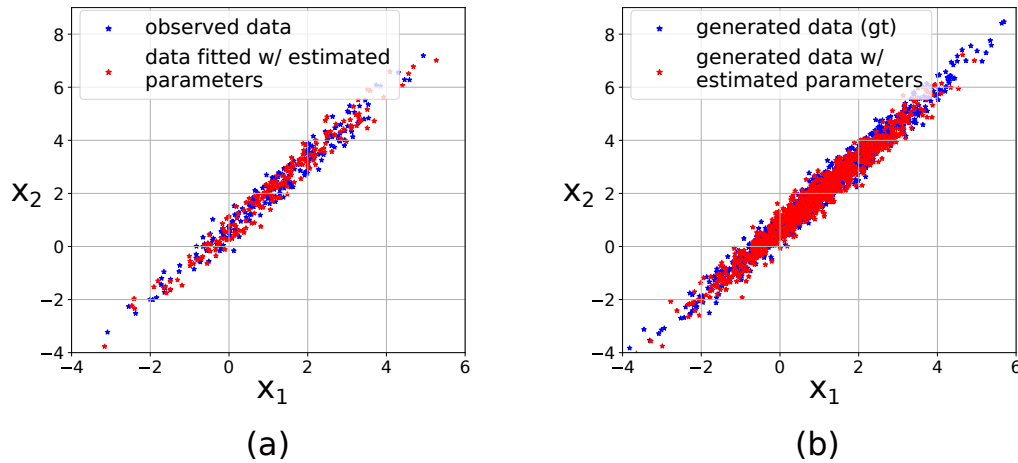


Figure 2.7: Comparison of the estimated probabilistic PCA with the ground truth model. (a) Comparison of the observed data samples (blue) with the data samples transformed by the estimated linear mapping and the latent variable values (red). (b) Comparison of 1000 generated samples from the ground truth PCA model and the PCA model with estimated parameter values.

maximum of the ELBO function. A solution to this is to run multiple rounds of the optimization, each with a different initialization. The estimated parameters will be chosen as the set that gives the highest ELBO value. Following this idea, we have run the optimization 10 times with random initialization. The ELBO convergence of these runs is shown in Figure 2.8. We can clearly see that some runs converge to a higher ELBO value than others. Examining the corresponding estimated mapping parameters, we notice the run with the highest ELBO has the following parameters:

$$\hat{\mathbf{W}}_{\text{high}} = \begin{bmatrix} 1.47 \\ 1.94 \end{bmatrix}, \hat{\boldsymbol{\mu}}_{\text{high}} = \begin{bmatrix} 1.00 \\ 1.98 \end{bmatrix}, \hat{\sigma}_{\text{high}} = 0.20. \quad (2.86)$$

This set of estimated parameters is much closer to the ground truth parameters, compared with the ones given in Equation (2.85).

2.5.6 Optimization by gradient descent

In this particular example, we have defined an extremely simple model - linear transformation between the latent and data variable and both Gaussian distributions are of isotropic diagonal form. In this case, we can derive the analytical solution of the update equations for all the parameters as shown in Section (2.5.3). However, there will be situations where deriving such update equations is too difficult or not possible. If this is the case, we can estimate the unknown parameters by applying the gradient ascent on the ELBO objective as discussed in Section 2.4.2. The benefit of the gradient ascent approach is that we can utilize many existing libraries, such as the autograd in Pytorch [50] and autodiff in Tensorflow [51], to compute the gradient efficiently for any given objective function.

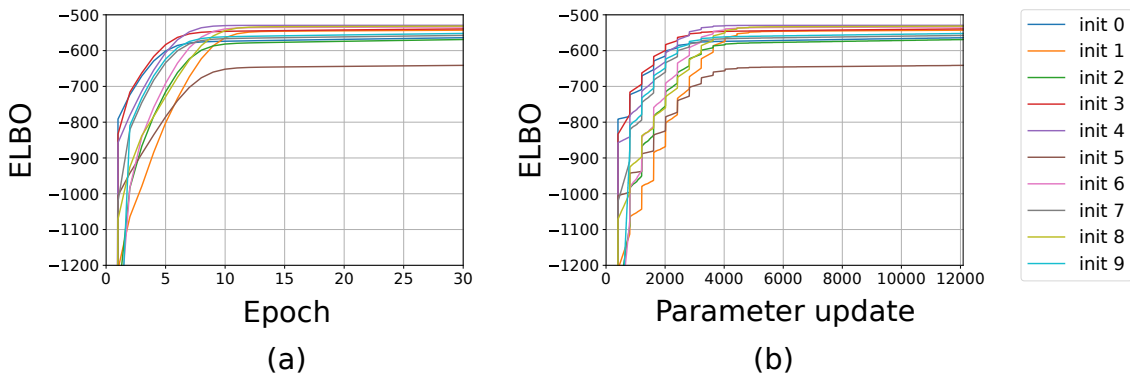


Figure 2.8: Optimizing parameters with 10 random initializations. (a) The ELBO convergence over 30 epochs. (b) The ELBO convergence plotted at every parameter update. Each coloured line corresponds to a different optimization run.

2. Background

Here we show the estimated parameters given by a gradient ascent algorithm, as below:

$$\hat{\mathbf{W}}_{\text{gd}} = \begin{bmatrix} 1.47 \\ 1.95 \end{bmatrix}, \hat{\boldsymbol{\mu}}_{\text{gd}} = \begin{bmatrix} 1.10 \\ 2.11 \end{bmatrix}, \hat{\sigma}_{\text{gd}} = 0.20. \quad (2.87)$$

We run the optimization 10 times with random initialization and the convergence of ELBO is shown in Figure 2.9. Different from the exact update procedure, all 10 runs of the gradient ascent seem to converge to the same optimum, despite the difference in the rate of convergence. We show the comparison of the fitted and the generated data points under these estimated parameters and the ground truth parameters in Figure (2.10). From this example, we can see that the gradient ascent/descent gives a sensible solution.

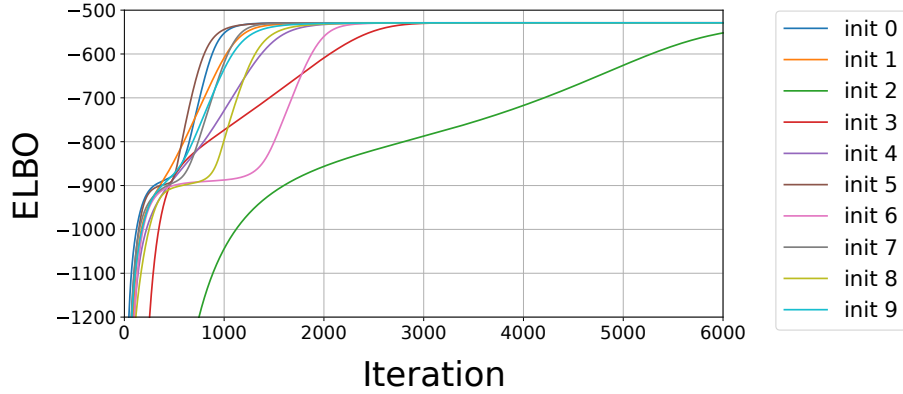


Figure 2.9: ELBO updates with the gradient ascent optimization. Here we optimize the model parameters using the gradient ascent algorithm discussed in Section 2.4.2.

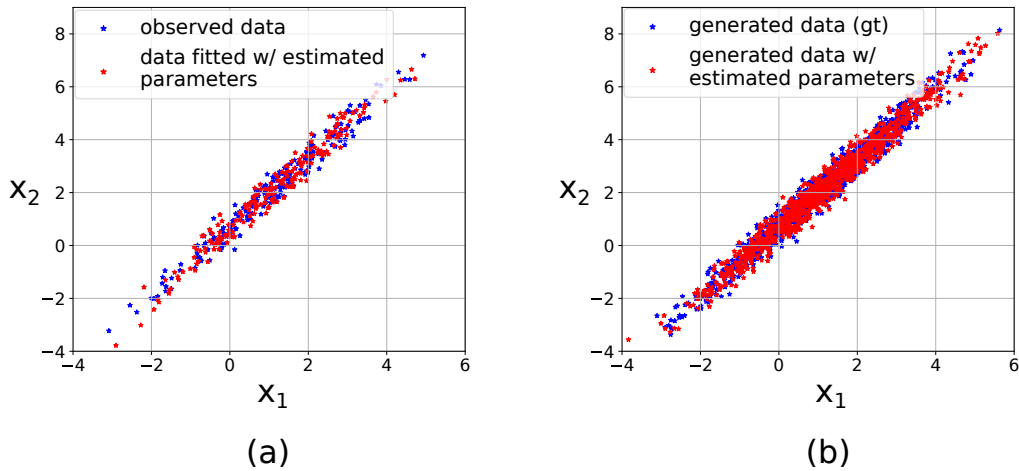


Figure 2.10: Comparison of the estimated probabilistic PCA (best from the 10 runs) with the ground truth model. The estimated model manages to capture the data distribution fairly well.

The biggest difference between the exact update procedure and the gradient ascent procedure lies in the rate of convergence. In this example, the exact update procedure reaches convergence within 30 iterations and on average this takes 0.82s on a 8-core CPU computer. In contrast, the gradient ascent procedure takes 3000 iterations of parameter updates to reach convergence 9 out of 10 cases. On average, it takes 3.49s to derive this estimation on the same computer. Therefore, we can conclude that both procedures can give reasonable estimates of the model parameters. The exact update procedure is more efficient. Hence, if we can derive the exact update solutions, we should do so. In the cases where such derivation is too complicated or not possible, then we can resort to gradient ascent and still obtain a good estimate, which will take a bit longer.

2.6 Summary

In this chapter, we have introduced the *density estimation* task. We show that the model parameters can be directly estimated if the target distribution is modelled by a simple distributions, such as a Gaussian distribution, under the *maximum likelihood objective*. For a more complex probabilistic model, such as a *latent variable model*, the model parameters need to be estimated through schemes, such as *variational inference* (VI), as the direct computation of the data likelihood might not be tractable. We derive the *evidence lower bound* objective that is key to a VI framework. We also show how the VI framework works with a pPCA model estimation example.

With a good understanding of the density estimation task and the VI framework, we will introduce *variational auto-encoder* (VAE) in the next chapter and show how the VI framework can be applied to estimate the model parameters in a VAE.

3

Variational Autoencoders

In this chapter, we will introduce variational autoencoders (VAEs) - a popular representation learning approach that combines the conventional statistical modelling - a *latent variable model* - with the flexible functional approximation capability of deep neural networks. We set up VAEs as a *non-linear extension of the probabilistic PCA* (pPCA) models. We show the *variational inference* (VI) technique introduced in the previous chapter can be applied here to estimate the model parameters in VAEs. We also introduce some common *metrics* that are often used to evaluate the representation quality derived from VAEs. We will end the chapter with a discussion on how VAEs are connected to other representation learning approaches, such as generative adversarial networks (GANs) and normalizing flows, as well as a review on existing research efforts in improving the generative modelling of VAEs.

3.1 Derivation of VAEs

VAEs [16], [17] are proposed to solve the latent variable model governed by the following joint probability:

$$p_{\theta}(\mathbf{x}, \mathbf{z}) = p_{\theta}(\mathbf{z})p_{\theta}(\mathbf{x}|\mathbf{z}), \quad (3.1)$$

as introduced in Section 2.4. In Section 2.5, we show how probabilistic principal component analysis (pPCA) models can offer a solution to the above latent variable model when the conditional probability $p(\mathbf{x}|\mathbf{z})$ is modelled by a linear mapping between the latent variable $\mathbf{z}$ and the data variable $\mathbf{x}$. VAEs consider a more flexible model, where the mapping between $\mathbf{z}$ and $\mathbf{x}$ can be non-linear. Specifically, the latent variable $\mathbf{z}$ follows a zero-mean, unit-covariance Gaussian distribution, i.e.

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z} | \mathbf{0}, \mathbf{I}), \quad (3.2)$$

and the data variable is obtained under a non-linear mapping from the latent variable $\mathbf{z}$, subject to perturbation from some white noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$, i.e.

$$\mathbf{x} = g_\theta(\mathbf{z}) + \epsilon, \quad (3.3)$$

where $g_\theta(\cdot)$ denotes the non-linear mapping which is parameterized by a set of parameters θ and ϵ denotes the random noise that is injected in the mapping. This gives us the conditional distribution $p(\mathbf{x}|\mathbf{z})$ as below:

$$p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x} | g_\theta(\mathbf{z}), \sigma^2 \mathbf{I}). \quad (3.4)$$

Similar to the pPCA example, we also consider a dataset of N observations, i.e. $\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$. We represent the corresponding graphical model for the VAE model defined in Equation (3.2) and (3.4) in Figure 3.1. Here we also include the graphical model for a pPCA model in the subfigure (b) as a comparison. Same as the pPCA task, our task here is to estimate the values of the model parameters θ and σ and to estimate the posterior distribution $p(\mathbf{z}|\mathbf{x})$ so we can infer the values of $\mathbf{z}$ given the observations of the data $\mathbf{x}$.

Recall that variational inference allows us to solve this task by approximating the posterior distribution with a probabilistic model of certain form, e.g. a Gaussian distribution that factorizes into the product of the latent variables of each individual sample, i.e. $q_\phi(\mathbf{z}) = \prod_{i=1}^N q_{\phi_i}(\mathbf{z}_i)$, for the pPCA example. For the VAE models, we adopt the same assumption (i.e. the mean-field assumption introduced in Section 2.4.3). In addition, we also assume that the posterior distribution for each sample is conditioned on the corresponding data sample and all the sample

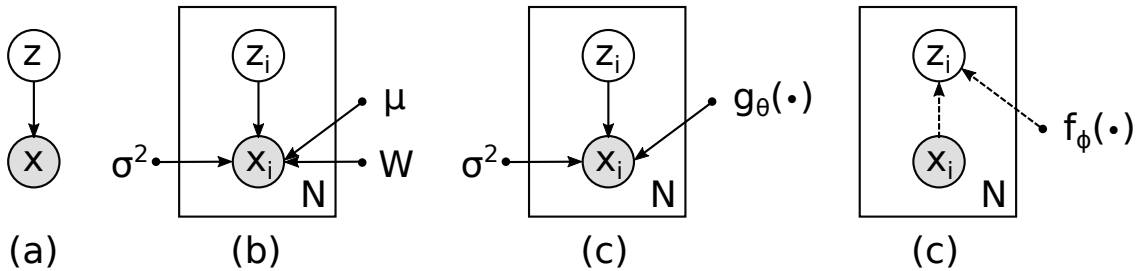


Figure 3.1: The graphical model for a VAE model. (a) The graphical model for the latent variable model expressed in Equation (3.1). (b) The graphical model for a pPCA model where N data samples are i.i.d. and $\{\mathbf{W}, \mu, \sigma\}$ are the model parameters. (c) The graphical model for a VAE model where N data samples are i.i.d. and $\{\theta, \sigma\}$ are the model parameters for this generative model. (d) Implicitly, a VAE model also includes an inference model that approximates the posterior distribution $q_\phi(\mathbf{z}|\mathbf{x}_i)$, which is governed by the model parameter ϕ .

posteriors are governed by a global function $f_\phi(\cdot)$ with parameter ϕ , i.e.

$$q_\phi(\mathbf{z}) = \prod_{i=1}^N q_\phi(\mathbf{z}_i|\mathbf{x}_i). \quad (3.5)$$

This change of converting the estimation of individual local posterior distributions to the estimation of a global function that governs all the posterior distributions is a technique often referred to as *amortization* [52]–[54]. The biggest benefit of amortization is that the inferred global function $q_\phi(\mathbf{z}_i|\mathbf{x}_i)$ is supposed to generalize to new data samples without the need of retraining the model parameters. In contrast, the variational inference framework without amortization will need to estimate all the model parameters all again, together with the parameters for the local posterior distributions of the new data samples, when a new data sample arrives. Therefore, the amortization introduced in the VAE models hugely simplifies the inference procedure.

We further assume that the posterior distribution for each sample follows a diagonal Gaussian distribution as below:

$$q_\phi(\mathbf{z}_i|\mathbf{x}_i) = \mathcal{N}(\mathbf{z}_i|\boldsymbol{\mu}_i, \mathbf{s}_i^2), \quad (3.6)$$

where the mean and the variance of the posterior distribution are the output from the non-linear function $f_\phi(\cdot)$, i.e. $[\boldsymbol{\mu}_i, \mathbf{s}_i] = f_\phi(\mathbf{x}_i)$. We demonstrate this inference process in the subfigure (d) with a dashed arrow going from a data sample $\mathbf{x}_i$ to its corresponding latent variable $\mathbf{z}_i$.

One of the key decisions that makes VAEs so powerful and versatile is the adoption of deep neural networks [20], [21] to parameterize the two non-linear functions - $g_\theta(\cdot)$ that govern the conditional distribution $p(\mathbf{x}|\mathbf{z})$ in Equation (3.4) and $f_\phi(\cdot)$ that governs the posterior distribution $q(\mathbf{z}_i|\mathbf{x}_i)$ in Equation (3.6). Neural networks' capability to approximate a wide range of complex functions allows VAEs to obtain certain mappings that can fit well with the current dataset while maximally satisfying the model assumptions, i.e. the latent variable $\mathbf{z}$ draws from zero-mean, unit-variance Gaussian and the posteriors follow diagonal Gaussian distributions. The neural network architecture of VAE models is often illustrated as in Figure 3.2. The function $f_\phi(\cdot)$ in the inference model is shown as an encoder that projects a data sample $\mathbf{x}$ to a latent code $\mathbf{z}$. The function $g_\theta(\cdot)$ in the generative model is shown as a decoder that reconstructs the latent code $\mathbf{z}$ back to the data space $\hat{\mathbf{x}}$.

3.1.1 Deriving ELBO objective

To estimate the model parameters ϕ and θ we have introduced in the probabilistic model in Equation (3.1-3.6), we resort to the variational inference framework as introduced in Section 2.4. To start with, we first derive the ELBO objective for the VAE models.

Recall that we have derived the ELBO objective as a function of the model parameters ϕ and θ in Equation (2.37) and (2.38) and we re-state the results below:

$$\text{ELBO}(\phi, \theta) = \mathbb{E}_{q_\phi(\mathbf{z})} [\log p_\theta(\mathbf{z}, \mathbf{x})] - \mathbb{E}_{q_\phi(\mathbf{z})} [\log q_\phi(\mathbf{z})] \quad (3.7)$$

$$= \log p(\mathbf{x}) - D_{\text{KL}}(q_\phi(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x})). \quad (3.8)$$

Replacing the posterior $q(\mathbf{z})$ with the amortized posterior defined in Equation (3.5), together with the i.i.d. assumption for the dataset $\mathcal{D}$, we have the ELBO for the VAE models as:

$$\text{ELBO}(\phi, \theta) = \sum_{i=1}^N \left[\mathbb{E}_{q_\phi(\mathbf{z}_i|\mathbf{x}_i)} [\log p_\theta(\mathbf{z}_i, \mathbf{x}_i)] - \mathbb{E}_{q_\phi(\mathbf{z}_i|\mathbf{x}_i)} [\log q_\phi(\mathbf{z}_i|\mathbf{x}_i)] \right] \quad (3.9)$$

$$= \sum_{i=1}^N \left[\log p(\mathbf{x}_i) - D_{\text{KL}}(q_\phi(\mathbf{z}_i|\mathbf{x}_i) \parallel p(\mathbf{z}_i|\mathbf{x}_i)) \right]. \quad (3.10)$$

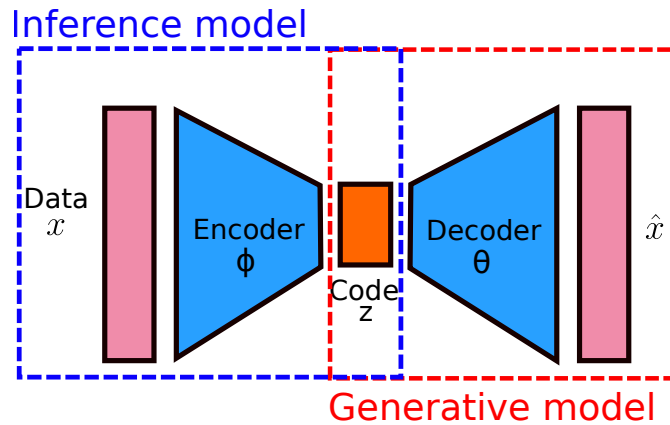


Figure 3.2: Typical VAE neural network architecture. The decoder model shown in the red dashed box represents the generative model defined in Equation (3.4), where a latent code $\mathbf{z}$ is mapped through a nonlinear function $g_\theta(\cdot)$ to a data sample. VAEs also introduce an approximate inference model that encodes a data sample $\mathbf{x}$ to a latent code $\mathbf{z}$ through an encoder network.

Now we denote $\text{ELBO}_i(\phi, \theta)$ as the ELBO for the data sample $\mathbf{x}_i$, i.e.

$$\text{ELBO}_i(\phi, \theta) = \mathbb{E}_{q_\phi(\mathbf{z}_i|\mathbf{x}_i)} [\log p_\theta(\mathbf{z}_i, \mathbf{x}_i)] - \mathbb{E}_{q_\phi(\mathbf{z}_i|\mathbf{x}_i)} [\log q_\phi(\mathbf{z}_i|\mathbf{x}_i)] \quad (3.11)$$

$$= \log p(\mathbf{x}_i) - D_{\text{KL}}(q_\phi(\mathbf{z}_i|\mathbf{x}_i) \parallel p(\mathbf{z}_i|\mathbf{x}_i)). \quad (3.12)$$

Apparently, $\text{ELBO}(\phi, \theta) = \sum_{i=1}^N \text{ELBO}_i(\phi, \theta)$.

In the previous chapter, we have derived an analytical form of the ELBO objective for the pPCA models using the expression in Equation (3.9). In the context of VAEs, we often re-organize the terms and adopt a slightly different form, as shown below:

$$\text{ELBO}_i(\phi, \theta) = \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}_i|\mathbf{x}_i)} [\log p_\theta(\mathbf{x}_i|\mathbf{z}_i)]}_{\text{① reconstruction likelihood}} - \underbrace{D_{\text{KL}}(q_\phi(\mathbf{z}_i|\mathbf{x}_i) \parallel p(\mathbf{z}))}_{\text{② prior constraint}}. \quad (3.13)$$

The above $\text{ELBO}_i(\phi, \theta)$ contains two terms: ① a reconstruction likelihood term that encourages good reconstruction through a VAE model's encoding and decoding process, i.e. an auto-encoding process; ② a prior constraint term that regularizes the posterior distribution $q_\phi(\mathbf{z}_i|\mathbf{x}_i)$ to remain as close as possible to the prior distribution for the latent variable $p(\mathbf{z})$, which is defined as a zero-mean, unit variance Gaussian in Equation (3.2).

We can derive an analytical solution to evaluate both terms in $\text{ELBO}_i(\phi, \theta)$ for a given set of model parameters, i.e. θ and ϕ . To achieve this, we first approximate the expectation in ① with a sampling approach [12], [55]. To do this, we draw L samples from the posterior distribution, i.e.

$$\mathbf{z}_i^{(l)} \sim q_\phi(\mathbf{z}_i|\mathbf{x}_i), \quad l = 1, \dots, L. \quad (3.14)$$

We then approximate the expectation as an average of the log likelihood (LL) evaluated at the sampled latent features, i.e.

$$\mathbb{E}_{q_\phi(\mathbf{z}_i|\mathbf{x}_i)} [\log p_\theta(\mathbf{x}_i|\mathbf{z}_i)] \approx \frac{1}{L} \sum_{l=1}^L \log p_\theta(\mathbf{x}_i|\mathbf{z}_i^{(l)}). \quad (3.15)$$

The log likelihood $\log p_\theta(\mathbf{x}_i|\mathbf{z}_i^{(l)})$ can be computed depending on the specific model definition. For example, we define the conditional distribution $p_\theta(\mathbf{x}|\mathbf{z})$ as a Gaussian distribution in Equation (3.4). This often works well when the data variable can be modelled as a continuous variable. In some other scenarios where the data variable is more discrete in nature. Then it is more suited to model the conditional distribution

$p_\theta(\mathbf{x}|\mathbf{z})$ with a Bernoulli or a multinoulli distribution (a generalization of a Bernoulli distribution to a categorical variable over K classes) as introduced in Section 2.1.1. Either way, the log likelihood can be evaluated exactly. Hence, the expectation term can be approximately evaluated.

The ② term has a closed-form solution as the benefit of having a simple Gaussian prior $p(\mathbf{z})$ and a Gaussian posterior $q_\phi(\mathbf{z}_i|\mathbf{x}_i)$. Using the result of the KL-divergence between two multivariate Gaussians from Appendix A.4, we have the KL term in the $\text{ELBO}_i(\phi, \theta)$ as:

$$D_{\text{KL}}(q_\phi(\mathbf{z}_i|\mathbf{x}_i) \| p(\mathbf{z})) = \frac{1}{2} \sum_{d=1}^D \left[(\boldsymbol{\mu}_i^{(d)})^2 + (\mathbf{s}_i^{(d)})^2 - \log (\mathbf{s}_i^{(d)})^2 - 1 \right], \quad (3.16)$$

where $\boldsymbol{\mu}$ and $\mathbf{s}^2$ are the mean and variance of the posterior distribution $q_\phi(\mathbf{z}_i|\mathbf{x}_i)$ (defined in Equation (3.6)) and the super-script $(\cdot)^{(d)}$ refers to the d -th dimension of the latent variable.

With the results from Equation (3.15) and (3.16), we have obtained the full analytical expression of the $\text{ELBO}_i(\phi, \theta)$. This analytical learning objective, together with the fact that the solution for the ② term is exact (i.e. not an approximation), contributes to the behaviour of the VAE models that are often easy to train and converge.

3.1.2 Learning model parameters

With the ELBO objective derived in the previous section, we are now ready to estimate the set of model parameters θ and ϕ that maximizes the ELBO objective. As θ and ϕ are parameters of two neural networks, deriving the exact solution for these parameters in the same way as we did in Section 2.5 for a pPCA is not possible. However, we can still resort to the iterative optimization approach introduced in Section 2.4.2. Recall that to obtain an update on the current values of the model parameters θ and ϕ , we need to compute the gradient of $\text{ELBO}_i(\phi, \theta)$ with respect to θ and ϕ , i.e.

$$\Delta\theta^{(s)} = \left. \frac{\partial \text{ELBO}_i}{\partial \theta^{(s)}} \right|_{\theta, \phi} \quad \text{for } s = 1, \dots \quad (3.17)$$

$$\Delta\phi^{(t)} = \left. \frac{\partial \text{ELBO}_i}{\partial \phi^{(t)}} \right|_{\theta, \phi} \quad \text{for } t = 1, \dots \quad (3.18)$$

When dealing with a complex function, such as a neural network, we can resort to a technique often referred to as *back-propagation* [56], [57], which uses the chain rule to break down the gradient computation of a complex function to a chain of

gradients of simple functions. However, there is a problem when applying the back-propagation to the VAE models - the sampling operation needed to approximate the reconstruction likelihood term in ① is not differentiable.

To resolve this issue, the authors of VAEs propose to adopt a *reparameterization* trick [16], [17] to re-position the sampling from right in the middle of the auto-encoder network to the input of the encoder. The trick works as follows: let's introduce a dummy random variable - $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. When the encoder network predicts the latent variable for a given data sample $\mathbf{x}_i$ as $\mathbf{z}_i \sim \mathcal{N}(\boldsymbol{\mu}_i, \mathbf{s}_i^2)$, we can obtain a sample of $\mathbf{z}_i$ by first sampling ϵ from the zero-mean, unit-variance Gaussian and transforming the sampled ϵ through the following mapping:

$$\mathbf{z}_i = \boldsymbol{\mu}_i + \mathbf{s}_i \cdot \epsilon. \quad (3.19)$$

The power of this trick is that it allows us to re-position the sampling of the latent variable at the output from the encoder network to the input of the encoder. Hence, the sampling does not block the gradient computation any more. Now, we can use back-propagation to compute the gradient of $\text{ELBO}_i(\phi, \theta)$ with respect to θ and ϕ .

There is a second issue in the gradient computation. The ELBO objective that we have focused on is the ELBO expression for a single data sample $\mathbf{x}_i$ as defined as $\text{ELBO}_i(\phi, \theta)$ in Equation (3.13). The true objective that we need to maximize in the variational inference framework is the ELBO for the entire dataset; this means that we need to sum $\text{ELBO}_i(\phi, \theta)$ over all N data samples in $\mathcal{D}$. This is indeed what we have done when solving for the model parameters for the pPCA model in Section 2.5.4. However, such computation at every iteration of the parameter update would make the optimization a very slow process, as the dataset often consists of millions of samples, the VAE model is often parameterized by neural networks that contain millions of parameters, and a common gradient descent optimization often takes more than 10k update steps. This would make the optimization not practical.

To handle this issue, VAEs adopt an estimator of the dataset ELBO objective through stochastic sampling [58], [59]. As the data samples in $\mathcal{D}$ are i.i.d., if we randomly draw a batch of M samples from the dataset denoted as $\mathcal{D}_M$ with $M \ll N$, then the expectation of any function computed with the subset $\mathcal{D}_M$ will be an unbiased estimator of the function evaluated with the entire dataset $\mathcal{D}$. In our case, we will have an unbiased estimator of the ELBO using small batches of the entire dataset, i.e.

$$\text{ELBO}(\theta, \phi) \approx \frac{N}{M} \sum_{i \in \mathcal{D}_M} \text{ELBO}_i(\theta, \phi). \quad (3.20)$$

Apparently, this will reduce the computation time for every update step and, hence, speed up the optimization by an order of $\frac{N}{M}$ compared to evaluating the exact ELBO over the entire dataset.

With all the components (amortization, the reparameterization trick and the stochastic sampling) introduced above, we have introduced the learning algorithm of a VAE model. Fundamentally, it solves a latent variable model that contains a non-linear mapping between the latent and the data variable. To solve the model parameters, we resort to the variational inference framework. Because of some good choice of probability models and some technique such as Monte Carlo sampling approaches, we can compute the ELBO objective analytically. We then use an iterative update algorithm to optimize the model parameters in the VAE model. To make sure the update can be computed efficiently, we use a stochastic estimator of the true ELBO for the entire dataset. In the following sections, we will discuss how a trained VAE model is evaluated under different metrics.

3.2 Evaluation metrics for VAEs

In this section, we will introduce the common metrics that are used to evaluate a VAE model's performance. As the ground truth solution to a VAE model is not available - the underlying latent variable model that has generated a given dataset is often not known and the latent features for individual data samples are also not observed, these evaluation metrics are all we have to decide whether a sensible model has been inferred and to compare performance across different possible VAE models.

3.2.1 Likelihood metrics

As VAEs essentially solve a density estimation task for a latent variable model, i.e. estimating the probabilistic model for the data distribution $p(\mathbf{x})$, the most natural metric for performance is to evaluate the log likelihood of the inferred model on a held-out test set of data samples [16]. In the VAE setting, the inferred model is:

$$p_{\phi, \theta}(\mathbf{x}, \mathbf{z}) = p_{\theta}(\mathbf{x}|\mathbf{z})p(\mathbf{z}) = q_{\phi}(\mathbf{z}|\mathbf{x})p(\mathbf{x}). \quad (3.21)$$

From the above equation, we can derive the log likelihood of the data distribution for a test sample $\mathbf{x}^*$ as (the full derivation is given in Appendix A.5):

$$\log p(\mathbf{x}^*) = \log \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x}^*)} \left[\frac{p_{\theta}(\mathbf{x}^*, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x}^*)} \right] = \log \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x}^*)} \left[\frac{p_{\theta}(\mathbf{x}^*|\mathbf{z})p(\mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x}^*)} \right], \quad (3.22)$$

where the expectation with respect to the posterior distribution $q_\phi(\mathbf{z}|\mathbf{x}^*)$ can be approximated through Monte Carlo sampling. To do so, we draw L samples from the posterior distribution, i.e.

$$\mathbf{z}^{(l)} \sim q_\phi(\mathbf{z}|\mathbf{x}^*), \quad l = 1, \dots, L. \quad (3.23)$$

We then approximate the expectation as an average of the log likelihood evaluated at the sampled latent features, i.e.

$$\log p(\mathbf{x}^*) \approx \frac{1}{L} \sum_{l=1}^L \log \frac{p_\theta(\mathbf{x}^*|\mathbf{z})p(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x}^*)}. \quad (3.24)$$

As a convention, we often report the negative of the above log likelihood, so the higher the value the better the model performs. We refer this metric as the *negative log likelihood* metric.

Another metric that is highly related to the negative log likelihood metric is a lower bound of the log likelihood defined in Equation (3.22). The lower bound can be obtained from applying Jensen's Inequality [12] to Equation (3.22), giving:

$$\mathcal{L}(\mathbf{x}^*) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^*)} \left[\log \frac{p_\theta(\mathbf{x}^*, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x}^*)} \right] \leq \log \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^*)} \left[\frac{p_\theta(\mathbf{x}^*, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x}^*)} \right] = \log p(\mathbf{x}^*). \quad (3.25)$$

Closely inspecting the above lower bound $\mathcal{L}(\mathbf{x}^*)$, we notice that $\mathcal{L}(\mathbf{x}^*)$ is effectively the same as the ELBO objective defined in Equation (3.11). Indeed, the average ELBO on the test set is commonly used to evaluate the performance of the inferred VAE model. Specifically, when a large number of samples are drawn from the posterior distribution $q_\phi(\mathbf{z}|\mathbf{x}^*)$, i.e. L is large, such ELBO is referred to as *importance weighted ELBO* [60]. It is believed that such ELBO gives a tighter bound to the log likelihood $\log p(\mathbf{x}^*)$ than the ELBO evaluated with a few samples from the posterior, i.e. L is small or even just 1.

3.2.2 Reconstruction error

Another commonly used metric is the *reconstruction error* of the inferred VAE model on a held-out test set. The reconstruction error is a sensible choice, as the VAE model effectively derives an auto-encoding scheme as shown in Figure 3.2. The amount of the reconstruction error indicates the ability of the VAE model to preserve the information in the input data samples through the VAE's auto-encoding scheme. In fact, the reconstruction error naturally appears in VAE's ELBO objective in the ① of Equation (3.13). If $\mathbf{x}$ is continuous, then ① becomes a weighted least square

error between the input $\mathbf{x}_i$ and its reconstruction $\hat{\mathbf{x}}_i$. If $\mathbf{x}$ is discrete, then ① becomes a cross-entropy error between $\mathbf{x}_i$ and $\hat{\mathbf{x}}_i$.

3.2.3 Cross-entropy between aggregate posterior and prior

A metric that is often used in tandem with the reconstruction error is the *cross-entropy between the aggregate posterior and the prior* distribution, i.e. $\mathbb{E}_{q_\phi(\mathbf{z})} [\log p(\mathbf{z})]$. We first define the *aggregate posterior* $q_\phi(\mathbf{z})$ as below [25], [61]:

$$q_\phi(\mathbf{z}) = \int q_\phi(\mathbf{z}|\mathbf{x})p_{\mathcal{D}}(\mathbf{x})d\mathbf{x} = \mathbb{E}_{p_{\mathcal{D}}(\mathbf{x})} [q_\phi(\mathbf{z}|\mathbf{x})] \approx \frac{1}{N} \sum_{i=1}^N q_\phi(\mathbf{z}|\mathbf{x}_i). \quad (3.26)$$

As shown in the above equation, the aggregate posterior is the average of all the sample posteriors. Therefore, the cross-entropy between $q_\phi(\mathbf{z})$ and the prior $p(\mathbf{z})$ measures the difference (or mismatch) between these two distributions. The reason why we care about this difference is that a VAE model is supposed to recover the generative model of the given dataset with the assumption that the generative model contains a latent variable that follows the prior distribution $p(\mathbf{z})$. The difference between $q_\phi(\mathbf{z})$ and $p(\mathbf{z})$ indicates how well the trained VAE model satisfies this assumption.

3.2.4 Generation quality metrics

As VAEs are designed to infer the generative model of a given dataset, it is important that we are able to evaluate a VAE model’s capability of generating new samples. We have metrics, such as *Inception score* [62] and Frechet inception distance (FID) [63], designed for this purpose.

Inception score relies on a pre-trained classifier $\mathcal{F}$ on the same dataset, where the VAE model is trained. It then examines the network’s classification predictions on a set of generated samples from the trained VAE model. If the generated samples resemble the features of the real data samples, then the individual class prediction $p(y|\mathbf{x}_i)$ ($\mathbf{x}_i$ is a data sample from a held-out test set $\mathcal{D}_T$ and y denotes the class label) should have a low entropy. In addition, the generated samples should also render a diverse set of classes. Therefore, the marginal probability $p(y) = \frac{1}{N} \sum_i p(y|\mathbf{x}_i)$ should have a high entropy. Indeed, the inception score is computed by considering these two properties, i.e. sample quality and sample diversity, as summarized below:

$$\text{IS} = \mathbb{E}_{\mathbf{x}_i \sim \mathcal{D}_T} [\mathbb{E}_{p(y|\mathbf{x}_i)} [\log p(y|\mathbf{x}_i)]] - \mathbb{E}_{p(y)} [\log p(y)]. \quad (3.27)$$

FID score evaluates the quality of a set of generated samples $\mathcal{D}_G$ by comparing

the statistics of these samples with the statistics of a set of real data samples $\mathcal{D}_R$. It also requires a pre-trained network $\mathcal{F}$, but uses it as a feature extractor rather than a classifier. Let's take the output from $\mathcal{F}$ on an input sample as the feature $\mathbf{f}$, i.e. $\mathbf{f}_i = \mathcal{F}(\mathbf{x}_i)$. We compute all the features for the generated samples and the real samples. We then assume these features follow a multivariate Gaussian distribution and we can estimate the Gaussian distribution by computing the mean and covariance for both datasets, i.e. $\boldsymbol{\mu}_G$, $\mathbf{K}_G$ for the generated samples and $\boldsymbol{\mu}_R$, $\mathbf{K}_R$ for the real samples. Then FID score is obtained by computing the Frechet distance (or Wasserstein-2 distance) as below:

$$\text{FID} = \|\boldsymbol{\mu}_G - \boldsymbol{\mu}_R\|^2 + \text{Tr}(\mathbf{K}_G + \mathbf{K}_R - 2\sqrt{\mathbf{K}_G\mathbf{K}_R}). \quad (3.28)$$

Please note that the inception score and FID score are originally proposed to evaluate the generation quality for natural images. In the original implementation, they both adopt an inception v3 [64] as their pre-trained neural network $\mathcal{F}$. In the description above, we have generalized the definition for both scores so that they can be applied to any data forms and with a different neural network model.

3.3 Connection to other representation learning approaches

As we have seen at the start of this chapter, VAEs and probabilistic PCA (pPCA) models are closely related. Both of the approaches solve a latent variable model which is formulated by Equations (3.1) and (3.2). We can resort to the variational inference framework as introduced in Section 2.4 to derive a solution to both problems. The biggest difference between the two approaches lies in the fact that pPCA models are constrained with the assumption that the conditional distribution $p(\mathbf{x}|\mathbf{z})$ is governed by a linear mapping between the latent variable $\mathbf{z}$ and the data variable $\mathbf{x}$, whereas VAE models relax this assumption and allow the conditional distribution $p(\mathbf{x}|\mathbf{z})$ to be governed by a non-linear mapping between the latent variable $\mathbf{z}$ and the data variable $\mathbf{x}$. This change in the model definition does affect the way the latent variable model is being solved. Specifically, we can derive an exact update rule for pPCA models, as shown in Section 2.5.3. However, the complexity of the non-linear mapping considered in VAE models often means the exact solution is not available and, hence, we have to go with an iterative update approach, as detailed in Section 3.1.2. This, in general, means that it will take more iterations for VAE models to converge than the corresponding pPCA models, but

the benefit is that VAE models can derive a set of latent features that are based on a more complex transformation on the raw input. Furthermore, the mappings from the data to the latent variable and vice versa derived in VAE models often obtain better reconstruction quality, indicating the latent representations derived in VAE models preserve the information from the original input data better than pPCA models.

Other than the connection to pPCA models, VAEs are often considered as an alternative to representation learning approaches, such as GANs [13] and normalizing flows [65], [66]. We devote the rest of this section to discuss the connection and differences between VAEs and these approaches.

3.3.1 Generative adversarial networks (GANs)

GANs [13] are a widely-known and widely-used representation learning approach. In essence, GANs solve the same latent variable model as VAEs do, as formulated by Equations (3.1) and (3.2). Rather than formulating the problem in the variational inference framework, GANs achieve the density estimation in an implicit way by setting up a game between two players - a generator and a discriminator [67]. The generator's goal is to map a randomly sampled latent feature vector $\mathbf{z}_i$ to a realistic data sample. The discriminator's goal is to identify the data samples that are generated by the generator. We formulate the generator as a non-linear function G_{θ_G} with parameters θ_G that maps a latent feature vector to a data sample, i.e.

$$\hat{\mathbf{x}}_i = G_{\theta_G}(\mathbf{z}_i), \quad (3.29)$$

and the discriminator as a non-linear function D_{θ_D} with parameters θ_D that estimates the probability of a data sample is drawn from the given dataset, i.e.

$$0 \leq D_{\theta_D}(\mathbf{x}) \leq 1. \quad (3.30)$$

Both functions are often implemented with neural networks. As the discriminator is effectively a binary classifier, we can formulate a loss function for it as below:

$$\mathcal{L}(\theta_G, \theta_D) = - \underbrace{\frac{1}{2} \mathbb{E}_{\mathbf{x}_i \sim p_{data}} [\log D_{\theta_D}(\mathbf{x}_i)]}_{\text{① real samples}} - \underbrace{\frac{1}{2} \mathbb{E}_{\mathbf{z}_i \sim \mathcal{N}(0,1)} [\log(1 - D_{\theta_D}(G_{\theta_G}(\mathbf{z}_i)))]}_{\text{② generated samples}}. \quad (3.31)$$

Both terms in the above equation are cross-entropy loss as defined in Equation (2.65). Clearly the loss function $\mathcal{L}(\theta_G, \theta_D)$ depends on both the discriminator and the generator function. For any given generator, there is a set of θ_D that minimizes

$\mathcal{L}(\theta_G, \theta_D)$. As the goal of the generator is to fool the discriminator, we can neatly re-use the same loss function $\mathcal{L}(\theta_G, \theta_D)$ to optimize the generator parameters, but we need to maximize rather than minimize the objective, i.e.

$$\theta_G^* = \arg \max_{\theta_G} \min_{\theta_D} \mathcal{L}(\theta_G, \theta_D). \quad (3.32)$$

The above equation succinctly summarizes the training procedure for a GAN model, which contains two steps. Firstly, given the current parameters of the generator θ_G , we need to update the discriminator's parameters θ_D in the direction that minimizes $\mathcal{L}(\theta_G, \theta_D)$. Secondly, given the updated θ_D , we now update the generator's parameters θ_G in the direction that maximizes $\mathcal{L}(\theta_G, \theta_D)$. This update procedure will continue until the loss function $\mathcal{L}(\theta_G, \theta_D)$ converges. The optimized set of θ_G^* and θ_D^* is often referred to as Nash equilibrium in the game theory context [68]. Nash equilibria occur at locations of the parameter space where the loss function is simultaneously a local minimum with respect to some parameters and a local maximum with respect to the rest of parameters.

We have seen GANs being applied to some interesting applications, especially when multiple outputs are all viable, i.e. the distribution of the model output is multimodal. For example, SRGAN [69] provides an impressive solution to the task of estimating the high-resolution image given a single low-resolution image as input. The adversarial objective in GANs' formulation prevents the method to penalize the different high resolution images which are all validate predictions given the same input image. iGAN [70] can generate an image based on some initial sketches drawn by a user. This opens a door to human-robot collaborative arts. Pix2pix [71] allows a feature phrased by the authors as "image-to-image translation", where an image of a certain type is converted to another image of the same content but in a different format, e.g. a sketch of a bag is translated to a real photo of the bag or a semantic map of a street scene is translated to a real photo of the same scene. This work offers a new direction to synthetically generate more data given the existing data one has.

Although GANs have resulted in some very innovative and impressive machine learning applications, they often struggle with the following challenges [67], [72]. Firstly, the adversarial nature of GANs learning objective often causes unstable training, i.e. the loss function fails to converge. Secondly, GANs often experience a problem called *mode collapse*, where the generator fails to generate a diverse set of data samples but only generates samples from a small region of the true data distribution [67], [73]. Thirdly, there is no explicit mapping from the data space to

the latent space in the original GAN's formulation. If such a mapping is needed for a particular application, then this requires an additional module to be added to the GAN model. We have seen great progress from the machine learning research in addressing the above challenges [74]–[77] and it is still an active research area to further improve GANs performance.

Compared to GANs, VAEs often have a more stable training, thanks to the exact learning objective - the ELBO loss - derived from the variational inference framework. As VAEs are trained with an auto-encoding (or reconstruction) module, the mode collapse behaviour is less severe in the VAE models, because the latent distribution is optimized to contain a diverse set of data samples as presented in the given dataset. Finally, VAEs offer the mapping from the data to the latent space and vice versa.

3.3.2 Normalizing flows

Normalizing flows [65], [66] offer a different way to estimate the distribution of a data variable $\mathbf{x}$. Unlike VAEs and GANs where little constraints are made on the transformation between the data variable $\mathbf{x}$ and the latent variable $\mathbf{z}$, normalizing flow approaches require the transformation between $\mathbf{x}$ and $\mathbf{z}$ to be bijective or invertible, i.e. the mapping $f : \mathbb{R}^n \mapsto \mathbb{R}^n$ satisfies $\mathbf{x} = f(\mathbf{z})$ and $\mathbf{z} = f^{-1}(\mathbf{x})$, where $f^{-1}(\cdot)$ indicates the inverse function of $f(\cdot)$. With such transformations, we can apply the change of variables equation and establish the connection between $p(\mathbf{x})$ and $p(\mathbf{z})$ as:

$$p_x(\mathbf{x}) = p_z(\mathbf{z}) \left| \det \left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right) \right|. \quad (3.33)$$

Applying the identity $\det(\mathbf{A}^{-1}) = \det(\mathbf{A})^{-1}$ where $\mathbf{A}$ is an invertible matrix, we can obtain the following change:

$$\det \left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right) = \det \left(\frac{\partial \mathbf{x}}{\partial \mathbf{z}} \right)^{-1} = \det \left(\frac{\partial f(\mathbf{z})}{\partial \mathbf{z}} \right)^{-1}. \quad (3.34)$$

Therefore, the density $p_x(\mathbf{x})$ is equivalent to:

$$p_x(\mathbf{x}) = p_z(\mathbf{z}) \det \left(\frac{\partial f(\mathbf{z})}{\partial \mathbf{z}} \right)^{-1}. \quad (3.35)$$

We can estimate the unknown parameters in the transformation $f(\cdot)$ by maximizing the log likelihood of the above equation, i.e.

$$\log p_x(\mathbf{x}) = \log p_z(\mathbf{z}) - \log \det \left(\frac{\partial f(\mathbf{z})}{\partial \mathbf{z}} \right). \quad (3.36)$$

The above log likelihood can be evaluated exactly, if the density $p_z(\mathbf{z})$ is tractable and the determinant of the Jacobian of $f(\cdot)$, i.e. $\det \left(\frac{\partial f(\mathbf{z})}{\partial \mathbf{z}} \right)$, is also tractable.

Indeed, these are the exact design features of normalizing flow methods. Normalizing flow methods assume that the distribution for the latent variable $p_z(\mathbf{z})$ is of simple, parametric forms - often a Gaussian distribution is used, and that the transformation $f(\cdot)$ is carefully designed so that the computation of $\det \left(\frac{\partial f(\mathbf{z})}{\partial \mathbf{z}} \right)$ can be done efficiently. For example, NICE [65] proposes additive coupling layers to achieve this. To do so, the latent variable $\mathbf{z}$ is divided into two halves - $\mathbf{z}_1$ and $\mathbf{z}_2$. Then the forward transformation $\mathbf{x} = f(\mathbf{z})$ is defined as follows:

1. $\mathbf{x}_1 = \mathbf{z}_1$.
2. $\mathbf{x}_2 = \mathbf{z}_2 + h_\theta(\mathbf{z}_1)$, where $h_\theta(\cdot)$ denotes a neural network with parameters θ .

Correspondingly, the inverse transformation $\mathbf{z} = f^{-1}(\mathbf{x})$ can be computed as:

1. $\mathbf{z}_1 = \mathbf{x}_1$.
2. $\mathbf{z}_2 = \mathbf{x}_2 - h_\theta(\mathbf{x}_1)$.

A convenient feature of the above transformation $f(\cdot)$ is that the Jacobian of $f(\mathbf{z})$, i.e. $\frac{\partial f(\mathbf{z})}{\partial \mathbf{z}}$ is a lower triangle matrix. This means the determinant of this Jacobian is the product of all the elements along the diagonal of the Jacobian, which is 1 as:

$$\frac{\partial \mathbf{x}_1}{\partial \mathbf{z}_1} = 1 \quad \text{and} \quad \frac{\partial \mathbf{x}_2}{\partial \mathbf{z}_2} = 1. \quad (3.37)$$

In general, normalizing flow methods offer a stable learning algorithm to estimate the data distribution $p(\mathbf{x})$. Such approaches are extremely suited for density estimation at low dimensions, as they are capable of representing distributions of very complex forms through transformations from some simple base distributions. However, normalizing flows often struggle when dealing with high dimensional variables. This is perhaps because normalizing flows expect to obtain an one-to-one mapping between every sample in the latent space and the sample in the data space. When the space expands to the high-dimensional domain, the search space for such setting is enormous and to obtain the right transformation $f(\cdot)$ we might need to have a flexible parametric form of the transformation and need a sufficient number of training examples to converge to the right optimum of the log likelihood

objective.

3.4 Efforts to improve the VAE modelling

With an increasing demand to apply VAEs in different application scenarios, there is a growing interest in improving the generative modelling capacity of VAEs further. This is because any improvement in the derived generative model, such as an encoding function that can better reserve information in the raw data and a generative model that can sample more diverse and realistic data, will likely result in better performance in the downstream tasks. In this section, we give a summary of research efforts that have been dedicated to improving the VAE models.

Flexible modelling of the posterior distribution $q_\phi(\mathbf{z}_i|\mathbf{x}_i)$. The original VAEs adopt a simple parametric model of the posterior distribution, i.e. a Gaussian distribution with a diagonal covariance matrix. Such formulation makes the computation of the ELBO objective easy and stable. However, this modelling assumption can limit the expressiveness of the inferred posterior and lead to approximation error where the inferred posterior distribution fails to represent the true posterior distribution for individual data samples [78]. To address this issue, proposals of more flexible modelling of the posterior distribution have been put forward.

Normalizing flows, as we introduce in Section 3.3.2, are often used to in this context to provide more flexible modelling of the posterior distributions. To do this, we first let the encoder to predict a latent variable that follows a simple base distribution, i.e. $\mathbf{z}_0 \sim q(\mathbf{z}_0)$. Then we can transform the base distribution under a chain of K invertible mapping f_k . Using the Equation (3.35), the new distribution becomes:

$$q_K(\mathbf{z}_K) = q(\mathbf{z}_0) \prod_{k=1}^K \det\left(\frac{\partial f_k}{\partial \mathbf{z}_{k-1}}\right)^{-1}, \quad (3.38)$$

from which we can compute the log density as:

$$\log q_K(\mathbf{z}_K) = \log q(\mathbf{z}_0) - \sum_{k=1}^K \det\left(\frac{\partial f_k}{\partial \mathbf{z}_{k-1}}\right). \quad (3.39)$$

We can substitute the posterior distribution $q_\phi(\mathbf{z}_i|\mathbf{x}_i)$ in the ELBO with this new distribution $q_K(\mathbf{z}_K)$. As long as $\det\left(\frac{\partial f_k}{\partial \mathbf{z}_{k-1}}\right)$ for $k = 1, \dots, K$ can be easily computed and is differentiable with respect to $\mathbf{z}_{k-1}$, we are able to update the model parameters by optimizing the ELBO via stochastic gradient descent as before. So the key challenge here is to define an invertible function f_k that satisfies the above condition.

To do so, [14] propose a simple flow with the following f transformation:

$$f(\mathbf{z}) = \mathbf{z} + \mathbf{u}h(\mathbf{w}^T\mathbf{z} + b), \quad (3.40)$$

where $\mathbf{u}$, $\mathbf{w}$ are vectors of the same dimension as $\mathbf{z}$, b is a scalar and $h(\cdot)$ is a smooth, differentiable non-linear function. [15] propose a flow that utilizes an auto-regressive mapping, i.e.

$$f(\mathbf{z}_d) = \mu(\mathbf{z}_{1:d-1}) + \sigma(\mathbf{z}_{1:d-1}) \cdot \mathbf{z}_d, \quad (3.41)$$

where $\mu(\cdot)$, $\sigma(\cdot)$ are complex mapping implemented by neural networks and $\mathbf{z}_d$ denotes the d -th dimension of the latent variable. Both works show that with sufficient depth of the flow transformation, the transformed distribution is able to model arbitrary complex distribution and the VAEs trained with such posterior distributions can reach higher likelihood score.

Adversarial Variational Bayes (AVB) [79] proposes to adopt an implicit distribution [80] parameterized by a neural network to model the posterior distribution. Such implicit distributions are very flexible and can approximate arbitrarily complex posterior distributions. To learn the parameters in the implicit posterior and the generative model, AVB introduces an additional discriminator network $T_\psi(\mathbf{x}_i, \mathbf{z}_i) = \log p(\mathbf{z}_i) - \log q_\phi(\mathbf{z}_i|\mathbf{x}_i)$, allowing the estimation of the ELBO objective. To infer the model parameters ϕ , θ and ψ , the authors also formulate the optimization of the ELBO as a two-player game between the discriminator and the VAE model in a similar way as how GAN models are trained. The authors show that AVB generate better quality samples and reaches higher likelihood score, compared to the original VAEs.

Flexible modelling of the prior distribution $p(\mathbf{z})$. The choice of the prior distribution has an important impact on the derived generative model. As we can see from Equation 3.12, the prior distribution directly regularizes the posterior distribution in the ELBO. An inflexible prior may place posterior distributions in a region of the latent space that should be left as low density area and penalize the posterior distributions in the high density areas. Therefore, several works have explored the idea of replacing the unit Gaussian prior with another more flexible distribution, leading to variants including a stick-breaking prior [81], a Chinese Restaurant Process prior [82], a Gaussian mixture (GM) prior [61], [83], [84] and an auto-regressive prior [85], [86]. All of these works report that higher likelihood scores can be reached with the adoption of more flexible prior models and many also show an improvement of the generation quality.

Addressing the dying units problem [54]. Sometimes we observe that VAEs have the tendency to ignore certain dimensions of the latent variable, leaving it the same as the prior distribution. This effect indicates that the derived VAE model fails to utilize the full capacity of the encoder model and learn an informative posterior distribution. [85] notices this problem in the context of an auto-regressive decoder, i.e. $p(\mathbf{x}|\mathbf{z}) = \prod_d p(\mathbf{x}_d|\mathbf{z}, \mathbf{x}_{<d})$. They address this problem by only limiting the input information that the decoder is conditioned on, i.e. $p(\mathbf{x}|\mathbf{z}) = \prod_d p(\mathbf{x}_d|\mathbf{z}, \mathbf{x}_{\text{WindowAround}(d)})$.

3.5 Summary

In this chapter, we have introduced VAEs as *a non-linear extension of the pPCA models*. We show that the VI framework can be used to estimate the model parameters in VAEs. To do so, we need to apply *amortization* to the posterior distribution, i.e. $q_{\phi_i}(\mathbf{z}_i) = q_{\phi}(\mathbf{z}_i|\mathbf{x}_i)$. This removes the need to have a set of local parameters for each data sample, which have to be (re-)optimized whenever a new data sample arrives. We also introduce other techniques, including Monte Carlo sampling, reparametrization trick and unbiased estimation via stochastic sampling, which are needed to allow efficient updates of model parameters with the stochastic gradient descent scheme.

To understand how well a VAE model estimate the target data distribution, we introduce four types of evaluation metrics that are commonly used in the VAE literatures. We have also discussed the differences between VAEs, GANs and normalizing flows. Compared to GANs, VAEs often have a more stable training, less severe mode collapse behaviours and a convenient inference model that allows efficient encoding of any data samples. Compared to normalizing flows, VAEs benefit from being able to derive a lower dimensional latent representation, whereas normalizing flows have to keep the latent variable of the same dimension as the data variable.

Finally, we discuss existing works that have been carried out to improve the generative modelling capacity of VAEs. We focus on three areas of improvement: 1) a more expressive posterior modelling, 2) a more informative prior modelling and 3) deriving a more powerful inference model.

With a good understanding of VAEs, we will introduce three challenges that we observe during the training and applications of VAEs in the next chapter. We will then present our solutions to these challenges in the subsequent chapters.

4

Challenges in the VAE modelling

In this chapter, we introduce two challenges that we often observe as the bottlenecks to deriving good VAE models. Firstly, we show that the constraint on the posterior distribution (hence, on the inference model (or the encoding function) of a VAE) from the prior distribution can be overly strict. This has a direct impact on the reconstruction and generation quality and implicitly it may cause the derived VAE model not being able to fit the given dataset well, hence leading to a low likelihood score. We refer to this effect as the *over-regularization* problem in VAE modelling. We show in this chapter that such over-regularization effects can happen either because of a variance parameter in the Gaussian decoder being set at a sub-optimal value, or because of an overly restrictive prior distribution. Secondly, we demonstrate that the auto-encoding function inferred from a VAE model may transform a data sample to something that is quite different. We refer to this effect as the *encoding drift* problem. This effect can be detrimental, as it indicates that crucial information in the raw data may be altered in the encoding processing and the posterior distribution may not be a good representation of the input sample to be used for downstream tasks.

4.1 Over-regularized encoding

A problem that often limits the quality of the derived representation from VAE models is the *over-regularization* issue. Recall the ELBO objective function introduced in Section 3.1.1 and included again below:

$$\text{ELBO}_i(\phi, \theta) = \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}_i|\mathbf{x}_i)} \left[\log p_\theta(\mathbf{x}_i|\mathbf{z}_i) \right]}_{\text{① reconstruction likelihood}} - \underbrace{D_{\text{KL}}(q_\phi(\mathbf{z}_i|\mathbf{x}_i) \parallel p(\mathbf{z}))}_{\text{② prior constraint}}. \quad (4.1)$$

The parameters in a VAE model are optimized by maximizing the above ELBO objective, which contains two terms: ① a reconstruction likelihood term that motivates the input information to be preserved through the auto-encoding process

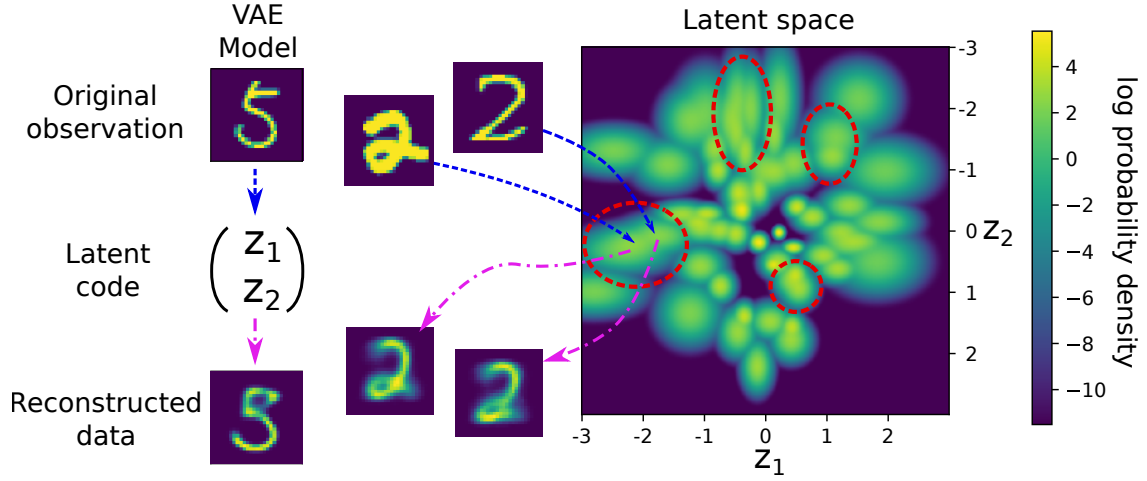


Figure 4.1: The effect of over-regularized ELBO on VAEs with a Gaussian decoder. Here we train a VAE to infer a 2-dimensional latent variable for MNIST dataset (left plot). The overly strict prior regularization leads to overly smooth latent space, where the posterior distributions of different input samples become indistinguishable. As a result, the reconstruction of different input samples end up being very similar (see the example of a pair of twos in the right plot).

and ② a prior constraint term that encourages individual sample encodings, i.e. Gaussian distributions formulated by $q_\phi(\mathbf{z}_i|\mathbf{x}_i)$, to stay as close as possible to a prior distribution $p(\mathbf{z})$, which is often chosen to be a zero-mean unit-variance Gaussian distribution. The ELBO objective is often described as a regularized objective for learning an effective auto-encoding function, where the amount of regularization is determined by the level of prior constraint enforced by the ② in the ELBO.

As the posterior distribution $q_\phi(\mathbf{z}_i|\mathbf{x}_i)$ appears in both terms in the ELBO, the level of the prior regularization on the posterior will also have an impact on the reconstruction quality and subsequently on quality of the inferred generative model. To see this, imagine we have an encoding function that maps all the input samples perfectly to the prior distribution. In this case, the prior constraint term of the ELBO objective is perfectly satisfied. However, there is no way that we can distinguish different samples from each other in their latent encodings. Therefore, the reconstruction likelihood term of the ELBO objective is severely compromised. As a result, the reconstruction of different input samples will end up being the same image [87], [88].

We have an example to illustrate such effects in Figure 4.1. Here we first train a VAE model with a 2D latent space for the MNIST dataset [39]. After the ELBO objective has converged, we encode 64 randomly drawn test samples and the encoding distributions in the 2D latent space are shown as the blobs in the density

plot on the right side of Figure 4.1. The over-regularization effect can be observed in the reconstructions of the two digit 2 images. Before the reconstruction, we can clearly distinguish the two images, as they bear clear visual differences - the two on the left hand side has a thicker stroke and has a little circle connecting the vertical line with the ending horizontal section, whereas the two on the right hand side has a thinner stroke and has a sharp corner connecting the vertical and horizontal section. After the auto-encoding process through the trained VAE (or reconstruction), these differences that distinguish one image from another have diminished and we can hardly tell the two images apart from their reconstructions.

Such effects occur, because the posterior distributions of the two samples $q_\phi(\mathbf{z}_i|\mathbf{x}_i)$ are overly regularized so that the details that distinguish one sample from another are no longer preserved. This is indeed the case, as we can see the posterior distributions of different input samples have merged into each other (see the dashed red circles where we highlight a few examples of such behaviours). As a result, crucial details from the original samples will be lost and samples generated from the derived latent variable model will show a limited diversity and lack important details that make individual samples appear to be realistic. It is worth noting that the impact of such effects depends on the downstream tasks. For example, if we would like to design a classifier that recognises different handwritten digits, then the information loss described above might not be a problem, as long as the details that distinguish different digits are preserved in the latent representations of the input images. However, if the downstream task is a fraud detection task where we would like to decide whether a piece of handwritten texts are from a specific individual, then the missing details described above will limit how well we can achieve on this task. In an ideal setting, we would like to reuse the derived representations of the raw data for a range of downstream tasks, each of which might be sensitive to a different set of features in the raw data. Therefore, it is a good principle to preserve as much information as possible from the raw data in the derived representations, so we will not limit the performance of downstream tasks at the stage of projecting the raw data to the representations.

In this thesis, we focus on two possible factors that can contribute to the over-regularization effects as described above.

4.1.1 Gaussian decoders with an arbitrary variance parameter

In the first case, we consider a VAE model with a Gaussian decoder, where the reconstruction likelihood $p_\theta(\mathbf{x}_i|\mathbf{z}_i)$ in ① from Equation 4.1 is modelled as a Gaussian

distribution, i.e.

$$p_{\theta}(\mathbf{x}_i^k | \mathbf{z}_i) = \frac{1}{(2\pi)^{\frac{1}{2}} \sigma} \exp \left(- \frac{(g_{\theta}^k(\mathbf{z}_i) - \mathbf{x}_i^k)^2}{2\sigma^2} \right), \quad (4.2)$$

where $g_{\theta}(\cdot)$ represents the decoding function that transforms a latent encoding $\mathbf{z}_i$ to a data sample $\mathbf{x}_i$; σ^2 is a variance parameter, which indicates the precision of the decoding function and usually accounts for the noise and the unmodelled effects in the dataset; and k represents the k -th dimension of the data variable $\mathbf{x}_i$. Due to the conditional independence, the prediction of the complete data sample $\mathbf{x}_i$ given $\mathbf{z}_i$ is $p_{\theta}(\mathbf{x}_i | \mathbf{z}_i) = \prod_{k=1}^d p_{\theta}(\mathbf{x}_i^k | \mathbf{z}_i)$, where d is the data dimension. Gaussian decoders are commonly used for modelling continuous, real-valued data variables $\mathbf{x}$.

Taking log of $p_{\theta}(\mathbf{x}_i | \mathbf{z}_i)$ and substituting it into Equation (4.1), we obtain the (negative) ELBO_{*i*} objective for a VAE model with a Gaussian decoder as

$$\underbrace{\frac{1}{2\sigma^2} \mathbb{E}_{q_{\phi}(\mathbf{z}_i | \mathbf{x}_i)} \left[\sum_{k=1}^d (g_{\theta}^k(\mathbf{z}_i) - \mathbf{x}_i^k)^2 \right]}_{\textcircled{3}} + \underbrace{D_{\text{KL}}[q_{\phi}(\mathbf{z}_i | \mathbf{x}_i) \parallel p(\mathbf{z})]}_{\textcircled{2}} + d \log \sigma + \frac{d}{2} \log(2\pi). \quad (4.3)$$

In the above equation, $\textcircled{1}$ denotes the element-wise square reconstruction error between a data sample $\mathbf{x}_i$ and the output from the decoder $g_{\theta}(\mathbf{z}_i)$; $\textcircled{2}$ denotes the prior constraint the same as the $\textcircled{2}$ from Equation (4.1); and, interestingly, $\textcircled{3}$ - a single parameter that depends on the variance parameter σ^2 - appears as a natural weight that balances between $\textcircled{1}$ and $\textcircled{2}$. σ^2 is commonly set as 1, which allows the ELBO objective from the above equation to be simplified to the sum of a least square reconstruction error and the prior regularization. However, setting σ^2 to an arbitrary value, such as 1, is likely to cause the over-regularization effects as described previously. In fact, some works [26], [27] argue that the over-regularization is necessary to derive a disentangled representation. Hence, they propose to reduce the influence of the reconstruction error term in the ELBO by adopting $\sigma^2 \gg 1$.

4.1.2 Overly restrictive prior

In the second case, we examine the choice of the prior distribution on the regularization imposed on the posterior distribution. To see this, we re-arrange the

ELBO objective as follows:

$$\begin{aligned}
 \text{ELBO}(\theta, \phi, \mathcal{D}_N) &= \mathbb{E}_{p_{\mathcal{D}}(\mathbf{x})} \left[\mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} [\log p_{\theta}(\mathbf{x}|\mathbf{z})] - D_{KL}(q_{\phi}(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z})) \right] \quad (4.4) \\
 &= \mathbb{E}_{p_{\mathcal{D}}(\mathbf{x})} \left[\underbrace{\mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} [\log p_{\theta}(\mathbf{x}|\mathbf{z})]}_{\text{① reconstruction likelihood}} - \underbrace{\mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} [\log q_{\phi}(\mathbf{z}|\mathbf{x})]}_{\text{② posterior entropy}} + \underbrace{\mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{z})]}_{\text{③ cross-entropy wrt prior}} \right], \quad (4.5)
 \end{aligned}$$

where we break down the KL divergence term in Equation (4.4) using the definition of KL divergence (see Equation (A.35) in Appendix A.4). Here we can see that ELBO consists of three terms: ① a reconstruction likelihood that encourages good reconstruction through the auto-encoding process, ② a negative posterior entropy that favours the posterior distribution $q_{\phi}(\mathbf{z}|\mathbf{x})$ for a data sample to have a large variance, and finally ③ a cross-entropy between the posterior $q_{\phi}(\mathbf{z}|\mathbf{x})$ and the prior $p(\mathbf{z})$ that regularizes the posteriors to comply with the target prior distribution. We can further re-arrange ③ into a cross-entropy term between the aggregate posterior $q_{\phi}(\mathbf{z})$ and the prior $p(\mathbf{z})$, as shown below:

$$\mathbb{E}_{p_{\mathcal{D}}(\mathbf{x})} [\mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{z})]] = \int_{\mathbf{z}} \int_{\mathbf{x}} \log p(\mathbf{z}) q_{\phi}(\mathbf{z}|\mathbf{x}) p_{\mathcal{D}}(\mathbf{x}) d\mathbf{x} d\mathbf{z} \quad (4.6)$$

$$= \int_{\mathbf{z}} \log p(\mathbf{z}) \left(\int_{\mathbf{x}} q_{\phi}(\mathbf{z}|\mathbf{x}) p_{\mathcal{D}}(\mathbf{x}) d\mathbf{x} \right) d\mathbf{z} \quad (4.7)$$

$$= \int_{\mathbf{z}} \log p(\mathbf{z}) \mathbb{E}_{p_{\mathcal{D}}(\mathbf{x})} [q_{\phi}(\mathbf{z}|\mathbf{x})] d\mathbf{z} \quad (4.8)$$

$$= \int_{\mathbf{z}} \log p(\mathbf{z}) q_{\phi}(\mathbf{z}) d\mathbf{z} \quad (4.9)$$

$$= \mathbb{E}_{q_{\phi}(\mathbf{z})} [\log p(\mathbf{z})], \quad (4.10)$$

where the aggregate posterior $q_{\phi}(\mathbf{z})$ is defined in Equation 3.26 in Section 3.2.3. This re-arrangement shows that the term ③ in the ELBO objective encourages the prior $p(\mathbf{z})$ and the inferred latent data distribution $q_{\phi}(\mathbf{z})$ to match each other.

When an overly restrictive prior is used (such as a unit Gaussian distribution in most of the VAE models), the learnt data distribution will fail to represent the true data distribution. We demonstrate this effect in Figure 4.2, where we compare the inferred data distribution represented by the aggregate posterior from a VAE model with the unit Gaussian prior distribution. We can clearly see that the two distributions are different. Such differences may have two impacts. Firstly, if we use the prior distribution to generate data samples, we are likely to encounter unrealistic samples as the prior distribution can draw latent samples in the low density region of the true data distribution. Secondly, the auto-encoding might be overly regularized

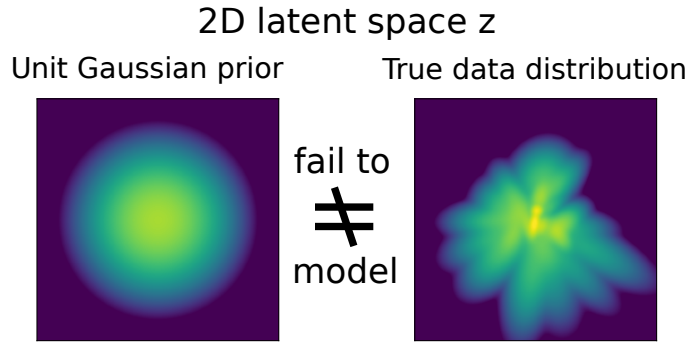


Figure 4.2: The zero-mean, unit-variance Gaussian can be overly restrictive for modelling the VAE’s prior distribution. Here we show the difference between the unit Gaussian prior distribution and an approximation of the true data distribution estimated by the aggregate posterior from a trained VAE model for the MNIST dataset. We can see that the prior distribution fails to represent the complex structures, such as multiple modes and low density areas, in the true data distribution. This is known as the prior hole problem [89].

by the prior distribution and place a posterior distribution in the areas of the latent space that should be kept as low density area. Therefore, we are likely to infer a generative model that is sub-optimal to describe the true data distribution.

Clearly, the level of regularization imposed by the prior distribution on the posterior distribution as defined in $D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) || p(\mathbf{z}))$ from the ELBO objective has a huge impact on the derived representation in a VAE model. We propose our solution to this challenge in Section 5.1.

4.2 Encoding drift

A great advantage of VAE models over the other generative models, such as GANs, is that we can derive both a generative function (modelled by the conditional distribution $p_\theta(\mathbf{x}_i|\mathbf{z}_i)$) and an inference function (modelled by the posterior distribution $q_\phi(\mathbf{z}_i|\mathbf{x}_i)$) at the same time. With these two functions, we can easily map a data sample to its latent representation in a lower dimensional space and effectively use the latent representation of a data sample for any subsequent tasks, such as making a decision based on the input observation. This underlying assumption for this scheme to work is that the inference function (or the encoding function) can preserve the salient information in the input sample that might be relevant for the downstream tasks. As the downstream task might not be known at the time a VAE model is being trained or even with the downstream task defined it might not be clear what information in the input sample is crucial to the task, it is sensible to favour an inference function that can preserve as much as possible information in

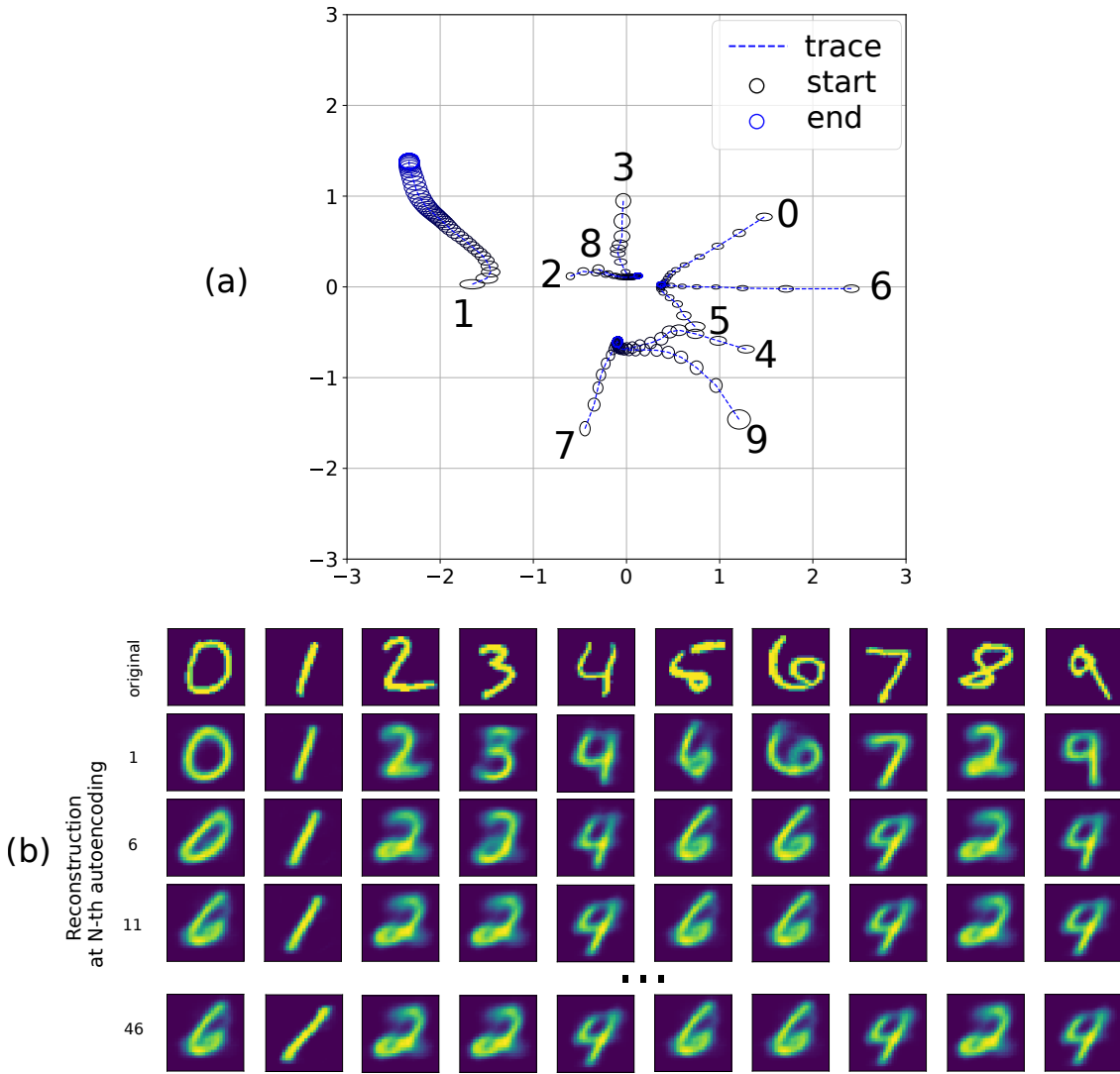


Figure 4.3: Information shift caused by the inconsistency between the inference (or encoding) and the generative (or decoding) model in a VAE. Here we apply the inferred encoding and decoding function from a VAE 50 times on a batch of 10 test samples. (a) The posterior distributions of these reconstructions drift further and further away from the posterior distribution of the input sample. (b) Such drifts cause the reconstruction to deviate away from the original input and eventually become a shape that is completely different.

the raw data. In the ideal case, we would want to have a perfectly matched pair of the inference and the generative function. Any data samples that are encoded to their latent representations using the inference function can be decoded back to its original form using the generative function.

However, it is very difficult to derive a VAE that contains a perfectly matching inference (or encoding) and generative (or decoding) model [78]. We are interested in finding out what the impacts of the imperfect match between the inference model

and the generative model have on the latent representations. Specifically, what information in an input sample would be altered by this imperfect auto-encoding mapping and can the latent representations still serve as a truthful image of the raw data?

To answer these questions, we carry out an experiment to investigate the changes of the latent representations (the posterior distributions $q_\phi(\mathbf{z}_i|\mathbf{x}_i)$) by applying the inferred auto-encoding (or reconstruction) repeatedly on the same input sample. The design principle behind this investigation is to make the information loss caused by the imperfect auto-encoding function visible by accumulating it over multiple rounds of such mapping.

We demonstrate the results of this investigation in Figure 4.3. Here we first train a VAE model for the MNIST digit dataset [39] under the ELBO objective as introduced in Section 3. Then we apply the inferred encoding and decoding function 50 times on a batch of test samples. In Figure 4.3(a), we show the posterior distributions (i.e. the latent representations) of the input sample and its subsequent 49 reconstructed samples on 10 digits. Each posterior distribution is shown as an ellipse, the centre of which corresponds to the mean of the Gaussian distribution and the two axes correspond to the variance. We can see that the posterior distributions are drifting further away as the auto-encoding function is repeatedly applied to the input sample and its reconstructions. This drift results in the reconstructed samples to diverge further from the input sample as the iterations of the auto-encoding continues. For example, as shown in Figure 4.3(b), the digit 0, 5 and 6 all slowly transform into an image of a blurry 6. Similarly, the digit 2, 3 and 8 are transformed into a blurry 2, whereas 4, 7 and 9 share the sample end image - a shape that lies in between 4 and 9. The transformation of the digit 1 behaves slightly different. It undergoes a gradual rotation and ends up with a quite tilted image of 1.

From this investigation, we demonstrate that the inconsistency between the inference (or encoding) and the generative (or decoding) model causes a drifting behaviour of the posterior distributions away from the posterior of the input sample. Although the drifts caused by a single auto-encoding mapping may not appear to be significant, we have shown that the auto-encoding mapping can be in the direction of completely altering the information in an input sample, such as a digit being modified in the direction of morphing into another digit. Furthermore, the magnitude of such drifts can grow bigger as we increase the dimension of the latent variable. Therefore, it is sensible to mitigate such drifts and reduce the inconsistency between the inference (or encoding) and the generative (or decoding) model. We

will introduce our solution to this problem in Chapter 6.

4.3 Summary

In this chapter, we have presented two challenges that may limit the generative modelling capacity of VAEs. We first show that an over-regularized posterior can result in damaging effects on the reconstruction quality. We show the two possible causes of such over-regularization effects. We then look at a drifting behaviour of the posterior distributions that are likely to be caused by the mis-match between the inference (or the encoding) and the generative (or the decoding) model. Such drifting indicates that the auto-encoding function inferred in a VAE model tends to transform the input sample to some signal that can be completely different. We denote this behaviour as a potential source of information loss.

In the next two chapters, we will introduce our solution to each of the above challenges and evaluate how well they can address the challenges.

5

Addressing the over-regularization issue

As shown in Section 4.1, an over-regularized posterior can cause damaging effect on the reconstruction quality and limit the ability of a VAE model to fit to the data samples well. We have analyzed two possible causes that can contribute to such effects.

In this chapter, we present a solution to each of the causes. Firstly, we show that the variance parameter σ^2 in the Gaussian decoder can be inferred from the training data to optimally set the regularization. Secondly, we address the restriction imposed by an inflexible prior, such as the zero-mean, unit Gaussian prior, by adopting a generative prior modelled by a VAE in the latent space. We show that both approaches can effectively reduce the over-regularization effects, improve the reconstruction quality and derive VAE models that reach high ELBO scores.

The relevant content in Section 5.2 is published as a conference paper - LaDDer: Latent Data Distribution Modelling with a Generative Prior at *The British Machine Vision Conference (BMVC)* 2020.

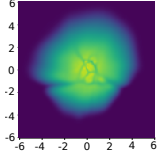
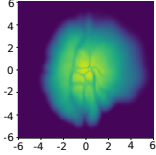
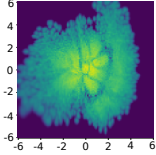
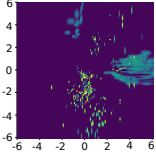
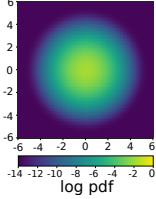
5.1 Inferring σ^2 from data

In Section 4.1.1, we show that the ELBO_i objective for a VAE with a Gaussian decoder can be expressed as Equation (4.3), which we present again as below:

$$\underbrace{\frac{1}{2\sigma^2} \mathbb{E}_{q_\phi(\mathbf{z}_i|\mathbf{x}_i)} \left[\underbrace{\sum_{k=1}^d \left(g_\theta^k(\mathbf{z}_i) - \mathbf{x}_i^k \right)^2}_{\textcircled{1}} \right]}_{\textcircled{3}} + \underbrace{D_{\text{KL}}[q_\phi(\mathbf{z}_i|\mathbf{x}_i) \parallel p(\mathbf{z})]}_{\textcircled{2}} + d \log \sigma + \frac{d}{2} \log(2\pi), \quad (5.1)$$

where σ^2 is the variance parameter for the conditional distribution $p_\theta(\mathbf{x}_i|\mathbf{z}_i)$, which is approximated by the Gaussian decoder. It is worth noting that other distributions can also be used to model the conditional distribution $p_\theta(\mathbf{x}_i|\mathbf{z}_i)$. For example,

Table 5.1: Visualization of the data distribution inferred by a VAE. Here we show the data distribution in the form of the aggregate posterior (as introduced in Section 3.2.3). We evaluate the aggregate posterior for VAEs trained with four different σ^2 values (top row). σ^2 has a clear impact on how tightly individual posterior distributions are placed next to each other and the difference between the aggregate posterior and the prior distribution (bottom row, estimated by the $D_{\text{KL}}(\hat{q}(\mathbf{z})||p(\mathbf{z}))$ using a Gaussian mixture approximation for the aggregate posterior).

σ^2	1	0.5	0.035 (Optimal)	0.01	Prior
Aggregate posterior					
$D_{\text{KL}}(\hat{q}(\mathbf{z}) p(\mathbf{z}))$	0.094 ± 0.003	0.132 ± 0.003	0.422 ± 0.011	1.552 ± 0.034	

truncated Gaussian distributions are better suited to model the image data, as the intensity values are limited to a finite range as opposed to the infinite support in the Gaussian distributions. Adopting truncated Gaussian distributions will effectively introduce a normalisation factor in the above equation and the rest of our analysis will still remain the same. If we consider the image data as a set of discrete variables each corresponding to a pixel that can take a single integer value between 0 and 255, then categorical distributions are a natural choice. The analysis we derive in this section does not apply to such distributions.

From the above equation, we can see that σ^2 naturally adjusts the relative weight between the reconstruction error in ① and the prior regularization on the posterior distribution in ②. Setting this σ^2 parameter to an arbitrary value can cause the regularization being applied overly strong or not enough. We demonstrate such effects in Table B.1. Here we train VAEs with σ^2 being set at 4 different values ranging from 0.01 to 1 (adopted in the original VAEs [16]). The value of σ^2 has a direct impact on how sharp the posterior distribution $q_\phi(\mathbf{z}_i|\mathbf{x}_i)$ is and subsequently how similar the aggregate posterior $q_\phi(\mathbf{z})$ (defined in Equation (3.26)) is to the prior distribution.

In this section, we show that we can infer the value of σ^2 from the training data. This way, we will obtain the optimal balance between the conflicting learning objectives in the ELBO loss.

5.1.1 Optimizing σ^2 under ELBO

For fixed encoder and decoder parameters ϕ and θ , we can derive a closed form solution for the optimal variance σ_*^2 , by taking derivative of the ELBO defined in Equation (5.1) with respect to σ and setting the derivative to zero, i.e.

$$\sigma_*^2 = \frac{1}{dN} \sum_i^N \mathbb{E}_{q_\phi(\mathbf{z}_i|\mathbf{x}_i)} \left[\sum_{k=1}^d \left(g_\theta^k(\mathbf{z}_i) - \mathbf{x}_i^k \right)^2 \right]. \quad (5.2)$$

We use the above equation to estimate the value of σ^2 alongside with θ and ϕ . To do so, we first set σ^2 as 1 to start the training of the VAE parameters, i.e. θ and ϕ . At every iteration, we update θ and ϕ according to the stochastic gradient descent scheme as introduced in Section 3.1.2. After we've updated θ and ϕ , then we evaluate the optimal value of σ_*^2 given the current set of θ and ϕ according to the above equation. Note that we obtain an approximate of the true σ^2 by drawing a stochastic batch of training samples rather than evaluating the equation with all the training samples. After obtaining an updated σ^2 , we then use this new σ^2 to evaluate ELBO_i and update θ and ϕ accordingly.

5.1.2 Extending to sample-specific, pixel-wise variances

So far we have assumed that all the data samples share a single variance parameter σ^2 across all input dimensions. Now we would like to allow the σ^2 estimation to be different for different data samples and for different input dimensions. To achieve these goals, we need to replace the global variance parameter σ^2 in Equation (4.2) with a variance estimation function $\sigma_\gamma^k(\mathbf{z}_i)$ which is parameterized by a neural network with parameter γ and conditioned on a latent encoding $\mathbf{z}_i$ (where k denotes the k -th data dimension). The corresponding $\log p_\theta(\mathbf{x}_i|\mathbf{z}_i)$ now becomes

$$-\frac{d}{2} \log(2\pi) - \sum_{k=1}^d \left[\frac{1}{2(\sigma_\gamma^k(\mathbf{z}_i))^2} \left(g_\theta^k(\mathbf{z}_i) - \mathbf{x}_i^k \right)^2 + \log \sigma_\gamma^k(\mathbf{z}_i) \right]. \quad (5.3)$$

Substituting the above reconstruction log likelihood in Equation (4.1), we obtain the ELBO objective for a VAE model with sample-specific, pixel-wise variance parameters. Such variance parameters effectively highlight the regions where the VAE model is uncertain when it performs a reconstruction or a generation task.

The parameters γ which govern the prediction of these variance parameters can be inferred by optimizing the ELBO with the above reconstruction likelihood with respect to all the model parameters θ , ϕ and γ . However, we notice that training the VAE model with the additional pixel-wise variance estimator $\sigma_\gamma(\cdot)$ turns out

Table 5.2: Evaluating the generation quality. We compare the FID scores for VAEs trained by five different methods on MNIST, fashion MNIST and CelebA datasets. For all datasets, optimizing σ^2 achieves the best visual quality (i.e. low FID scores).

	Ours	VAE [16]	β -VAE [26]	DIP- VAE [24]	WAE [90]	Real images
MNIST	8.9	74.9	438.1	114.9	142.8	1.9
Fashion MNIST	10.6	123.8	281.5	103.6	149.5	0.7
CelebA	74.7	78.4	106.4	87.5	85.4	2.9

to be quite challenging and the optimization often gets stuck at predicting very small value variances. We reckon that this can be caused by the strong gradient of the $\log \sigma_\gamma^k(\mathbf{z}_i)$ in Equation (5.3) dominating the gradient update for γ and the strong negative gradient of the log function causes the $\sigma_\gamma(\cdot)$ to predict smaller and smaller variance values. To prevent this and obtain stable learning, we propose a staged learning process where we start training with first estimating a single global variance as described in Section 5.1.1 until convergence. Then we switch to the pixel-wise variance estimator with the autoencoder parameters (ϕ and θ) being updated from the values estimated from the previous stage. We also use the optimized global variance σ_*^2 as an effective lower bound to prevent the predicted variance from getting stuck with small values.

5.1.3 Performance gain resulted from optimizing σ^2

In this section, we show that optimizing the variance parameter σ^2 can lead to a better latent variable model in terms of generation quality and reaching a high likelihood score. We compare the performance of a VAE trained with an optimized σ^2 with a range of VAE models, including the original VAE model [16], [17], β -VAE [26], DIP-VAE [24] and WAE [90]. We compare the performance of these VAE models under two commonly used metrics: 1) FID score [63] (introduced in Section 3.2.4), which is designed to measure the visual quality of the generated samples from a generative model (the lower, the better); and 2) importance weighted autoencoder (IWAE) test log likelihood (LL) [60] (introduced in Section 3.2.1), which is proposed as a tighter lower bound to the log likelihood $\log p(\mathbf{x})$ than ELBO and, hence, gives a better evaluation on how well the inferred latent variable model approaches the true data log likelihood objective (the higher, the better).

We evaluate different VAE models' performance on three datasets, including

Table 5.3: Evaluating the IWAE test likelihood. We evaluate the IWAE test log likelihood (LL, higher is better) for VAEs trained by four different methods on MNIST and Fashion MNIST. For our method, we evaluate the LL for two different variance estimation settings: 1) a global variance of the entire dataset and 2) input dependent pixel level variances. Optimizing σ^2 results in a significant increase on the IWAE test LL. Pixel-level σ^2 gives much higher test LL than a single σ^2 shared by all images in a dataset.

		Optimizing σ^2 (Ours)		VAE	β -VAE	DIP-VAE
		dataset	pixel			
MNIST	Prior	1307	6037	-462	-730	-461
	Approx. Agg. Posterior	1323	6052	-461	-730	-460
Fashion MNIST	Prior	1145	5708	-463	-731	-463
	Approx. Agg. Posterior	1157	5724	-462	-731	-462

MNIST [39], Fashion MNIST [40] and CelebA [41]. For all datasets, images are treated as real-valued data so that our Gaussian decoder formulation is appropriate. We take the original VAE [16], [17] and β -VAE [26] as baseline methods. We also compare to WAE [90] and DIP-VAE [24]. More results are given in Appendix B.1.

Table 5.2 lists the result of the FID scores across 5 different methods, including the original VAE ([16], [17]), β -VAE ([26]), DIP-VAE ([24]), WAE [90] and our method where we optimize the variance parameter σ^2 in the ELBO objective as discussed in Section 5.1.1. For all three datasets, our method achieves the lowest FID score, i.e. generating samples with the best visual quality.

In Table 5.3, we list the IWAE test log likelihood evaluated over 500 test data samples with 20 importance points per test sample for the MNIST and the fashion MNIST dataset. Our method significantly improves the test log likelihood in comparison to the other VAE models. More interestingly, there is a huge performance boost when we extend the global σ^2 to pixel-wise σ^2 in our model. This indicates that predicting the pixel-wise variance results in a model that can explain the given dataset significantly better than just using a single-valued variance parameter.

5.1.4 Uncertainty indicated by the pixel-wise variances

As the estimated variances indicate where the reconstruction error tends to be high, we can use the pixel-wise variance prediction as a tool to indicate where the VAE model is uncertain in its reconstructed and generated samples. Figure 5.1 illustrates such uncertainty among the reconstructed samples for the MNIST and the CelebA dataset. Note that high σ^2 values highlight the regions where the re-

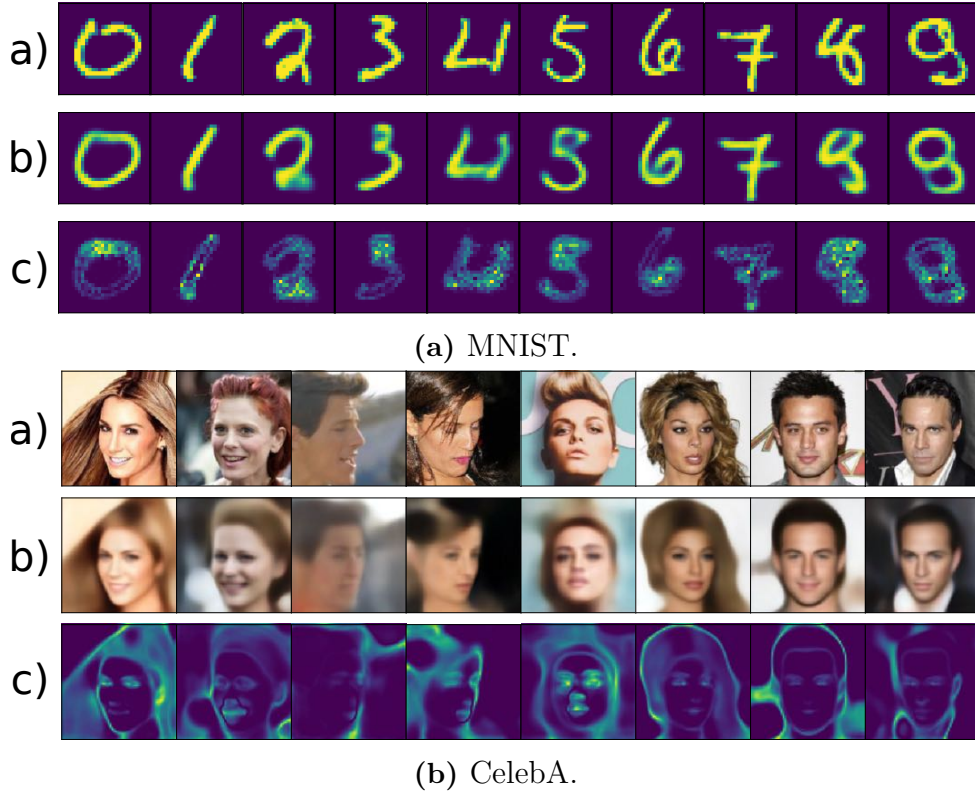


Figure 5.1: Uncertainty estimation for the reconstructed data. (a) Input images. (b) Reconstructed images. (c) Estimated pixel-wise σ^2 indicating uncertainty in the reconstruction.

construction appears to be different from the input data samples. During training, σ^2 prediction is trained to suppress such differences between the reconstructed and the original data samples. Hence, a high σ^2 value tends to be predicted at such regions. For example, we often observe relatively high reconstruction error for the digit zero at the top of the digit where the stroke starts and ends. This is likely because different people write zeros differently in this part and, hence, this is the most unpredictable part of this digit. We observe our VAE model with pixel-wise variances tend to predict high values of variances around this region to suppress the high reconstruction error.

Figure 5.2 illustrates the variances predicted in generated samples for both the MNIST and the CelebA datasets. Notice that in Figure 5.2b, the high uncertainty regions often correspond to the facial regions of the eyes, mouths, edges of hair and backgrounds, which again are highly variable across the dataset. We reckon the high variances among these regions are sensible, as they indicate that the VAE model is aware that there are many different ways to generate realistic samples in these regions and the current sample only renders one of them.

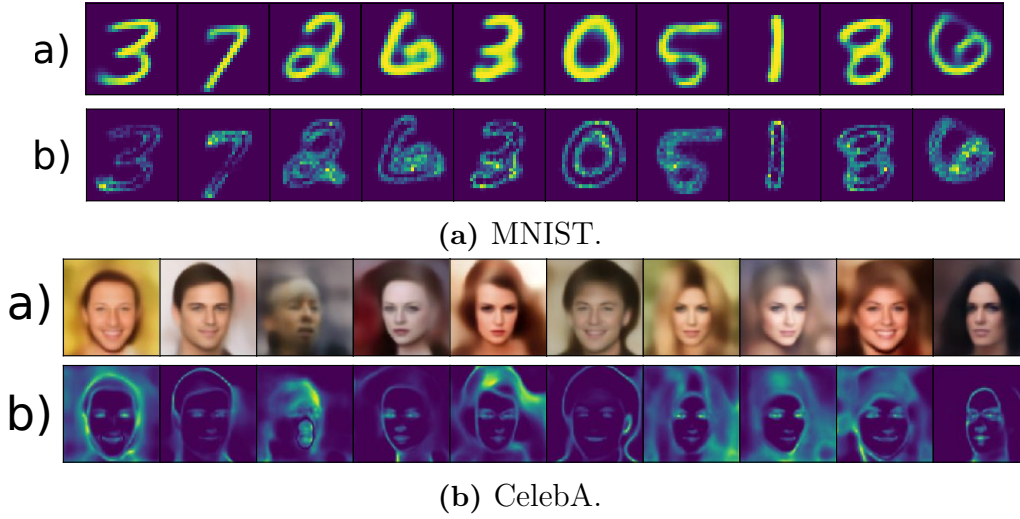


Figure 5.2: Uncertainty estimation for the generated data. (a) Generated samples. (b) Pixel-wise σ^2 predicted with the same latent embedding drawn from the approximate aggregate posterior. We can interpret this as the estimated uncertainty in the generation process of this sample.

5.2 Estimating the prior distribution

As discussed in 4.1.2, an overly simple prior, such as zero-mean, unit-variance Gaussian prior, can also cause the over-regularizing effect on the posterior distribution modelling and, hence, limit a VAE model’s ability to approximate the true data distribution.

In this section, we propose LaDDer, a method that allows us to accurately model the prior distribution in a VAE framework. The central idea of our method is a *meta-embedding* concept which, in short, derives a latent embedding of a latent embedding. Specifically as shown in Figure 5.3, our approach consists of multiple VAE models each acting on its predecessor’s latent representation and forming a ladder of encodings. We use a non-parametric mixture as the hyper prior for the innermost VAE. The hyper prior together with the intermediate encoders forms a generative prior for the outermost VAE. We learn the parameters of all the VAE networks, along with the non-parametric mixture, in a unified variational inference framework. From extensive experiments, we show that our LaDDer model is able to accurately estimate complex latent data distribution and results in improvement in the representation quality. We also propose a novel latent space interpolation method and show how to best utilize the derived latent distribution in further tasks.

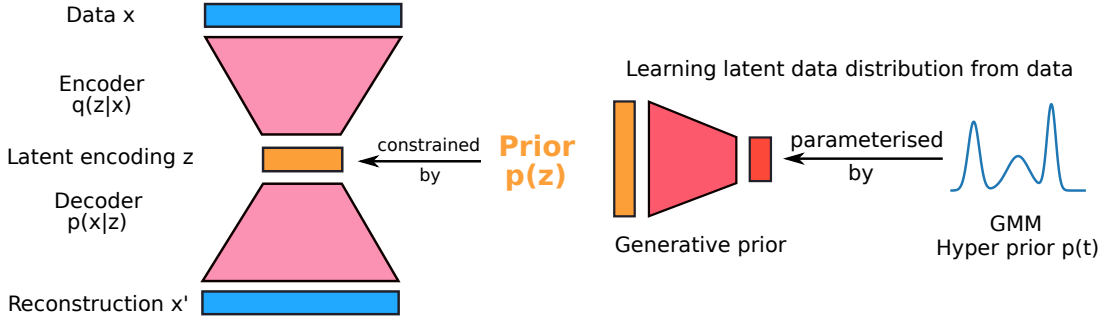


Figure 5.3: LaDDer’s key concept. LaDDer adopts a generative prior, which consists of a mixture hyper prior and a series of VAEs each acting on its predecessor’s latent encodings.

5.2.1 Introducing a generative prior

As shown in Figure 5.3, we propose a generative prior, which includes an additional VAE model to project the original data VAE’s encodings to an even lower dimensional space and a hyper prior for this prior VAE. We parameterize the hyper prior as a Gaussian mixture model (GMM). This design allows us to accurately estimate the latent data distribution, as the generative prior is sufficiently flexible to fit any arbitrarily complex distribution. At the same time, the optimization of our LaDDer model can be easily integrated into the VAE ELBO objective, which we will demonstrate in this section. In the end, we also demonstrate how to use the latent data distribution derived from our LaDDer learning to facilitate a latent space interpolation task.

We first introduce the VAE unit in our generative prior. Similar to the latent variable formulation of the original VAE model, we also formulate the prior distribution $p(\mathbf{z})$ in the VAE model by another latent variable model with a latent variable $\mathbf{t}$, i.e.

$$p_{\alpha}(\mathbf{z}, \mathbf{t}) = p_{\alpha}(\mathbf{z}|\mathbf{t})p(\mathbf{t}). \quad (5.4)$$

We reckon this prior latent variable model governs the generation of latent encodings $\mathbf{z}_i$ through 1) a hyper prior $p(\mathbf{t})$ that resides in an even lower dimensional space and 2) an encoding decoder $p_{\alpha}(\mathbf{z}|\mathbf{t})$ which is parameterized by a neural network with parameters α . To infer this prior latent variable model, we can introduce a variational encoder $q_{\beta}(\mathbf{t}|\mathbf{z})$ parameterized by a neural network with parameter β and learn both α and β by maximizing an ELBO objective $\text{ELBO}(\mathbf{z}; \alpha, \beta)$ similar

to Equation (4.5), i.e.

$$\text{ELBO}(\mathbf{z}; \alpha, \beta) \triangleq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\mathbb{E}_{q_\beta(\mathbf{t}|\mathbf{z})} [\log p_\alpha(\mathbf{z}|\mathbf{t})] - \mathbb{E}_{q_\beta(\mathbf{t}|\mathbf{z})} [\log q_\beta(\mathbf{t}|\mathbf{z})] + \mathbb{E}_{q_\beta(\mathbf{t}|\mathbf{z})} [\log p(\mathbf{t})] \right]. \quad (5.5)$$

Notice that $\text{ELBO}(\mathbf{z}; \alpha, \beta)$ is a lower bound to the log likelihood of the prior distribution $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\alpha(\mathbf{z})]$, which is equivalent to the cross-entropy term ③ in Equation (4.5). This connection allows us to integrate the learning objective of this prior VAE, i.e. $\text{ELBO}(\mathbf{z}; \alpha, \beta)$, into the ELBO for the original VAE for the data variable $\mathbf{x}$, i.e. $\text{ELBO}(\mathbf{x}; \theta, \phi)$, and obtain a new lower bound $\text{ELBO}'(\mathbf{x}; \theta, \phi, \alpha, \beta)$ to the ELBO for the data variable $\text{ELBO}(\mathbf{x}; \theta, \phi)$, i.e.

$$\text{ELBO}'(\mathbf{x}; \theta, \phi, \alpha, \beta) = \mathbb{E}_{p_D(\mathbf{x})} \left[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log q_\phi(\mathbf{z}|\mathbf{x})] + \text{ELBO}(\mathbf{z}; \alpha, \beta) \right]. \quad (5.6)$$

5.2.2 A Gaussian mixture model hyper prior

The prior VAE unit introduced in Equation 5.4 introduces a hyper prior distribution $p(\mathbf{t})$. Following a similar logic as the insight shown in Equation (4.6), we know that the optimal hyper prior distribution should be matched to the aggregate hyper posterior, i.e. $p(\mathbf{t}) \approx q_{\beta, \phi}(\mathbf{t})$. To facilitate the matching, we parameterize $p(\mathbf{t})$ with a Gaussian mixture model (GMM) of M components ($M \ll N$): $p(\mathbf{t}) = \sum_{m=1}^M w_m \mathcal{N}(\mathbf{t}; \mu_m, \Sigma_m)$, where w_m is the weight for each Gaussian mixture ($w_m > 0$ and $\sum_m w_m = 1$) and μ_m and Σ_m are the mean and covariance matrix for the m -th Gaussian mixture. To fit $p(\mathbf{t})$ to $q_{\beta, \phi}(\mathbf{t})$, we resort to variational inference techniques introduced in [91]. Here we give a high-level sketch of the algorithm and we refer interested readers to [91] for further mathematical details. Firstly, we define the generative model of the GMM by introducing the following distributions to the GMM parameters $(\mathbf{w}, \mu, \Sigma)$ and all N prior encoding samples $\mathbf{t}_i$ (the encodings of $\mathbf{z}_i$): $w_m \sim \text{Beta}(1, \alpha_0)$, $\mu_m \sim \mathcal{N}(0, \mathcal{I})$, $\Sigma_m \sim \mathcal{W}(d_t, \mathcal{I})$, $k_i \sim \text{Cat}(\mathbf{w})$, $\mathbf{t}_i \sim \mathcal{N}(\mu_{k_i}, \Sigma_{k_i})$, where k_i indicates the choice of mixture components for i -th sample $\mathbf{t}_i$. Secondly, we introduce the following variational distributions under the mean-field assumption for all the model variables $W = (\mathbf{w}, \mu, \Sigma, k_i)$: $w_m \sim \text{Beta}(\gamma_{m,1}, \gamma_{m,2})$, $\mu_m \sim \mathcal{N}(\mathbf{v}_m, \mathcal{I})$, $\Sigma_m \sim \mathcal{W}(a_m, \mathcal{B}_m)$, $k_i \sim \text{Discrete}(\mathbf{r}_i)$, where $\xi = (\gamma_{m,1}, \gamma_{m,2}, \mathbf{v}_m, a_m, \mathcal{B}_m, \mathbf{r}_i)$ for $m = 1, \dots, M$ and $i = 1, \dots, N$ denotes variational variables and needs to be optimized. Thirdly, a variational bound on the log likelihood of $p(\mathbf{t})$ is introduced as the learning objective to optimize ξ , as shown

below:

$$\log p(\mathbf{t}|\alpha_0, M) \geq \mathcal{L}(\mathbf{t}; \xi) = \mathbb{E}_{q_\xi(W)}[\log p(W, \mathbf{t}|\alpha_0, M)] - \mathbb{E}_{q_\xi(W)}[\log q_\xi(W)], \quad (5.7)$$

where $p(W, \mathbf{t}|\alpha_0, M)$ denotes the product of all the generative distributions introduced in step 1 and $q_\xi(W)$ denotes the product of all the variational distributions introduced in step 2.

Now substituting $\mathcal{L}(\mathbf{t}; \xi)$ in Equation (5.7) into Equation (5.5) to replace $\log p(\mathbf{t})$, we obtain a lower bound $\text{ELBO}'(\mathbf{z}; \alpha, \beta, \xi)$ to the prior ELBO $\text{ELBO}(\mathbf{z}; \alpha, \beta)$, i.e.

$$\text{ELBO}'(\mathbf{z}; \alpha, \beta, \xi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\mathbb{E}_{q_\beta(\mathbf{t}|\mathbf{z})} [\log p_\alpha(\mathbf{z}|\mathbf{t})] - \mathbb{E}_{q_\beta(\mathbf{t}|\mathbf{z})} [\log q_\beta(\mathbf{t}|\mathbf{z})] + \mathbb{E}_{q_\beta(\mathbf{t}|\mathbf{z})} [\mathcal{L}(\mathbf{t}; \xi)] \right]. \quad (5.8)$$

Now substituting $\text{ELBO}'(\mathbf{z}; \alpha, \beta, \xi)$ into the lower bound $\text{ELBO}'(\mathbf{x}; \theta, \phi, \alpha, \beta)$ in Equation (5.6) to replace $\text{ELBO}(\mathbf{z}; \alpha, \beta)$, we obtain a final lower bound $\text{ELBO}''(\mathbf{x}; \Theta)$ to the data likelihood $\mathbb{E}_{p_D(\mathbf{x})}[\log p_\theta(\mathbf{x})]$, i.e. $\mathbb{E}_{p_D(\mathbf{x})}[\log p_\theta(\mathbf{x})] \geq \text{ELBO}''(\mathbf{x}; \Theta)$, where $\Theta = [\theta, \phi, \alpha, \beta, \xi]$ denotes all the parameters in our model, which includes a VAE between the data variable $\mathbf{x}$ and the latent variable $\mathbf{z}$, a prior VAE between the latent variable $\mathbf{z}$ and the hyper latent variable $\mathbf{t}$ and a GMM hyper prior, and

$$\text{ELBO}''(\mathbf{x}; \Theta) = \mathbb{E}_{p_D(\mathbf{x})} \left[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log q_\phi(\mathbf{z}|\mathbf{x})] + \text{ELBO}'(\mathbf{z}; \alpha, \beta, \xi) \right]. \quad (5.9)$$

5.2.3 Optimizing model parameters via block coordinate ascent algorithm

To optimize all the model parameters $\Theta = [\theta, \phi, \alpha, \beta, \xi]$ involved in the lower bound defined in Equation (5.9), we resort to the block coordinate ascent algorithm as introduced in [47]. Notice that the GMM hyper prior's parameters ξ are only involved in the term $\text{ELBO}(\mathbf{t}; \xi)$ from Equation (5.7), so we can update ξ by maximizing $\text{ELBO}(\mathbf{t}; \xi)$ alone without worrying about other parameters. With the newly updated ξ , we can then update the prior VAE's parameter α, β , which are only presented in the prior VAE's ELBO $\text{ELBO}'(\mathbf{z}; \alpha, \beta|\xi)$ from Equation (5.8). Finally, with ξ, α, β updated, we can now update the parameters θ, ϕ of the VAE between $\mathbf{x}$ and $\mathbf{z}$ by maximizing $\text{ELBO}''(\mathbf{x}; \theta, \phi|\alpha, \beta, \xi)$ from Equation (5.9) with α, β, ξ being fixed. An algorithmic illustration for this optimization procedure is given below. Notice that we initialize the GMM hyper prior as an uninformative unit Gaussian to begin the optimization.

Algorithm 2: Block coordinate ascent to optimize the parameters in a LaDDer model

Data: A dataset of N observations $\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$

Result: Optimized model parameters $\Theta = [\theta, \phi, \alpha, \beta, \xi]$

```

1 Init
2   Neural network parameters:  $\phi, \theta, \alpha, \beta \leftarrow$  Random initialization;
3   Hyper prior distribution:  $\xi \leftarrow \text{Unit}(0, 1)$ ;
4   Count:  $t = 0$ ;
5   ELBO'': Compute  $\text{ELBO}''_0$  according to Equation (5.9),  $\Delta \text{ELBO}'' = 1$ ;
6   Termination condition:  $\epsilon = 0.01$ 
7 while  $\Delta \text{ELBO}'' > \epsilon$  do
8    $\theta, \phi \leftarrow$  Update by optimizing  $\text{ELBO}''(\mathbf{x}; \theta, \phi | \alpha, \beta, \xi)$  using
      AdamOptimizer;
9    $\alpha, \beta \leftarrow$  Update by optimizing  $\text{ELBO}'(\mathbf{z}; \alpha, \beta | \xi, \phi)$  using
      AdamOptimizer;
10   $\xi \leftarrow$  At the end of an epoch, update by optimizing  $\mathcal{L}(\mathbf{t}; \xi)$  using the
      updates in coordinate ascent [91];
11   $t = t + 1$ ;
12  Compute  $\text{ELBO}''_t$  according to Equation (5.9);
13   $\Delta \text{ELBO}'' = \text{ELBO}''_t - \text{ELBO}''_{t-1}$ 

```

5.2.4 Experimental results

We carry out extensive experiments on MNIST [39], Fashion MNIST [40] and CelebA [41] datasets to understand the performance of our LaDDer model. For all datasets, images are treated as real-valued data and we use a Laplace distribution to model the decoder $p_\theta(\mathbf{x}|\mathbf{z})$. The corresponding reconstruction likelihood (term ① in Equation (4.5)) is stated below:

$$p_\theta(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^d \frac{1}{2\sigma} \exp - \frac{|\mathbf{x}^k - \mu_\theta^k(\mathbf{z})|}{\sigma}, \quad (5.10)$$

where σ is a the variance parameter that we also optimize under the ELBO objective as introduced in Section 5.1. Compared to the commonly used Gaussian decoder, using a Laplace distribution to model the reconstruction likelihood term often gives sharper reconstruction, thanks to absolute difference in Equation (5.10) to model the reconstruction error.

We compare our method to four other approaches, including the original VAE with a normal prior [16], [17], VAE with a GMM prior [83], [84], [92], VAE with a hierarchical prior [93] where a generative prior with a normal hyper prior is used, and VampPrior [61] where the prior is modelled as an average encoding of a set

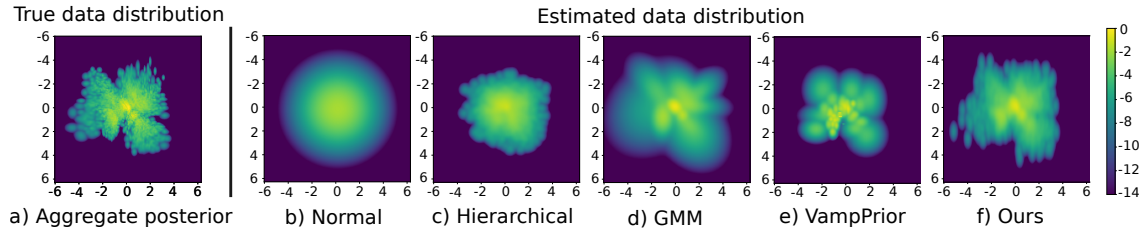


Figure 5.4: Visualizing the data distribution for MNIST in a 2D latent space. We first train a VAE model with the unit Gaussian prior. Then we fix the autoencoder model and fit different prior models to the derived encodings. The true data distribution (aggregate posterior $q_\phi(\mathbf{z})$) is shown in (a), where we encode 10k training images and visualize their posteriors. In (b)-(f), we visualize the pdfs of the 5 different prior methods. Our generative prior produces the best fit to the true data distribution.

of inferred pseudo-inputs. More results, details of data pre-processing and model architectures are given in Appendix B.3.1.

Our method better estimates the latent data distribution. We first show that our LaDDer model can achieve a better modelling of the latent data distribution, i.e. more accurate estimation of the aggregate posterior $q_\phi(\mathbf{z})$. Figure 5.4 visualizes the aggregate posterior $q_\phi(\mathbf{z})$ and the optimized priors from 5 different approaches, where VAE models are trained for 2D latent space on MNIST dataset. The true data distribution shown in Figure 5.4a contains complex low density regions, which correspond to natural boundaries between different object classes. An inflexible prior model, such as a unit Gaussian (Figure 5.4b), over-represents such low density regions. The hierarchical prior (Figure 5.4c) also fails to model $q_\phi(\mathbf{z})$ accurately, because the prior VAE is overly regularized by its inflexible hyper prior. The other three methods (Figure 5.4d-f), where the prior $p(\mathbf{z})$ is parameterized with mixture models, obtain better fitting. The difference across these three methods lies in the number of mixture components. Both the GMM prior and the VampPrior have a finite number of mixture components, whereas our LaDDer prior contains an infinite number of mixture components, as our prior is estimated by integrating over all t -values: $p_\alpha(\mathbf{z}) = \int p_\alpha(\mathbf{z}|\mathbf{t})p(\mathbf{t})d\mathbf{t}$. As a result, our method is able to smoothly fit any arbitrarily complex $q_\phi(\mathbf{z})$ with no artificial boundaries as the ones introduced in VampPrior.

Having a prior that can better fit $q_\phi(\mathbf{z})$ also leads to improved generation quality. Figure 5.5 gives examples of both reconstructed and generated samples from the 5 VAE models trained under the CelebA dataset. All 5 methods can produce good reconstruction, but the generation quality varies significantly. This indicates that the bad generation quality commonly reported in VAE models does not merely

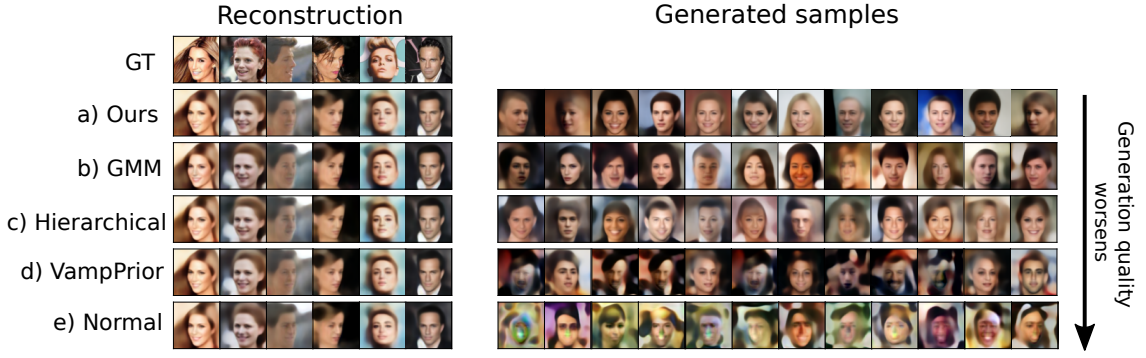


Figure 5.5: Comparison of reconstruction and generation quality across different VAE models. All five models trained with celebA dataset can produce similar quality reconstruction, but the generation quality varies significantly samples. The difference between reconstruction and generation quality indicates the importance of a prior model in modelling the data distribution and generating good quality samples. Our generative prior can model the data distribution $q_\phi(\mathbf{z})$ well, hence generating best quality samples with minimal gap to the quality of reconstructed samples. More examples are in Appendix B.3.2.1.

Table 5.4: Evaluating the generation and the reconstruction quality. We compute FID scores (lower is better) of generated and reconstructed samples. Our LaDDer model achieves the best sample quality and the minimal gap between the two.

	Normal	VampPrior	GMM	Hierarchical	Ours
Generation	250.3 ± 1.5	182.9 ± 1.2	140.4 ± 0.9	143.8 ± 1.0	132.7 ± 0.8
Reconstruction	100.3 ± 0.6	95.5 ± 0.7	102.3 ± 0.5	99.7 ± 0.6	95.3 ± 0.5
Difference	150.0 ± 2.1	87.4 ± 1.9	38.1 ± 1.4	44.1 ± 1.6	37.4 ± 1.3

result from the autoencoder architecture, but is also a result of the prior $p(\mathbf{z})$ failing to represent the topology of the learned latent data distribution. Generating from the prior that does not represent the true data distribution unsurprisingly produce unrealistic samples. Our proposed generative prior addresses this issue and produces generation quality that is almost as good as the model’s reconstruction quality. The improved generation quality is supported by the FID score [63], which is a quantitative measure to evaluate visual quality of image samples, shown in Table 5.4.

We also evaluate the cross-entropy $E_{q_\phi(\mathbf{z})}[\log p(\mathbf{z})]$ (③ in Equation (4.5)) between the aggregate posterior $q_\phi(\mathbf{z})$ and the prior $p(\mathbf{z})$ to measure the level of matching between the two distributions. Higher $E_{q_\phi(\mathbf{z})}[\log p(\mathbf{z})]$ indicates a better fit of the prior $p(\mathbf{z})$ to the learnt data distribution $q_\phi(\mathbf{z})$. As shown in Table 5.5, GMM

Table 5.5: Comparing the similarity between the aggregate posterior and the prior distribution. Here we evaluate the cross-entropy $E_{q_\phi(\mathbf{z})}[\log p(\mathbf{z})]$ (higher is better) between the aggregate posterior and the prior, which indicates the level of similarity between the two distributions. Our LaDDer model achieves the best fit to $q_\phi(\mathbf{z})$ in more complex datasets.

Dataset	Normal	VampPrior	GMM	Hierarchical	Ours
MNIST	-19.0 ± 0.1	-16.2 ± 0.1	-8.4 ± 0.9	-17.5 ± 0.1	-12.3 ± 0.1
fashion-MNIST	-52.5 ± 0.6	-60.8 ± 1.0	-45.2 ± 0.8	-63.8 ± 0.9	-32.8 ± 0.7
CelebA	-351.4 ± 4.3	-380.7 ± 3.4	-290.6 ± 54.8	-358.0 ± 7.3	-71.9 ± 2.1

Table 5.6: ELBO (higher is better) and per pixel **reconstruction error** (lower is better). Our method achieves highest ELBO and lowest reconstruction error across all datasets.

Metric	Dataset	Normal	VampPrior	GMM	Hierarchical	Ours
ELBO	MNIST	1528.6 ± 21.3	1476.3 ± 19.4	1558.2 ± 24.4	1534.9 ± 19.8	1562.5 ± 25.1
	fashion	1078.4 ± 23.4	1069.6 ± 22.1	1133.3 ± 23.9	1152.9 ± 24.9	1174.0 ± 29.7
	CelebA	55564.8 ± 1751.3	56463.4 ± 1683.9	54162.5 ± 1716.5	56176.2 ± 1763.9	58596.1 ± 1815.3
Per pixel recons. error ($\times 0.01$)	MNIST	2.3 ± 0.06	2.5 ± 0.06	2.3 ± 0.05	2.4 ± 0.06	2.2 ± 0.06
	fashion	4.3 ± 0.12	4.1 ± 0.11	4.0 ± 0.11	3.7 ± 0.11	3.6 ± 0.11
	CelebA	5.8 ± 0.21	5.9 ± 0.25	5.9 ± 0.20	5.7 ± 0.17	5.5 ± 0.18

priors can achieve very good fitting for relatively simple datasets, such as MNIST. However, when the dataset becomes more complex and the required latent dimension increases, such as fashion-MNIST and CelebA datasets, our generative prior, which is more flexible to fit complex distributions, obtains the highest cross-entropy and achieve the best fit.

Overall generative modelling performance improves. A more flexible prior $p(\mathbf{z})$ also leads to better generative modelling performance, in terms of better reconstruction quality and higher ELBO objective. This effect is clearly shown in Table 5.6, where our method produces the highest ELBO value and the lowest reconstruction error across all datasets. All models are trained under the same conditions and the evaluation is repeated for 5 times to estimate the variance of the results.

Furthermore, our LaDDer model produces an embedding scheme that better

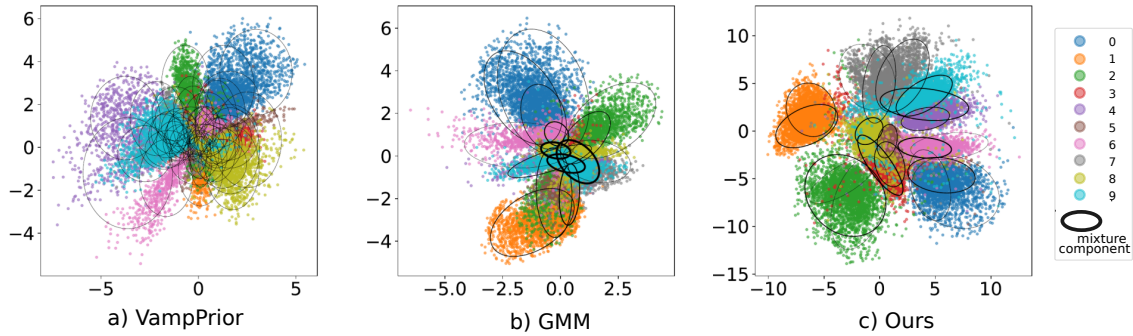


Figure 5.6: The mixtures in our hyper prior naturally aligns with MNIST class labels. We visualize encodings coloured by the digit labels for (a) VampPrior, (b) GMM prior and (c) our generative prior. Our method produces better clustering and the mixture components inferred in our method are naturally aligned with different classes.

preserves the semantics of the data. To see this, we visualize 30k latent encodings of the MNIST dataset in Figure 5.6, where a VAE of 2D latent space is trained with a GMM prior, VampPrior and our generative prior respectively. All encodings are coloured by their class labels. Our method clearly gives better clustering results, where encodings of different classes are better separated. Furthermore, the mixture components estimated in our method are better aligned with class labels, whereas the mixtures from VampPrior and GMM prior have significant overlaps and do not have a consistent correspondence to specific class labels.

5.2.5 Better latent space interpolation

In the previous sections, we introduce a generative prior and an optimization scheme to facilitate the accurate modelling of the latent data distribution for a given dataset. Now we demonstrate how to use the derived latent distribution in a latent space traversal task, where a path along the learnt data manifold needs to be inferred to interpolate between two data samples. Such interpolation has been used commonly in previous works [94]–[97] to illustrate the smoothness of the learnt data manifold. Traditionally, the traversal is done by a linear interpolation between the encodings of a pair of query images, which we refer as shortest path (SP) interpolation. This method ignores the nonlinear topology of the data manifold in the latent space and often results in poor interpolated images, which are generated faraway from the data manifold.

To achieve better interpolation results, we propose to formulate the traversal task as an optimization problem which aims to find an optimal path that is shortest while remaining close to the data manifold. We refer to our interpolation scheme as

shortest likelihood path (SLP) interpolation and design an optimization objective as follows:

$$\mathcal{O}_{\text{SLP}} = \underbrace{L_{\text{path}}(\mathbf{s})}_{\text{path length}} + \underbrace{\text{std}_{\text{step length}}(\mathbf{s})}_{\text{equal step length}} - \underbrace{\log p(\mathbf{t} = \mathbf{s})}_{\text{path likelihood}}, \quad (5.11)$$

where $\mathbf{s} = [s_1, \dots, s_J]^T$ denotes a list of J steps along the traversal path and each s_j denotes a single encoding that can be projected back to the data space as an interpolated image. As can be seen, our $\mathcal{O}_{\text{SLP}}$ objective contains three terms: 1) $L_{\text{path}}(\mathbf{s})$ minimizes the current path length, 2) $\text{std}_{\text{step length}}(\mathbf{s})$ requires steps to be evenly distributed along the path and 3) $\log p(\mathbf{t} = \mathbf{s})$ ensures all the interpolated encodings to be generated from the inferred data distribution $p(\mathbf{t})$. To find an optimal path $\mathbf{s}$, we minimize $\mathcal{O}_{\text{SLP}}$ wrt $\mathbf{s}$ by a standard optimization scheme, such as AdamOptimizer [98] that is used in our experiments. Notice that our SLP interpolation can be generally applied to any representation learning algorithm, as long as the likelihood of the latent data distribution $p(\mathbf{z})$ can be easily evaluated.

In Section 5.2.5, we formulate the latent space interpolation between a pair of images as a shortest likely path (SLP) optimization task, which favours the paths that go through high density regions of the inferred latent data distribution $p(\mathbf{z})$ or $p(\mathbf{t})$. Here we demonstrate our SLP method informed by the learnt latent distribution outperforms the conventional linear shortest path (SP) interpolation.

Figure 5.7 illustrates the interpolation between 3 pairs of MNIST images through the data manifolds estimated by our method, a GMM prior and a normal prior. Notice that our LaDDer model can produce the same quality samples from a much lower dimension latent space compared to the VAE models with the GMM and the normal prior (2D vs 16D). The super low latent dimension allows us to visualize the learnt data distribution $p(\mathbf{t})$ and the traversed path in Figure 5.7b-c. Our SLP interpolation results in paths that only step on regions with high likelihood of $p(\mathbf{t})$, whereas SP interpolation ignores the topology of $p(\mathbf{t})$ and often lands on the low density regions which do not correspond to realistic data samples, hence producing unrealistic samples (bottom rows in Figure 5.7d-f). Notice that neither SLP nor SP gives a good interpolation for the normal prior model. This is because the normal prior poorly represents the latent data distribution and even if an encoding has a high likelihood wrt the normal prior, it does not correspond to a realistic data sample. This reinforces the importance of obtaining an accurate modelling of the inferred data distribution in being able to utilize it for later tasks.

Figure 5.8 illustrates the interpolation between 2 pairs of CelebA images over

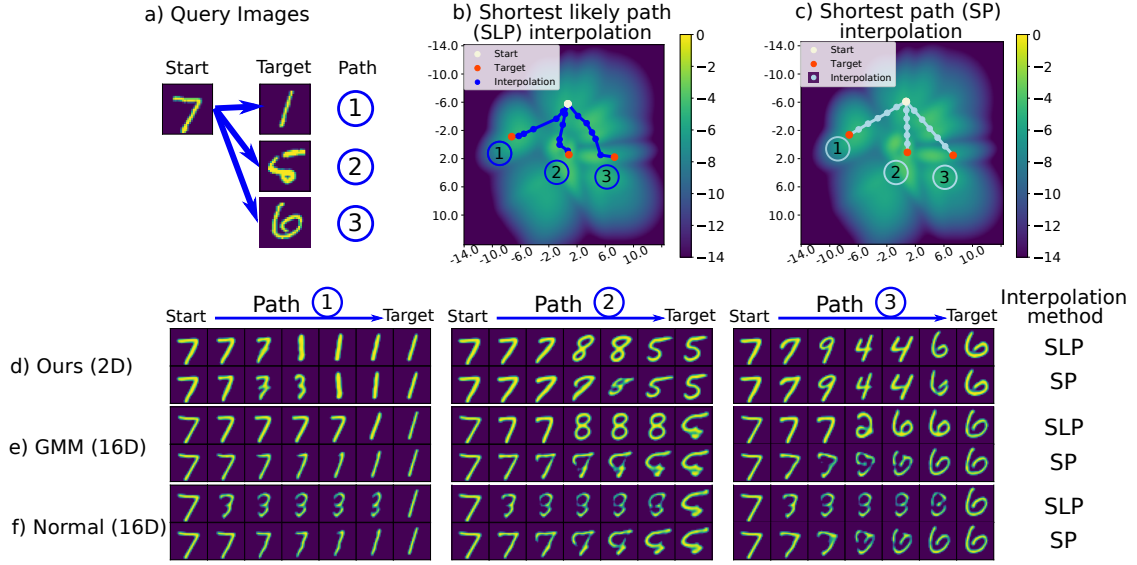


Figure 5.7: Latent space interpolation for MNIST. (a) The 3 pairs of MNIST images for the interpolation task. (b-c) The traversed paths produced by shortest likely path (SLP) and shortest path (SP) are visualized in our very low-dimensional t -space. (d-f) The interpolated images for both interpolation methods using the data distribution estimated by our method, a GMM prior and a normal prior.

the data manifold produced by our method, where the SLP optimization takes place in the 32D latent space of the prior VAE. Here, we plot the different objectives in our SLP optimization (blue line) for each example in (c-f). The overall objective ($\mathcal{O}_{\text{SLP}}$ in Equation (5.11)) smoothly converges in all examples. It is clear that our SLP solution trade-offs path length and allows different step lengths for obtaining high likelihood over the traversed path. As a result, our SLP method only produces realistic images which smoothly transform into the target face, whereas the SP method produces unrealistic faces along the traversed path.

5.3 Summary

In this chapter, we have introduced two effective approaches to mitigating the over-regularization issue in the VAE ELBO objective. Firstly, we show that the variance parameter in the Gaussian decoder of a VAE model can be optimized under the ELBO objective and can serve as an automatic weighting factor that controls the level of regularization in the ELBO. We also extend this single variance parameter to a set of variance parameters that can vary across different input samples and across different data dimensions. We show that such modelling can lead to a significant improvement in the likelihood score. The estimated variance map gives a good indication of where the reconstruction error is high and, hence, can be used to

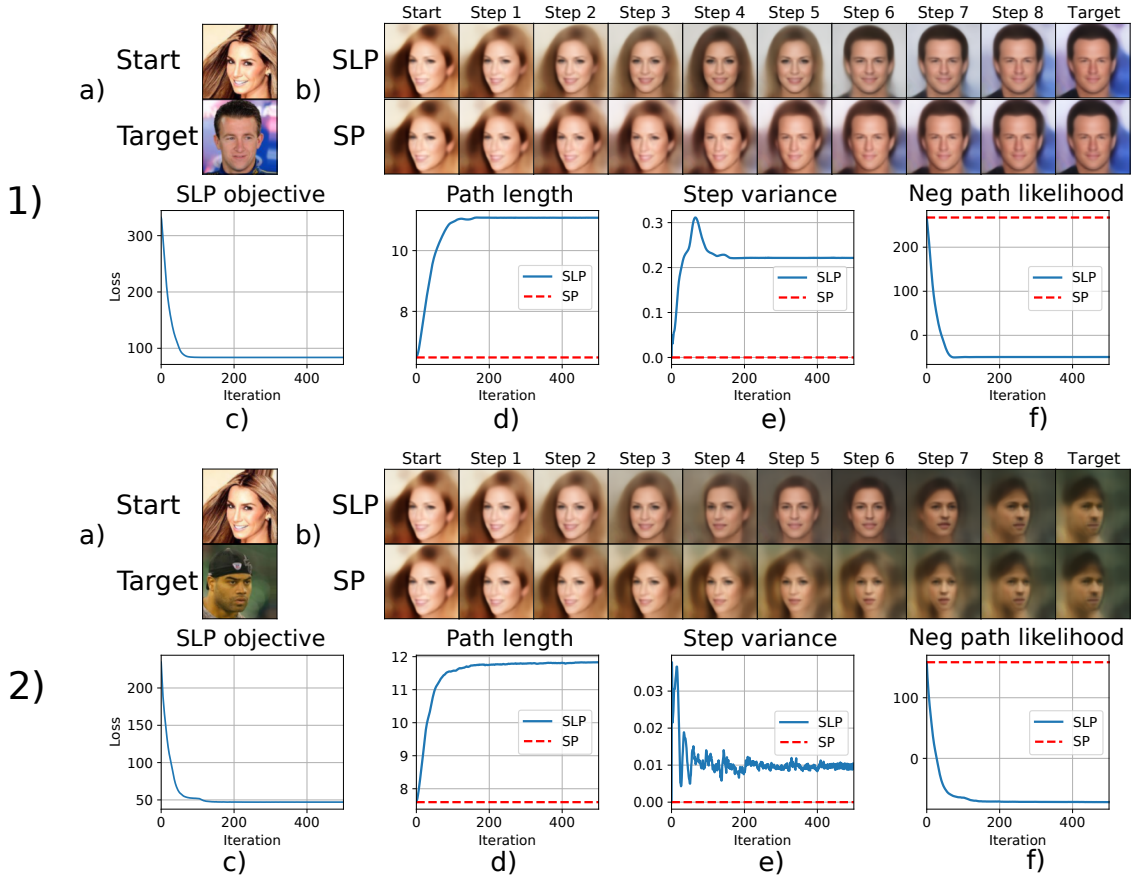


Figure 5.8: Latent space interpolation for CelebA. (a) The pair of images to be interpolated. (b) The sequence of interpolated images along the data manifold estimated by our LaDDer model, using two different interpolation methods. (c-f) optimization of different objectives in the our SLP objective. More examples are given in Appendix B.3.2.4.

indicate the areas of uncertainty during reconstruction and generation.

Secondly, we adopt a flexible prior to facilitate the accurate modelling of the inferred latent data distribution. To achieve this, we propose LaDDer, which consists of multiple VAE models each acting on its predecessor’s latent representation and a non-parametric mixture as the hyper prior for the innermost VAE. From extensive experiments, we show that LaDDer is able to accurately model latent data distribution of complex data. We also demonstrate how to use the derived latent distribution to facilitate further tasks, such as producing better interpolation along the derived data manifold.

In the next chapter, we address a different challenge in the VAE training, where the inferred auto-encoding function causes significant drifts between the posterior distribution of the original sample and the posterior distribution of the reconstructed sample. We believe such drifts indicate overly severe distortion of the information

in the input sample under the auto-encoding mapping. We will present our solution to this challenge.

6

Mitigating the encoding drift in VAEs

As introduced in Section 4.2, we show that an inconsistency exists in between the inference (or encoding) and the generative (or decoding) model in a VAE. This inconsistency results in an auto-encoding function that slightly alters the information in the input sample. Although the change over a single auto-encoding transformation may be small, we see that a data sample can be converted to a shape that is completely different over a few iterations of the auto-encoding mapping.

We aim to address this problem in this chapter. We propose to augment the ELBO objective with an additional consistency loss. This consistency loss is designed to mitigate any differences between the posterior distribution of the input sample and the posterior distribution of the reconstructed sample. This new learning objective can effectively reduce the level of information drift caused by the inferred auto-encoding function.

The relevant content in this chapter is published as a student abstract paper - Towards Consistent Variational Auto-Encoding at *Proceedings of the AAAI Conference on Artificial Intelligence (Student Abstract)* 2020.

6.1 Introducing a consistency loss

Recall that the VAE's ELBO objective is defined in Equation (4.1). We can visualize different loss terms in the ELBO objective in a plot as shown in Figure 6.1. Firstly, we have ① $\mathbb{E}_{q_\phi(\mathbf{z}_i|\mathbf{x}_i)}[\log p_\theta(\mathbf{x}'_i|\mathbf{z}_i)]$ a reconstruction likelihood term that requires the reconstructed data sample $\mathbf{x}'_i$ to preserve as much information as possible from the original data sample $\mathbf{x}_i$. Secondly, we have ② $D_{\text{KL}}(q_\phi(\mathbf{z}_i|\mathbf{x}_i) \parallel p(\mathbf{z}))$ a prior constraint term that encourages the posterior distribution $q_\phi(\mathbf{z}_i|\mathbf{x}_i)$ to remain close to a prior distribution $p(\mathbf{z})$. Both these terms are indicated by the black dashed lines in Figure 6.1.

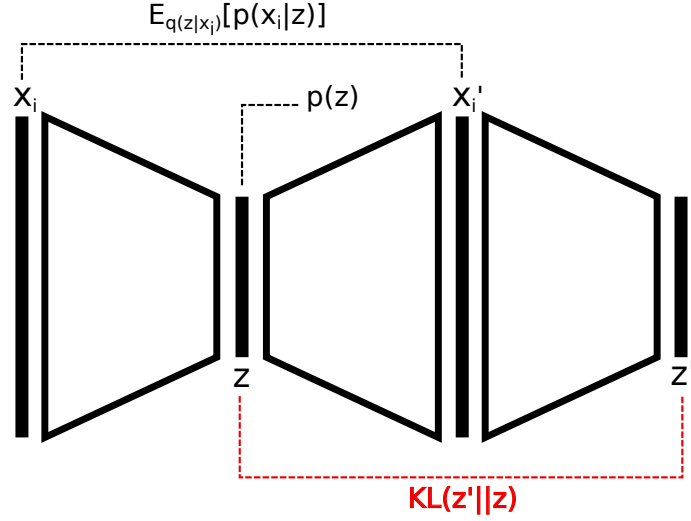


Figure 6.1: An illustration of our matching loss. We propose an additional matching loss $D_{KL}(z'_i||z_i)$ (highlighted in red) that encourages the posterior of a reconstructed sample to remain the same as the posterior of the original input sample.

As we have discovered previously, inconsistency between the inference and the generative model occurs when we encode the reconstructed data sample again with the VAE encoder. To examine the level of such inconsistency, we add an additional encoding step to the original VAE model as shown in the last encoding box on the right in Figure 6.1. This gives us the second latent code variable z'_i , which follows the posterior distribution $q_\phi(z'_i|x'_i)$ of the reconstructed sample x'_i . Now we are able to measure the amount of the encoding drift by computing the distance between the two posterior distributions, i.e. $D_{KL}(z'_i||z_i)$. If the inference model matches the generative model perfectly, then the two posterior distributions would be the same and $D_{KL}(z'_i||z_i) = 0$. On the other hand, if inconsistency exists (or the encoding drift occurs), then $D_{KL}(z'_i||z_i)$ would be positive. As $q_\phi(z'_i|x'_i)$ drifts further away from $q_\phi(z_i|x_i)$, $D_{KL}(z'_i||z_i)$ would be larger.

In order to mitigate the encoding drift (or the inconsistency between the inference and the generative model), we propose to include this inconsistency term in the ELBO objective, as follow:

$$\mathcal{L}(x_i; \phi, \theta) = \text{ELBO}_i(\phi, \theta) - \alpha \cdot D_{KL}(z'|z), \quad (6.1)$$

where $\alpha \geq 0$ is an adjustable weight that controls the strength of the consistency constraint. As $D_{KL}(z'||z)$ is non-negative, our objective is effectively a lower bound to the original ELBO objective and the bound is tight when $D_{KL}(z'||z) = 0$, i.e. when the inference and the generative models perfectly match. This means that even with this additional term, our consistency-aware objective $\mathcal{L}(x_i; \phi, \theta)$ can still

approach to the data log likelihood if we can discover a pair of the inference and the generative model that are effectively the inverse of each other.

There are several benefits of our proposal. Firstly, we can directly measure the degree of inconsistency between the inference and the generative models through this consistency term $D_{\text{KL}}(\mathbf{z}'||\mathbf{z})$. Secondly, our solution introduces no additional parameters to the original VAE model. Thirdly, our proposed objective is in fact a lower bound to the original ELBO loss. At the optimum of the original VAE's formulation, i.e. when $D_{\text{KL}}(q_\phi(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = 0$, we expect the consistency term $D_{\text{KL}}(\mathbf{z}'||\mathbf{z})$ will also approach to zero. Therefore, we will approach the true data log likelihood as the bound gets tighter.

6.2 Compute the consistency term $D_{\text{KL}}(\mathbf{z}'||\mathbf{z})$

$D_{\text{KL}}(\mathbf{z}'||\mathbf{z})$ is the Kullback-Leibler (KL) divergence between the posterior distribution of the reconstructed and the original data point respectively, i.e. $q_\phi(\mathbf{z}'|\mathbf{x}'_i)$ and $q_\phi(\mathbf{z}|\mathbf{x}_i)$. $D_{\text{KL}}(\mathbf{z}'||\mathbf{z})$ can be used to quantitatively measure the amount of mismatch between the current inference and the generative model, as this quantity is proportional to the information drift when the same observation is passed through the inference and the generative model consecutively.

Here, we show how to evaluate this consistency term. Firstly, as we show in Appendix A.4, the KL divergence between two Gaussian distributions can be analytically computed. Following the common assumption that the posterior distribution in the VAE model is a diagonal Gaussian, i.e. $q_\phi(\mathbf{z}_k|\mathbf{x}_i) = \mathcal{N}(\mu_1^k, (\sigma_1^k)^2)$ and $q_\phi(\mathbf{z}'_k|\mathbf{x}'_i) = \mathcal{N}(\mu_2^k, (\sigma_2^k)^2)$, where k denotes the k -th dimension of the latent variable $\mathbf{z}_i$ and the dimensionality of the latent space is d_z , we can evaluate $D_{\text{KL}}(\mathbf{z}'||\mathbf{z})$ as the following form:

$$D_{\text{KL}}(\mathbf{z}'||\mathbf{z}) = \frac{1}{2} \sum_{k=0}^{d_z} \frac{(\sigma_1^k)^2}{(\sigma_2^k)^2} + \frac{1}{2} \sum_{k=0}^{d_z} \frac{(\mu_2^k - \mu_1^k)^2}{(\sigma_2^k)^2} + \sum_{k=0}^{d_z} \log \sigma_2^k - \sum_{k=0}^{d_z} \log \sigma_1^k - d_z. \quad (6.2)$$

Secondly, if the posterior distribution takes more complex form other than a Gaussian distribution, an analytic computation might not be available. In this case, we can compute $D_{\text{KL}}(\mathbf{z}'||\mathbf{z})$ through Monte Carlo sampling, i.e.

$$D_{\text{KL}}(\mathbf{z}'||\mathbf{z}) \approx \mathbb{E}_{q_\phi(\mathbf{z}'|\mathbf{x}'_i)}[\log q_\phi(\mathbf{z}'|\mathbf{x}'_i) - \log q_\phi(\mathbf{z}|\mathbf{x}_i)]. \quad (6.3)$$

To do so, we draw L samples from the reconstructed posterior distribution $\mathbf{z}'_l \sim q_\phi(\mathbf{z}'|\mathbf{x}'_i)$ for $l = 1, \dots, L$. We then evaluate the log likelihood of these L samples with

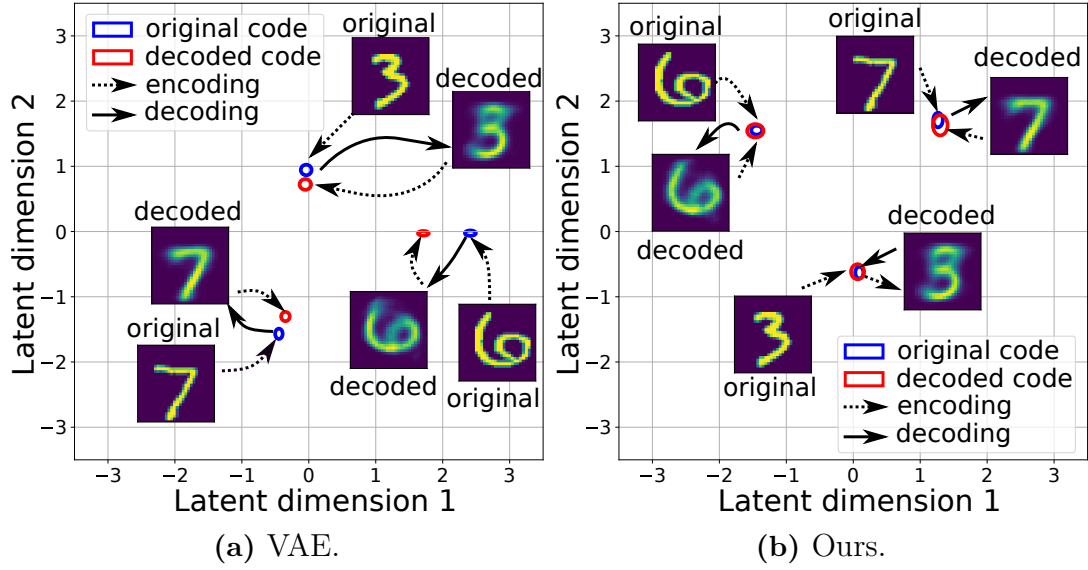


Figure 6.2: Our matching loss has effectively reduced the drift between the posterior distribution of an input sample and the posterior distribution of the reconstructed sample. Here we show the posterior distributions for three data samples. We denote the posterior distribution for the original sample as a blue circle and the posterior distribution for the reconstructed sample as a red circle. (a) The change of the posterior distributions for a VAE trained under ELBO loss. (B) The change of the posterior distributions for a VAE under our modified ELBO loss. Our method successfully reduces the drift in the posterior distribution.

respect to both posteriors $\log q_\phi(\mathbf{z}'|\mathbf{x}_i')$ and $\log q_\phi(\mathbf{z}|\mathbf{x}_i)$. Finally, we take the average of these L values and that gives us an approximation of the $D_{\text{KL}}(\mathbf{z}'||\mathbf{z})$. Based on our experiment, $L = 20$ is a sensible value that gives fairly accurate approximation which is very close to the value computed from the analytical solution in Equation (6.2).

6.3 Reducing drift in the posterior distributions

We have observed that training a VAE model under our proposed learning objective as introduced in Equation (6.1) can successfully reduced the inconsistency between the inference and the generative models. To demonstrate this, we visualize the posterior distribution of an input sample and the posterior distribution of the reconstructed image of the same sample in Figure 6.2. We show the results for a VAE trained under the ELBO objective in Figure 6.2(a) and for a VAE model trained with our learning objective in Figure 6.2(b). We can clearly see that the drifts that occur in sub-figure (a) have been significantly reduced in sub-figure (a), thanks to the consistency loss term in our learning objective.

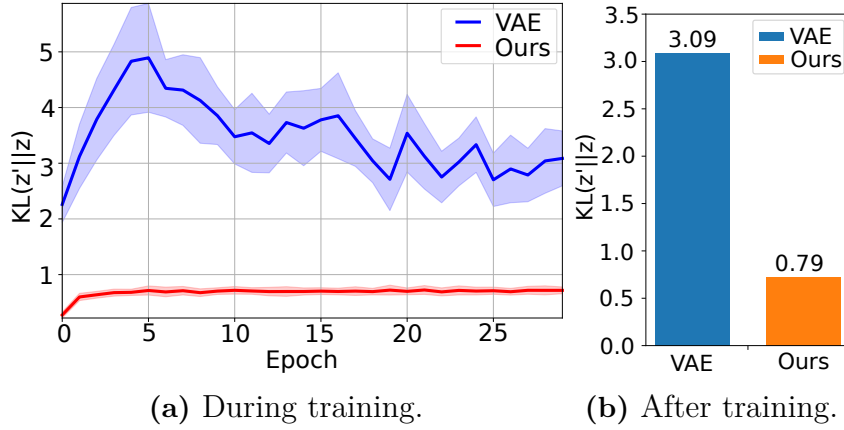


Figure 6.3: $D_{KL}(z'|z)$ during and after training on a held-out test set for MNIST dataset (lower is better).

We also monitor the inconsistency loss $D_{KL}(z'|z)$ between the two posteriors $q_\phi(z|x_i)$ and $q_\phi(z'|x'_i)$ during the training phase, as shown in Figure 6.3a, and after training on a test set of 10k images, as shown in Figure 6.3b. We can clearly see that the two posterior distributions start to drift apart during training without the supervision of the consistency constraint $D_{KL}(z'|z)$. In contrast, the difference between the two posterior distributions for a VAE model under our learning objective with the additional consistency constraint remains low. As a result, our method manages to reduce the inconsistency between the inference and the generative models measured by $D_{KL}(z'|z)$ to less than a third of the level of a VAE model trained under ELBO objective.

6.4 Impact on other metrics

Other than reducing the inconsistency between the inference and the generative model in a VAE, we are also interested in the impacts the additional consistency constraint term $D_{KL}(z'|z)$ has on the other aspects of a VAE model. In this section, we investigate the performance of our proposed consistency-aware learning objective on a list of metrics, including the negative ELBO loss (lower is better), the reconstruction loss (lower is better), the prior compliance loss $D_{KL}(q_\phi(z|x_i)||p(z))$ (lower is better), and our proposed consistency loss (lower is better) on a held-out test set over training for both MNIST (Figure 6.4a) and Fashion MNIST (Figure 6.4b) dataset. For both datasets, we can clearly see that our method reaches much lower consistency loss compared to the original VAE algorithm. As for other metrics, our method achieves a lower negative ELBO loss and a lower reconstruction loss but a worse prior compliance loss in the Fashion MNIST dataset. In the MNIST dataset,

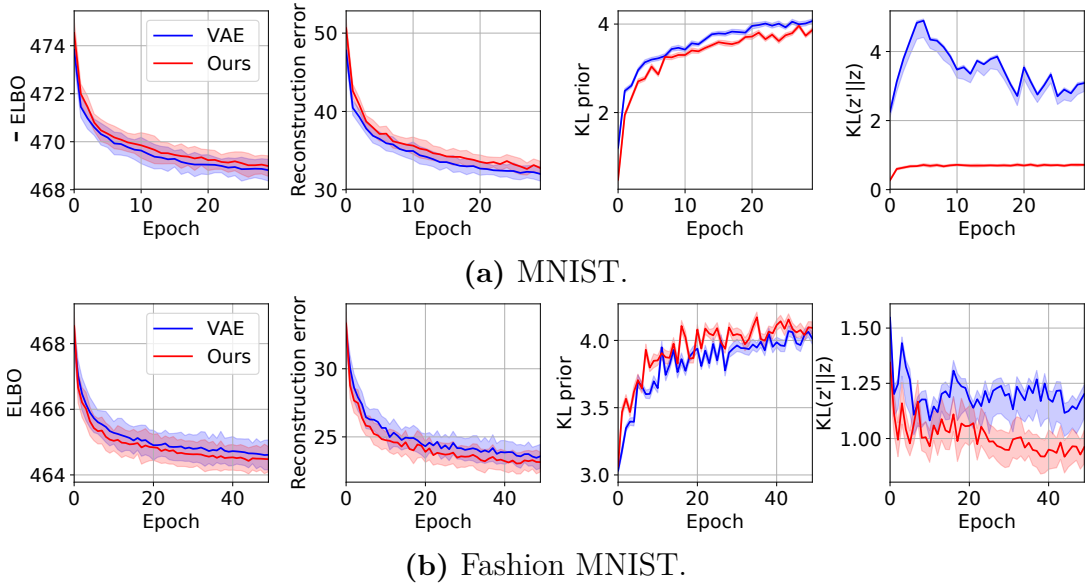


Figure 6.4: Monitoring four metrics, including negative ELBO (lower is better), reconstruction error (lower is better), $D_{\text{KL}}(q_{\phi}(\mathbf{z}|\mathbf{x}_i)||p(\mathbf{z}))$ (lower is better) and $D_{\text{KL}}(\mathbf{z}'||\mathbf{z})$ (lower is better), over training on MNIST and fashion MNIST dataset. Blue lines indicate a VAE trained under ELBO loss and red lines indicate a VAE trained under our loss. Our method consistently gives a more consistent posterior (i.e. low $D_{\text{KL}}(\mathbf{z}'||\mathbf{z})$). However, our method does not give a consistent improvement on the other metrics.

Weight α	0 (VAE)	10^{-3}	10^{-2}	10^{-1}	1
$D_{\text{KL}}(\mathbf{z}' \mathbf{z})$	3.09	1.75	1.07	0.82	0.71

Table 6.1: The impact of the weight α on the consistency loss $D_{\text{KL}}(\mathbf{z}'||\mathbf{z})$ evaluated on a held-out test set of 10k samples for MNIST dataset. A larger α leads to smaller consistency loss $D_{\text{KL}}(\mathbf{z}'||\mathbf{z})$, i.e. a smaller drift in between the encoding of the reconstructed sample and the encoding of the original sample.

we see an opposite story that the original VAE ELBO objective gives a lower negative ELBO loss and a lower reconstruction loss but a worse prior compliance loss. However, the amount of impacts on the three metrics - the negative ELBO loss, the reconstruction loss and the prior compliance loss - are rather marginal in both cases. Therefore, we reckon that optimizing our proposed consistency loss might have a small impact on the reconstruction task and/or on the prior compliance task, but the impact is limited, given that the consistency loss is properly applied (we will discuss this point in the next section).

6.5 The weight parameter α

As shown in Equation (6.1), the significance of the consistency constraint $D_{\text{KL}}(\mathbf{z}'||\mathbf{z})$ on the original ELBO loss is controlled by a weight α . When $\alpha = 0$, the consistency loss $D_{\text{KL}}(\mathbf{z}'||\mathbf{z})$ effectively does not take any effect, the objective function simply reduces to the original ELBO loss. As α increases, the effect of the consistency constraint enhances, i.e. the model will be optimized to prioritize the consistency constraint than the other two learning tasks - the auto-encoding (or reconstruction) task and the prior compliance task. We can see this increasing level of impact as the weight parameter increases on a held-out validation set for trained models under different α values, as shown in Table 6.1. Clearly, the consistency loss is lowest when we allow the consistency constraint to be on at full strength. However, this value of α is not an ideal choice, as the VAE model is likely to fail on the other two tasks, such as failing to generate reasonable quality reconstruction.

Therefore, to get the training started, we need to choose a sensible value of α that does a good balance among different learning tasks in a VAE model. To do so, we carry out a grid search on different values of α and we choose the α that causes the minimal compromise on the reconstruction error and the prior compliance loss while keeping the consistency loss low.

6.6 An interesting observation

We notice in Figure 4.3 that the auto-encoding function tends to transform different input samples to the same pattern over sufficient iterations of auto-encoding. We are interested in finding out whether this applies to all the data samples. If so, how many of such patterns are there and what do these patterns look like? Furthermore, what impacts does the additional consistency loss have on these patterns?

To find out the answers to the above questions, we apply repeated reconstruction over 500 iterations to a set of 250 test samples and we visualize the changes in the mean values of these samples' posterior distributions as the blue trajectories in Figure 6.5. The posterior distributions from a VAE trained with the ELBO objective is shown in sub-figure (a) and the posterior distributions from a VAE trained with our objective that includes the consistency loss is shown in sub-figure (b). We indicate the mean value at the last reconstruction for each trajectory with a red dot. It is interesting to see that the posterior distributions of the 250 data samples converge to only 5 distinctive locations for the VAE trained with the ELBO objective. The data samples that correspond to these 5 convergence features are shown in the

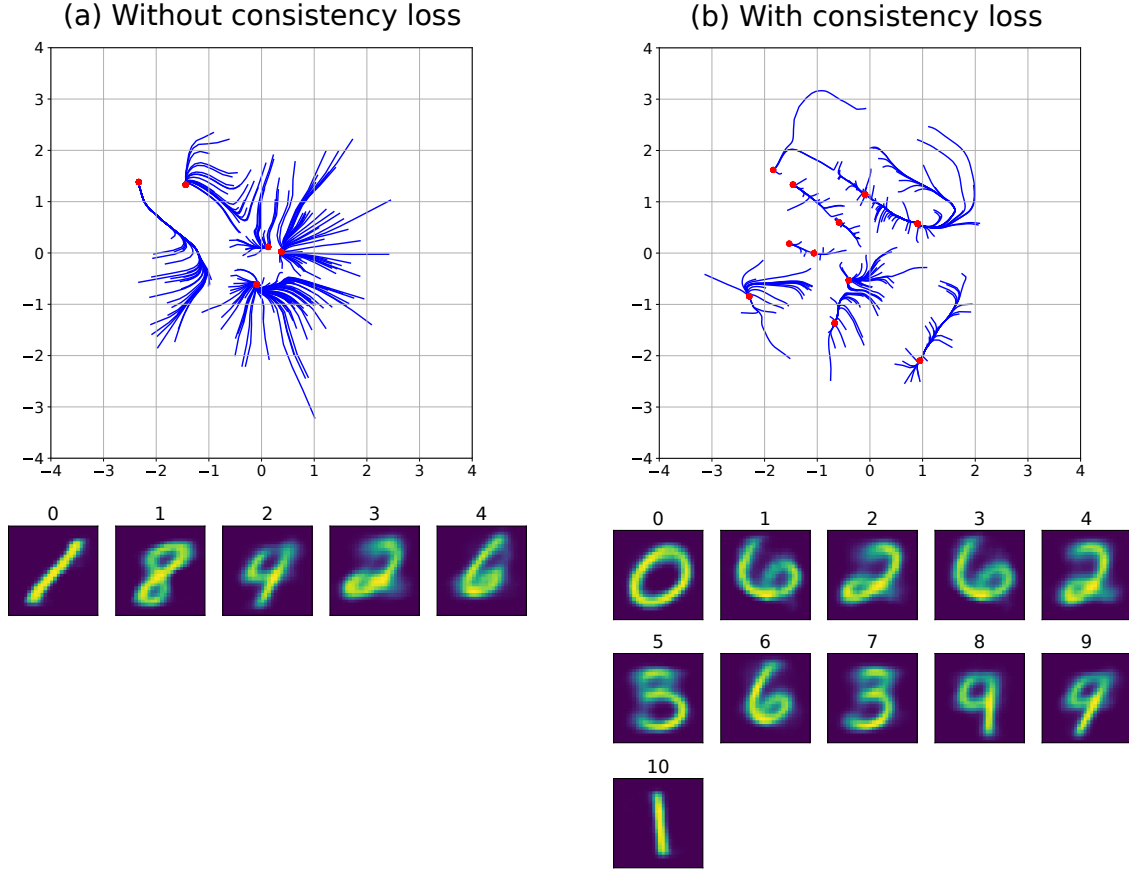


Figure 6.5: Different input samples converge to a fixed set of points over iterations of auto-encoding mapping. Here we apply repeated reconstruction over 500 iterations to a set of 250 test samples. The blue trajectories denote the changes of posterior distribution’s mean values and the red points denote the locations where all posterior distributions from different samples converge to. We also visualise the corresponding data images of these red points at the bottom.

images below the sub-figure. With the consistency loss, we see the data samples converge to a larger set of features - 11 distinctive locations to be precise.

To interpret these distinctive locations, we describe the repeated auto-encoding setting as a dynamical system that is governed by the following ordinary differential equation:

$$\dot{\mathbf{x}} = f(\mathbf{x}), \quad (6.4)$$

where $\mathbf{x}$ denotes an input sample and $f(\mathbf{x})$ denotes the derivative of the auto-encoding function with respect to the data variable [99]. From our investigation, VAEs tend to naturally behave as a stable dynamical system, as all data samples converge to a finite set of values. These final set of values can be referred to as the fixed points [100] of this auto-encoding dynamical system and they are reached

when

$$\dot{\mathbf{x}} = 0, \tag{6.5}$$

i.e. an input sample at this location will remain unchanged under the auto-encoding mapping. Exactly how to interpret these fixed points in the contexts of VAEs and whether they have an impact on the reconstruction quality and on how well the inferred VAE can fit the data remain unclear to us. We leave these questions as potential future work.

6.7 Summary

In this chapter, we address the challenge of *an information drift* that occurs when an input sample is encoded to its posterior distribution with an inaccurate inference model (or encoding function) from a VAE. We visualize this drift behaviour by auto-encoding a data sample multiple times and we notice that auto-encoding function gradually transforms an input sample to a shape that can be quite different. We propose to mitigate this effect by adding a consistency loss to the ELBO objective. This consistency loss does not require any additional parameters and can be conveniently added to any variants of the VAE models. The new objective is a lower bound of the ELBO, hence remaining an effective lower bound to the data log likelihood. We observe that the introduction of this consistency loss effectively reduces the drifting effects without causing any damage to other metrics, such as ELBO and reconstruction error.

In the next two chapters, we will explore the possibilities of applying VAEs to challenging real world applications, such as anomaly detection for time series and uncertainty estimation for classification tasks.

7

Anomaly Detection for Time Series

In this chapter, we propose a hybrid anomaly detection method that combines the representation learning power of a variational autoencoder (VAE) with the temporal modelling ability of a long short-term memory RNN (LSTM). Via the VAE module our model aims to capture the structural regularities of the time series over local windows, while the LSTM module attempts to model the longer term trend. Both the VAE and LSTM units do not require labelled anomalies for training.

The relevant content in this chapter is published as a conference paper - Anomaly Detection for Time Series Using VAE-LSTM Hybrid Model at *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* 2020.

7.1 Background and related work

Anomaly detection for time series is concerned with detecting unexpected system behaviours across time to provide informative insights. In many industrial applications, anomaly detection is used for monitoring sensor failures, alerting users of external attacks and detecting potential catastrophic events at an early stage [101]. Despite all the benefits, designing a good anomaly detection algorithm is extremely challenging. This is because the training data are often unbalanced with very few labelled anomalies. Furthermore, most anomalous behaviour types are not known a priori and a good anomaly algorithm is expected to be able to detect even unseen anomalies. Due to these constraints, anomaly detection algorithms often have to be trained in an unsupervised fashion.

Broadly, we may characterize three types of anomalies that commonly occur in time series: namely point, context and collective anomalies [102]. Of the three types, *point anomalies* are the easiest to detect as data points can be treated independently during detection and no temporal relationships need to be considered. For this

reason, simple threshold approaches or multi-layer perceptron (MLP) based methods [103] work relatively well for point anomalies. *Context and collective anomalies*, on the contrary, are more challenging. Context anomalies depend on the value of surrounding data points and thus local information is required for their detection and convolutional neural networks (CNNs) with larger receptive fields have been shown to work well in this case [104]. *Collective anomalies* occur when a series of data points together exhibits abnormal behaviours. As collective anomalies always occur in sequence over a reasonably long period, recurrent neural networks (RNNs), have been shown to be effective [105].

Due to the importance of the anomaly detection task, a number of approaches have been proposed. VAEs are extremely suited for modelling the normal behaviours in an anomaly detection task, as they can infer a generative model that fits with the majority of the training data without any labels of the abnormal behaviours. As a result, VAEs have been used for anomaly detection in various works with promising results [32], [106]–[108]. However, anomaly detection algorithms based only on VAEs often fail in detecting long-term anomalies, as VAE models cannot analyse information beyond a short local window. Our approach overcomes this limitation by using VAEs as local feature extractor and coupling it with a LSTM module to take care of long-term trends.

Another line of research [105], [109] explores the idea of detecting anomalies using recurrent neural networks (RNNs). RNNs [110] are a stateful variant of feed-forward neural networks that propagate a hidden state from one time step to the next. Because of their stateful nature, RNNs are particularly suited for processing time series data. In this paper we use the LSTM [37], [38] variant of RNNs which endow the network with an explicit memory "cell" in addition to the hidden state found in a standard RNN. Our method differs from those approaches in the way that our LSTM module is not applied to raw samples but to embeddings that represent a local window. Such setup makes our algorithm capable of ignoring redundant raw samples and tracking events over longer terms.

Hybrid models are a common approach for video analysis where a representation learning module is used to distil spatial information in a single image frame and a sequential module is applied to model the temporal correlation across a series of frames [111], [112]. [113] applies such a hybrid model to detect rare events in a surveillance video clip. The major difference between our approach and the hybrid models in the video applications is that the representation learning module for videos is applied to an image, i.e. a data point at a single time stamp, whereas our representation learning module for time series processes a sequence of data points

over a short period of time to form a local temporal basis for the latter sequential module to build upon. This difference allows our method to create a hierarchical structure for the given time series and analyse it over multiple time scales.

7.2 Our method

Given a time series $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$, where $\mathbf{x}_i \in R^m$ is a m -dimensional reading at the i -th time stamp that contains information about m different channels, we formulate the anomaly detection task as follows. At time t ($L \leq t \leq N$), we are allowed to use a sequence of L past readings, i.e. $\mathbf{S}_t = [\mathbf{x}_{t-L+1}, \dots, \mathbf{x}_t]$, to predict a binary output $y_t \in \{0, 1\}$ with 1 indicating an anomaly has occurred in $\mathbf{S}_t$. Such formulation allows our algorithm for *online detection*. Figure 7.1 gives an overview of our detection algorithm, which consists of a VAE module for extracting local features of a short window and a LSTM module for estimating the long-term trends. In this section, we will first introduce how these two modules are trained in an unsupervised manner and then explain how our algorithm is used for anomaly detection.

7.2.1 Training VAE-LSTM Models

To train our VAE-LSTM models in an unsupervised manner, we first need to separate a training and a test set from the given time series. An example of training-test set separation is given in Figure 7.2, where we take a continuous segment of the given time series which contains *no anomalies* as the training data and the rest of the time series with anomalies is kept as the test data for evaluation.

The VAE model consists of an encoder and a decoder. It takes a local window of p consecutive readings as input, estimates a low-dimensional embedding of q dimensions through the encoder and reconstructs the original window through the decoder. To train the VAE model, we generate rolling windows from the training data. For example, $\mathbf{w}_t = [\mathbf{x}_{t-p+1}, \dots, \mathbf{x}_t]$ denotes the window ending at time t . The LSTM model operates on the VAE embeddings of a sequence of k non-overlapping windows. We use $\mathbf{W}_t = [\mathbf{w}_{t-(k-1) \times p}, \mathbf{w}_{t-(k-2) \times p}, \dots, \mathbf{w}_t]$ to denote the sequence of windows ending at time t and $\mathbf{E}_t = [\mathbf{e}_t^1, \dots, \mathbf{e}_t^k]$ to denote the corresponding embeddings in $\mathbf{W}_t$ with $\mathbf{e}_t^i$ indicating the embedding of the i -th window in $\mathbf{W}_t$. For a training data of N_{train} readings, we can generate $N_{\text{train}} - p$ rolling windows for training the VAE model and $N_{\text{train}} - pk$ rolling sequences for training the LSTM model. We reserve a randomly drawn 10% of the generated sequences from the training data as a validation set and all the windows and sequences in the validation

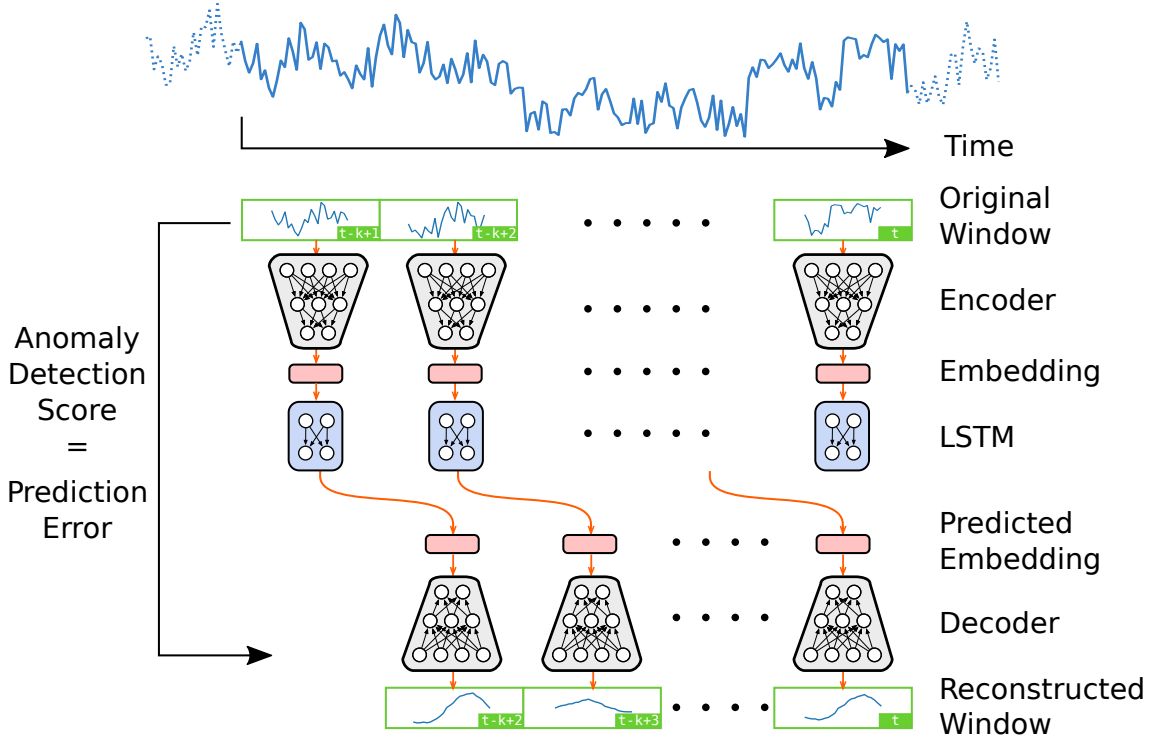


Figure 7.1: System overview. Our VAE-LSTM model detects anomalies over a sequence of k consecutive windows of a given time series. i -th window $\mathbf{w}_i$ is encoded into a low-dimensional embedding $\mathbf{e}_i$, which is fed into a LSTM model to predict the next window's embedding $\hat{\mathbf{e}}_{i+1}$. The predicted embedding is then decoded to reconstruct the original window $\hat{\mathbf{w}}_{i+1}$. The reconstruction error serves as our anomaly detection score.

set are excluded from training.

With the remaining windows in the training set, we optimize the VAE model parameters to maximize the ELBO loss defined in Section 3. To ensure the derived embeddings are informative to the raw data, we follow the procedure proposed in Chapter 5.1 to avoid over-regularization on the auto-encoding function. After the VAE model has been optimized, we use the encoder from the trained VAE model to estimate all the embedding sequences $\mathbf{E}_t$ in the training set. To train the LSTM model, we have the LSTM model take the first $k - 1$ embeddings in a sequence $\mathbf{E}_t$ and predict the next $k - 1$ embeddings, i.e.

$$[\hat{\mathbf{e}}_t^2, \dots, \hat{\mathbf{e}}_t^k] = \text{LSTM}([\mathbf{e}_t^1, \dots, \mathbf{e}_t^{k-1}]) \quad (7.1)$$

We optimize the LSTM model parameters by minimizing the prediction error of the final embedding, i.e. $\min \|\hat{\mathbf{e}}_t^k - \mathbf{e}_t^k\|$. Notice that all the model parameters for both VAE and LSTM units are optimized without anomaly labels.

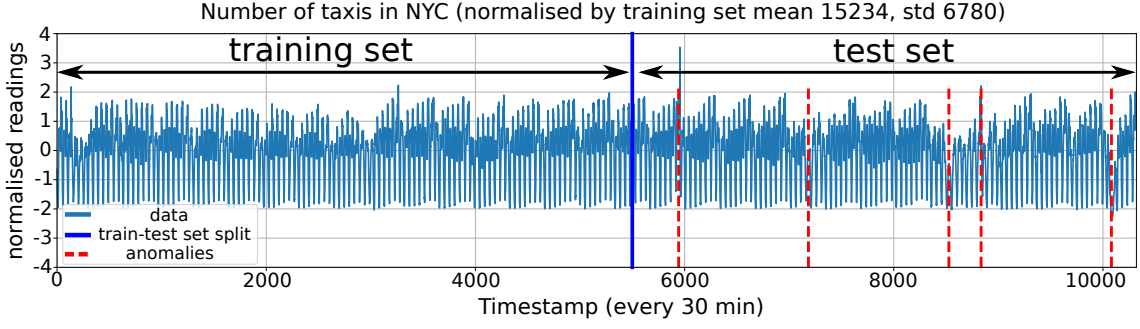


Figure 7.2: Training and test set separation. Here we give an example on how to obtain the training and test set from a time series that contains an anomalous events. The time series (blue line) records the volume of taxi requests in New York City. The red dashed lines indicate the anomalous events. We take the section that does not contain any anomalous events (left side of the solid blue line) as the training set and reserve the sections that contain anomalous events as test set for evaluation.

7.2.2 Anomaly Detection using the VAE-LSTM Model

After training, our VAE-LSTM model can be used for anomaly detection in real time. At time t , the VAE-LSTM model analyses a test sequence $\mathbf{W}_t$ that contains $k \times p$ past readings tracing back from t . Our model first uses the encoder from the VAE to estimate the sequence of embeddings $\mathbf{E}_t$ in $\mathbf{W}_t$. Then it feeds the first $k - 1$ embeddings to the LSTM model to predict the next $k - 1$ embeddings $[\hat{e}_t^2, \dots, \hat{e}_t^k]$, as given in Equation (7.1). Finally, our model reconstructs the last $k - 1$ windows using the predicted embeddings and the VAE decoder, i.e.

$$\hat{\mathbf{w}}_{t-(k-i) \times p} = \text{Decoder}(\hat{e}_t^i), \text{ for } i = 2, \dots, k. \quad (7.2)$$

With the reconstructed windows, we can define a score function d_t to flag anomalous behaviours by summing up the prediction errors for $\mathbf{W}_t$, i.e.

$$d_t = \sum_{i=2}^k \|\hat{\mathbf{w}}_{t-(k-i) \times p} - \mathbf{w}_{t-(k-i) \times p}\|_2. \quad (7.3)$$

To detect an anomaly, we also need to define a threshold θ on the score function d_t , above which we will flag an anomaly alert $y_t = 1$ at the current t . The corresponding sequence $\mathbf{W}_t$ will mark a suspicious window where the anomaly event might have occurred. When there is sufficient data, we should determine θ using a validation set that contains both normal and anomalous examples. The threshold that gives the best performance, such as F1 score or other metrics, on this validation set is the detection threshold for the given time series. When there is limited

Table 7.1: Evaluating our VAE-LSTM performance on five real-world time series. Here we compare different anomaly detection methods’ performance on precision, recall and F1 score. For all methods, we choose the threshold for classifying a window of the time series as anomalous as the threshold that gives the best F1 score (L: detection window length).

Dataset	Method	L	Prec	Recall	F1
Ambient temperature	Ours	168	0.806	1.0	0.892
	VAE [32]	24	0.686	0.5	0.573
	LSTM-AD [105]	24	1.0	0.5	0.666
	ARMA [114]	24	0.184	1.0	0.311
CPU utilization AWS	Ours	144	0.694	1.0	0.819
	VAE [32]	24	0.348	0.5	0.410
	LSTM-AD [105]	24	0.274	1.0	0.430
	ARMA [114]	24	0.234	1.0	0.380
CPU utilization EC2	Ours	192	0.993	1.0	0.996
	VAE [32]	24	0.949	1.0	0.974
	LSTM-AD [105]	24	1.0	0.436	0.608
	ARMA [114]	24	0.938	1.0	0.968
Machine temperature	Ours	288	0.559	1.0	0.717
	VAE [32]	48	0.211	1.0	0.207
	LSTM-AD [105]	48	1.0	0.5	0.667
	ARMA [114]	48	0.142	1.0	0.248
NYC taxi	Ours	168	0.961	1.0	0.980
	VAE [32]	24	0.662	0.8	0.725
	LSTM-AD [105]	24	1.0	0.2	0.333
	ARMA [114]	24	0.769	0.4	0.526

data, we can use a validation that only contains normal samples to evaluate the distribution of the score function and define the threshold as a given percentile of this distribution, for example.

7.3 Experiments and results

We evaluate our VAE-LSTM algorithm on five real world time series with actual anomaly events: ambient temperature in an office, CPU usage from Amazon Web Services (AWS) and from a server in Amazon’s East Coast data center, internal temperature of an industrial machine and number of taxi passengers in New York City [42]. We compare our algorithm with three other commonly used anomaly detection algorithms for time series: VAE [32], LSTM-AD [105] and ARMA [114]. Table 7.1 lists the numerical results as well as the detection window length. We evaluate three metrics - precision, recall and F1 score (all metrics are evaluated at

the threshold that gives the best F1 score). The detection window length for each dataset is chosen to be equal across all methods. Ours appears longer because of the hierarchical structure of our model which allows us to detect events over longer periods. Counting the true and false positives/negatives among the detection results can be difficult, as anomalous events occur only at a single time stamp whereas all detection algorithms reason over a window. We adopt a simple strategy proposed by [108] to alleviate this issue. Specifically, if a ground truth anomaly is within a detection window where a positive flag $y_t = 1$ is predicted, then we consider the anomaly is correctly detected and all windows that contain the anomaly will be counted as true positives; on the other hand, if none of the windows that contain the anomaly is flagged correctly, then all of them will be counted as false negatives.

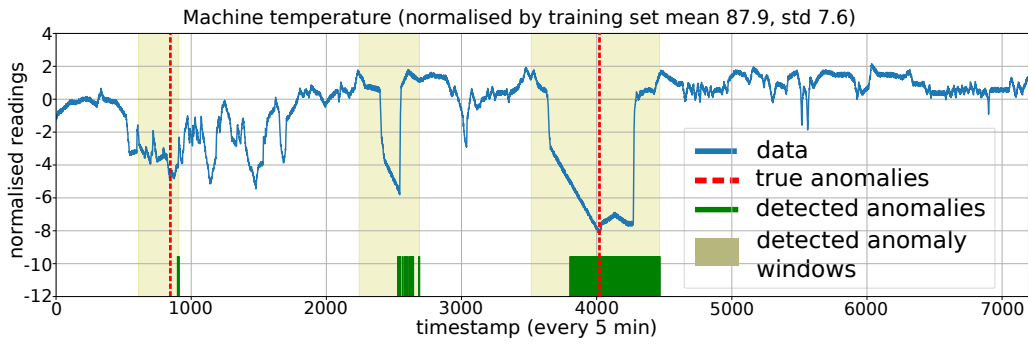
A visualisation of the anomaly detection results given by all four methods on the machine temperature series is shown in Figure 7.3 (see the plot at the end of this chapter). LSTM-AD achieves a high precision in most datasets but suffers from low recall, indicating accurate detected anomalies but high chance of missing out true anomalies (Figure 7.3 (c)). VAE has a good recall but low precision, indicating a significant amount of false positive detections (Figure 7.3 (b)). ARMA does not perform well in either precision and recall and its detection is clearly the worst as (Figure 7.3 (d)).

In contrast, our VAE-LSTM algorithm achieves 100% recall for all datasets, meaning no missed anomaly and the ability to detecting all types of anomalies. At the same time, our method also achieves a decent precision, indicating that the number of false positives is low. For example, a false positive is reported in the ambient temperature series (Figure 7.4(a), see the plot at the end of this chapter) around time $t = 2000$. From visual inspection there exists an unusual spike in the highlighted window and, hence, it might be sensible to raise up human supervisors' attention. We would argue such precaution is indeed beneficial in failure critical scenarios. Good performance in both precision and recall results in a high F1 score achieved by our method, leading all other methods by a good margin. Examples of our method's detection result are given in Figures 7.3(a) and 7.4 (see the plot at the end of this chapter).

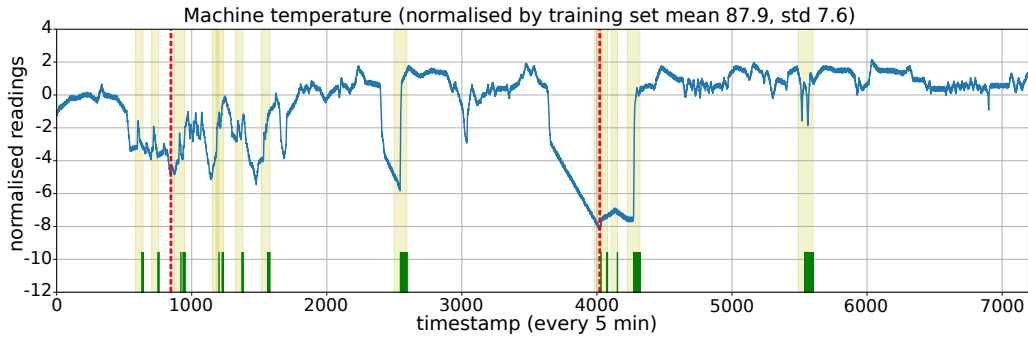
A potential downside of our method is the delay in the anomaly detection for some cases. For example, the first anomaly in the EC2 CPU utilization series is only detected after about 150 time steps. This could be alleviated using multiple scale windows and This is definitely not ideal, but we can mitigate the delay by utilizing our method's ability to form detection sequences over multi-scales. we leave this for future research.

7.4 Summary

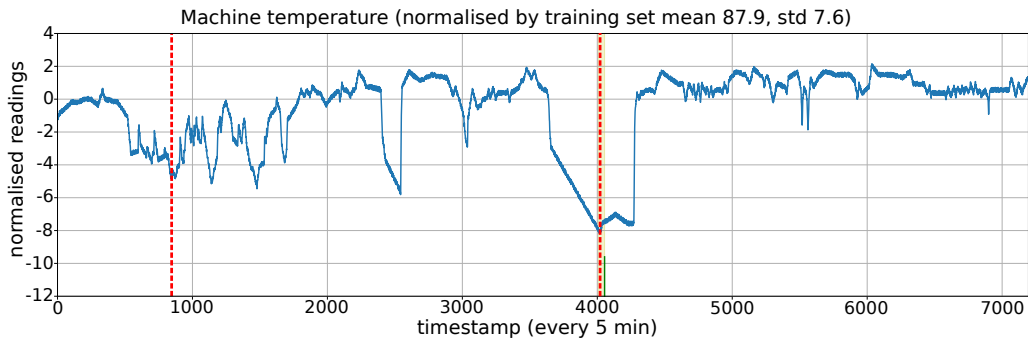
In this chapter, we have proposed a VAE-LSTM hybrid model as an effective approach for anomaly detection in time series. Our model utilizes both a VAE module that forms robust local features over a short local window of an input sequence and a LSTM module that estimates the long term correlations in the sequence. As a result, our detection algorithm is capable of identifying different types of anomalies that might span over multiple time scales. We demonstrate the effectiveness of our detection algorithm on five real world sequences and show that our method outperforms other commonly used detection methods.



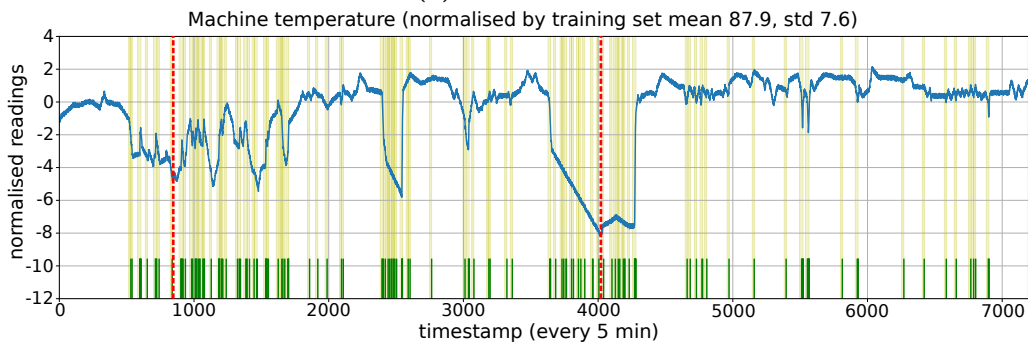
(a) Our VAE-LSTM algorithm.



(b) VAE.



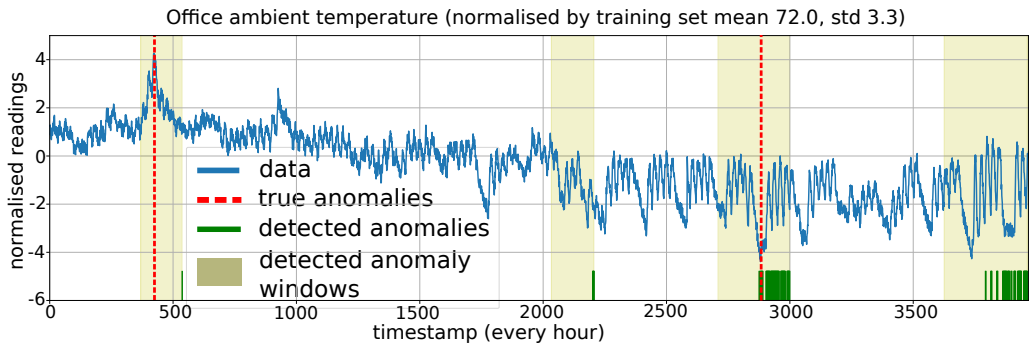
(c) LSTM-AD.



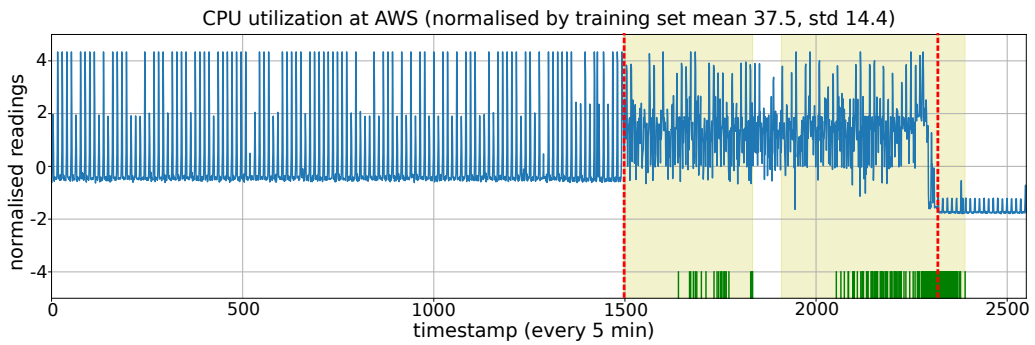
(d) ARMA.

Figure 7.3: Visualizing the detected anomalies from different methods. We show the detected anomalies (indicated by the short green line and the yellow region) using our VAE-LSTM, VAE, LSTM-AD and ARMA methods on a time series that records the temperature of an industrial machine. Our method manages to detect both anomalies without generating many false positives.

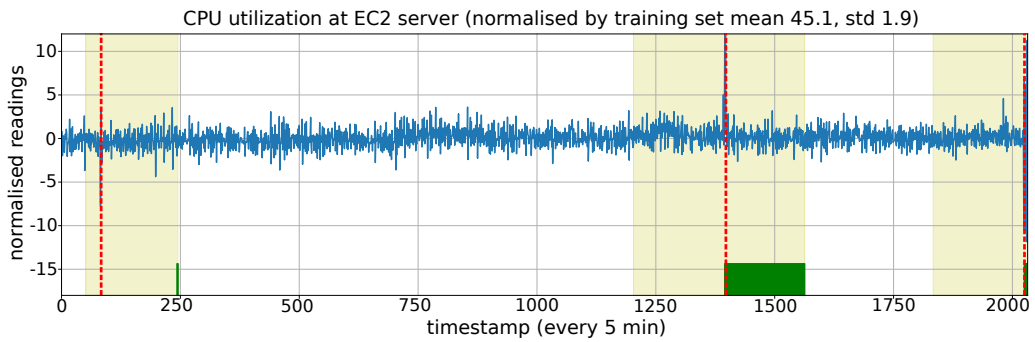
7. Anomaly Detection for Time Series



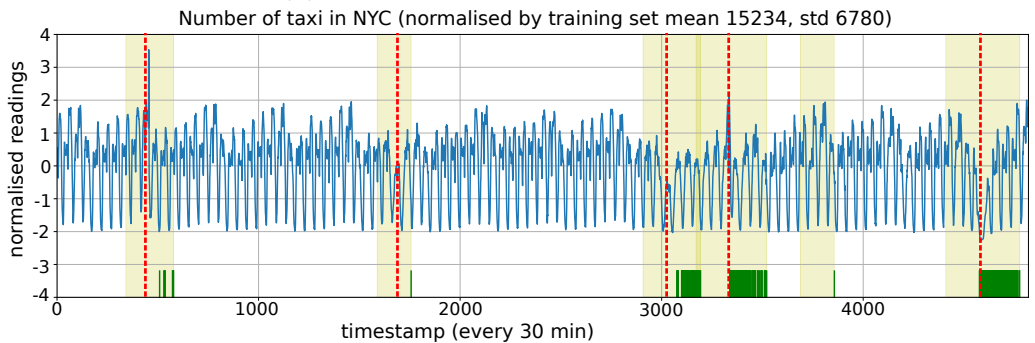
(a) Ambient office temperature.



(b) CPU utilization at AWS server.



(c) CPU utilization at EC2 server.



(d) The number of NYC taxi passengers.

Figure 7.4: Visualizing the detected anomalies from our method on other time series. Here we should the detected anomalies given by our VAE-LSTM hybrid model on another four time series. In all cases, our method manages to detect all anomalies, while occasionally generating some false positives.

8

Uncertainty estimation for classification

Following the success of applying variational autoencoders (VAEs) to anomaly detection, we recognize that a VAE model’s capability to infer whether a data sample has been seen frequently in the training data can be extremely useful for uncertainty estimation. In fact, the ability to detect out-of-distribution data samples (the data samples that are not from the training distribution) and domain shifts (where the data distribution at test time is different from the data distribution in the training time) is closely related to uncertainty estimation [35], [115]–[117]. Therefore, we propose a hybrid classifier that contains a generative unit in the form of a variational autoencoder (VAE) which maps an input sample to a low-dimensional posterior distribution and a discriminative unit that infers a distribution over different class labels from the posterior distribution estimated from the VAE’s encoder.

In this chapter, we first introduce how our hybrid classifier is designed and can be used to estimate class distributions that give sensible uncertainty estimation in the challenging scenarios when the training labels are limited or when OOD data are encountered. We also demonstrate the performance of our method on a public medical dataset, HAM10k [43], for the task of pigmented skin lesion diagnosis. We show that our method gives better calibrated confidence prediction compared to the commonly used uncertainty estimation approaches, such as the MC dropout approach [36] and the deep ensemble approach [30].

The relevant content in this chapter is published as a conference paper - Uncertainty Estimation with a VAE-Classifer Hybrid Model at *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* 2022.

8.1 Background and related work

Uncertainty estimation has been a key topic in machine learning research [118]. We have seen machine learning algorithms achieve state-of-the-art performance in a wide range of tasks, such as image recognition [6] and natural language processing [119]. However, to deploy these algorithms in real world applications, we need them to be able to quantify the uncertainty associated with their prediction. This is crucial for applications such as healthcare and autonomous driving, where an over-confident wrong decision can be detrimental. Furthermore, in many situations, the machine algorithms will need to work alongside humans. The ability of these algorithms to sensibly predict uncertainty can help to indicate when human intervention might be needed and also help to build trust between the human operators and the machine algorithms.

Evaluating a method’s uncertainty performance is a challenging task. We focus on two aspects that we consider are important for real world applications. Firstly, we examine calibration, which measures uncertainty from a frequentist’s perspective and highlights the discrepancy between a method’s predictive confidence and the empirical accuracy [30], [120], [121]. Intuitively, a well-calibrated method will report a 80% confidence when the prediction is correct 80% of the time. Therefore, calibration can give a clear indication whether a method is over- or under-confident in its prediction. Secondly, we examine whether a method can report sensible uncertainty when out-of-distribution (OOD) data are presented [115]. As domain shifts are common in real world applications, being sure about what a method does not know and reporting low confidence accordingly is an important property.

Uncertainty estimation is a fast growing field. Bayesian neural networks [122], [123] introduce priors over the network parameters and then infer the posterior distributions of the parameters given the training data. This can be done using sampling-based approaches such as MCMC or HMC [124]–[127]. Alternatively, the posterior distribution can be inferred using approximations, such as Laplace approximation [128], [129], variational inference [130]–[132] and Monte-Carlo dropout [36]. Non-Bayesian approaches, such as training ensembles of deterministic NN classifiers [30] or re-calibrating the predictive distribution with held-out validation data [133], have also shown effective performance in terms of classification accuracy and calibration quality.

8.2 Our Method

In this section, we first describe how we introduce a generative module in the classification task. We show how to make class prediction and uncertainty estimation in our hybrid classifier setup. We then introduce how to train our model and the statistical treatment needed to consider OOD data.

8.2.1 Classification with a hybrid model

For our learning task, we assume a training dataset $\mathcal{D}_N$ that consists of N i.i.d. pairs of input-label samples, i.e. $\mathcal{D}_N = \{\mathbf{x}_i, y_i\}_{i=1}^N$, where the input $\mathbf{x} \in \mathbb{R}^{L_x}$ represents a L_x -dimensional sample and the label y denotes one of K classes, i.e. $y \in \{1, \dots, K\}$. During training, we aim to estimate the predictive distribution $p_\theta(y|\mathbf{x})$ over the class labels, where θ denotes the model parameters to be determined. To make a prediction for a new input observation $\mathbf{x}_t$, we compute the predictive distribution and the predicted label is sampled from this distribution, i.e. $y_t \sim p_\theta(y|\mathbf{x} = \mathbf{x}_t)$.

To estimate $p_\theta(y|\mathbf{x})$, we introduce a latent variable $\mathbf{z} \in \mathbb{R}^{L_z}$. With $\mathbf{z}$, we can rewrite $p_\theta(y|\mathbf{x})$ as:

$$p_\theta(y|\mathbf{x}) = \int_{\mathbf{z}} p_\theta(y|\mathbf{z})p_\theta(\mathbf{z}|\mathbf{x})d\mathbf{z} = \mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x})}[p_\theta(y|\mathbf{z})], \quad (8.1)$$

where $p_\theta(\mathbf{z}|\mathbf{x})$ denotes an encoder that maps an input sample to a distribution in the latent space and $p_\theta(y|\mathbf{z})$ denotes a classifier that is conditioned on the latent variable. The significance of this formulation is that we have introduced a latent representation symbolized by $\mathbf{z}$ in the class prediction task. This representation can be very useful in practice. For example, we can utilize additional unlabelled data (i.e. data that does not have labels for y) to learn a better encoder $p_\theta(\mathbf{z}|\mathbf{x})$ and/or we can reuse the same encoder for multiple tasks.

According to Eq. 8.1, we need to obtain both the encoder $p_\theta(\mathbf{z}|\mathbf{x})$ and the classification module $p_\theta(y|\mathbf{z})$ to evaluate the predictive distribution $p_\theta(y|\mathbf{x})$. In this paper, we derive the latent representation with a variational auto-encoder (VAE, [16], [17]) and implement the classifier via a multi-layer perceptron (MLP [134]) with a softmax activation at the last layer. Putting the two modules together, we compute the overall predictive distribution $p_\theta(y|\mathbf{x})$ via sampling, as an analytical form for the integration in Eq. 8.1 is not available. Specifically, we draw M samples of the latent features from the encoder, i.e. $z_m \sim p_\theta(\mathbf{z}|\mathbf{x}_i)$, and approximate the expectation by taking the average of these samples, i.e. $\mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x}_i)}[p_\theta(y|\mathbf{z})] = \frac{1}{M} \sum_{m=1}^M p_\theta(y|z_m)$.

The class prediction $\hat{y}$ is made by selecting the most probable label according to $p_\theta(y|\mathbf{x})$ and the confidence τ is reported by the corresponding probability, i.e. $\hat{y} = \arg \max_k p_\theta(y=k|\mathbf{x})$, $\tau = \max_k p(y=k|\mathbf{x})$.

8.2.2 Optimizing model parameters

A VAE [16], [17] contains a probabilistic encoder $q_\theta(\mathbf{z}|\mathbf{x}_i)$ that maps an input sample $\mathbf{x}_i$ to the latent space, and a decoder $p_\phi(\mathbf{x}|\mathbf{z}_i)$ that generates an input sample from a latent feature $\mathbf{z}_i$. We make the further assumption that the encoder is parameterized by a diagonal Gaussian, i.e. $q_\theta(\mathbf{z}|\mathbf{x}_i) = \mathcal{N}(\mu_\theta(\mathbf{x}_i), \sigma_\theta^2(\mathbf{x}_i))$ and that latent representations for the samples are confined to a prior distribution $p(\mathbf{z})$, which is chosen to be a unit Gaussian distribution. To estimate the parameters for the encoder $p_\theta(\mathbf{z}|\mathbf{x})$, we maximize VAE’s evidence lower bound (ELBO) objective $\mathcal{L}_{\text{ELBO}} = \frac{1}{N} \sum_i \mathcal{L}(\mathbf{x}_i)$, where $\mathcal{L}(\mathbf{x}_i)$ is defined as follows:

$$\mathcal{L}(\mathbf{x}_i) = \mathbb{E}_{q_\theta(\mathbf{z}|\mathbf{x}_i)}[\log p_\phi(\mathbf{x}_i|\mathbf{z})] - D_{\text{KL}}[q_\theta(\mathbf{z}|\mathbf{x}_i)||p(\mathbf{z})]. \quad (8.2)$$

We train a VAE on the available training input samples, i.e. $\mathcal{D}_x = \{\mathbf{x}_i\}_{i=1}^N$. We then take the optimized VAE encoder as the encoder in Eq. 8.1, i.e. $p_\theta(\mathbf{z}|\mathbf{x}) = q_\theta(\mathbf{z}|\mathbf{x})$.

To optimize the parameters in the MLP, we maximize the log likelihood of the class prediction with respect to $\mathcal{D}_N$, i.e. $\mathcal{L}_{\text{LL}} = \frac{1}{N} \sum_{i=1}^N \mathbf{e}_{y_i} \log p_\theta(y|\mathbf{x}_i)$, where $\mathbf{e}_i$ denotes the standard base vector with 1 at i -index and 0 elsewhere.

The best classification performance would be achieved when both the VAE module and the MLP module are trained together, as the encoder is also optimized to give good class prediction. In practice, it is more useful to train the modules in separate stages, as the input and the label might not be available at the same time or the same input might be applied for a different classification task. To reflect the performance in reality, we first optimize the encoder’s parameters by maximizing the ELBO objective and then optimize the MLP’s parameters by maximizing the log likelihood objective.

8.2.3 Considering out-of-distribution data

One issue with the standard VAE formulation described in the previous section is that it does not handle out-of-distribution (OOD) data sensibly. Very often, the encoder projects OOD samples to sharp Gaussians at locations that correspond to specific in-distribution (ID) samples. This causes the latent classifier to make over-confident prediction on these samples. We address this issue by incorporating a

hypothesis parameter H in the encoder module, as follows:

$$p(\mathbf{z}|\mathbf{x}_i) = \int_H p(\mathbf{z}|\mathbf{x}_i, H)p(H|\mathbf{x}_i)dH \quad (8.3)$$

Given a data sample $\mathbf{x}_i$, H has two possible outcomes $\{H_{\mathcal{D}}, H_{-\mathcal{D}}\}$, where $H_{\mathcal{D}}$ denotes $\mathbf{x}_i$ is ID data, $H_{-\mathcal{D}}$ denotes the opposite and $p(H_{\mathcal{D}}|\mathbf{x}_i) + p(H_{-\mathcal{D}}|\mathbf{x}_i) = 1$.

During training, we assume that all data samples are ID, so $p(H_{\mathcal{D}}|\mathbf{x}_i) = 1$ and the encoder simplifies to the original VAE's encoder, i.e. $p(\mathbf{z}|\mathbf{x}_i, H_{\mathcal{D}}) = \mathcal{N}(\mu_{\theta}(\mathbf{x}_i), \sigma_{\theta}^2(\mathbf{x}_i))$. During inference, the input sample $\mathbf{x}_i$ can be either be OOD or ID. Therefore, we have to evaluate the encoder by considering both scenarios, according to the following equation:

$$p(\mathbf{z}|\mathbf{x}_i) = p(\mathbf{z}|\mathbf{x}_i, H_{\mathcal{D}})p(H_{\mathcal{D}}|\mathbf{x}_i) + p(\mathbf{z}|\mathbf{x}_i, H_{-\mathcal{D}})p(H_{-\mathcal{D}}|\mathbf{x}_i), \quad (8.4)$$

where $p(\mathbf{z}|\mathbf{x}_i, H_{\mathcal{D}})$ denotes the original VAE's encoding distribution. We define the other three probabilities as follows:

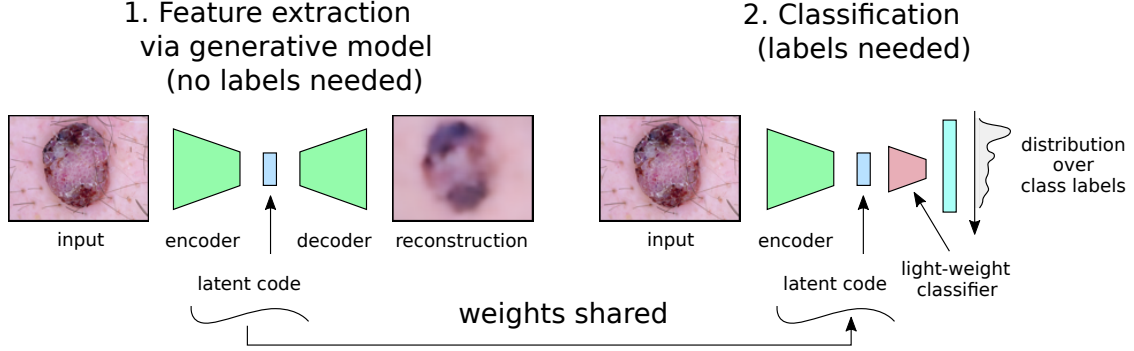
$$p(H_{\mathcal{D}}|\mathbf{x}_i) = \frac{1}{1 + \exp(a \cdot \text{ELBO}(\mathbf{x}_i) + b)}, \quad (8.5)$$

$$p(H_{-\mathcal{D}}|\mathbf{x}_i) = 1 - p(H_{\mathcal{D}}|\mathbf{x}_i), \quad (8.6)$$

$$p(\mathbf{z}|\mathbf{x}_i, H_{-\mathcal{D}}) = \mathcal{N}(0, \boldsymbol{\mu}_{\max}^2), \quad (8.7)$$

where the hyperparameters a and b are chosen using the validation set so that $p(H_{\mathcal{D}}|\mathbf{x}_i)$ is 0.5 at -3 standard deviation away from the mean ELBO and 0.05 at the minimum ELBO; $\boldsymbol{\mu}_{\max}$ denotes the maximum absolute value of the encoding mean across all latent dimensions in the validation set; $\text{ELBO}(\mathbf{x}_i)$ evaluates the ELBO objective for an input sample $\mathbf{x}_i$ according to Eq. 8.2. The above equations ensure that an input sample with a high ELBO value will be assigned a high probability for $p(H_{\mathcal{D}}|\mathbf{x}_i)$ and its encoding will be primarily governed by the VAE's encoder - sharp and localized in the latent space. On the other hand, an input sample with low ELBO value will be assigned a high probability for $p(H_{-\mathcal{D}}|\mathbf{x}_i)$ and its encoding will be dominated by an uninformative wide Gaussian. Such an encoder allows the latent classifier to behave more sensibly, as the predictive distribution (averaged over multiple samples from the latent distribution) will tend to spread out over different class labels for the OOD data.

Training



Testing

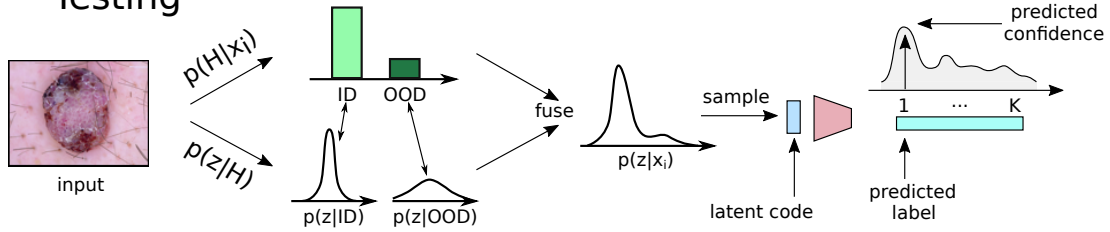


Figure 8.1: Our method overview. To train our system, we first train a VAE on the input data (no labels are needed at this stage). After the ELBO loss of the VAE has converged, we freeze the parameters in the VAE encoder and train a light-weight classifier which predicts a class distribution for the latent code drawn from the posterior distribution estimated by the encoder module. To deploy our system for testing new samples, we need to consider the scenario where an input sample might not be drawn from the same distribution of the training data. To do so, we added an additional out-of-distribution detection mechanism as introduced in Section 8.2.3.

8.2.4 System overview

Putting everything together, we demonstrate how our VAE-classifier hybrid system is trained and is used with a skin lesion classification example in Figure 8.1. We first train the parameters in our hybrid model in two stages: 1) we train a VAE model on the entire dataset of different skin lesions and this allows our method to represent an input sample with a low-dimensional feature vector; 2) with the trained VAE’s encoder (we kept its weights frozen), we then train a light-weight classifier (a 3-layer of fully connected neural network) under the cross-entropy classification loss.

When the training is complete, we can deploy our model for new samples. However, we need to consider the possibility that the new observations might be an OOD sample, as discussed in Section 8.2.3. To address this issue, we infer the probability of both situations when the test sample is in-distribution and OOD.

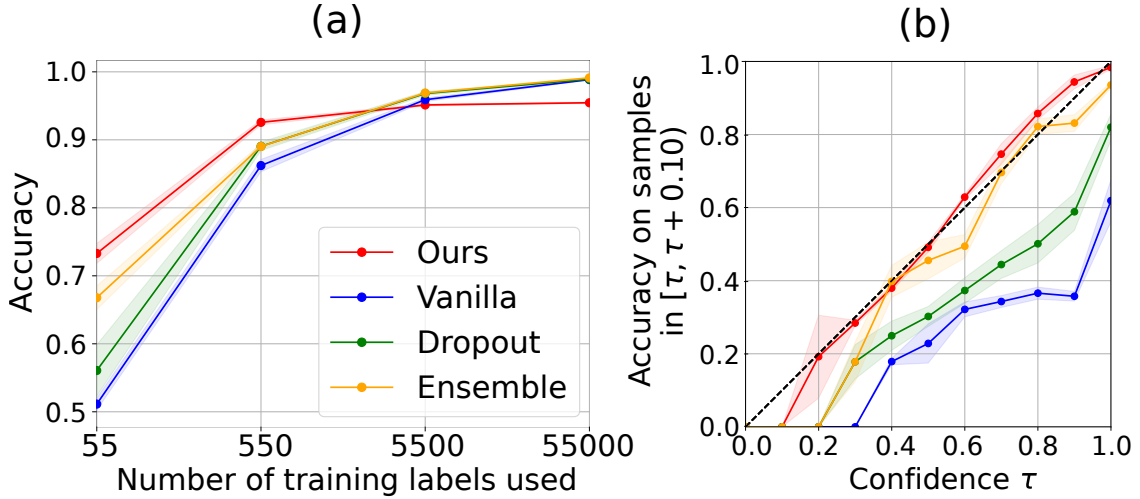


Figure 8.2: Label efficiency study. Classifiers are trained with 55, 550, 5500 or 55K training labels available. (a) Test accuracy. (b) calibration plot for classifiers trained at 55 labels. Our method outperforms other approaches in accuracy and uncertainty calibration at limited label settings.

We then marginalize the different hypotheses and obtain an estimate of the latent distribution $p(\mathbf{z}|\mathbf{x}_i)$. By drawing multiple samples from this posterior distribution, our method gives as output both the predicted class label and the corresponding confidence for this prediction.

8.3 Experiments and Results

We evaluate our method on two datasets: 1) MNIST [39], which contains 60k/10k train/test images and corresponding labels of handwritten digits; 2) HAM10k [43], a medical dataset for pigmented skin lesion diagnosis, which contains 10015 dermatoscopic images and corresponding lesion labels. We compare our method against three baseline approaches: a standard neural network classifier (Vanilla [115]), a classifier trained with Monte-Carlo dropout at a rate 0.1 (Dropout [36]) and ensembles of $M = 3$ vanilla classifiers (Ensemble [30]). We use the same model architecture for all the approaches - i.e. LeNet [135] for MNIST and ResNet18 [136] with a 3-layer MLP for HAM10k. We evaluate two metrics - classification accuracy and calibration. The calibration is evaluated via a calibration plot, where the test samples are grouped into 10 buckets according to their confidence τ (defined in Sect. 8.2.1). The accuracy of each group is evaluated and plotted against the centre confidence of each bucket. A well calibrated classifier will generate line along the diagonal.

We first show that our method works better than other approaches when training labels are limited. We train different approaches with different amount of

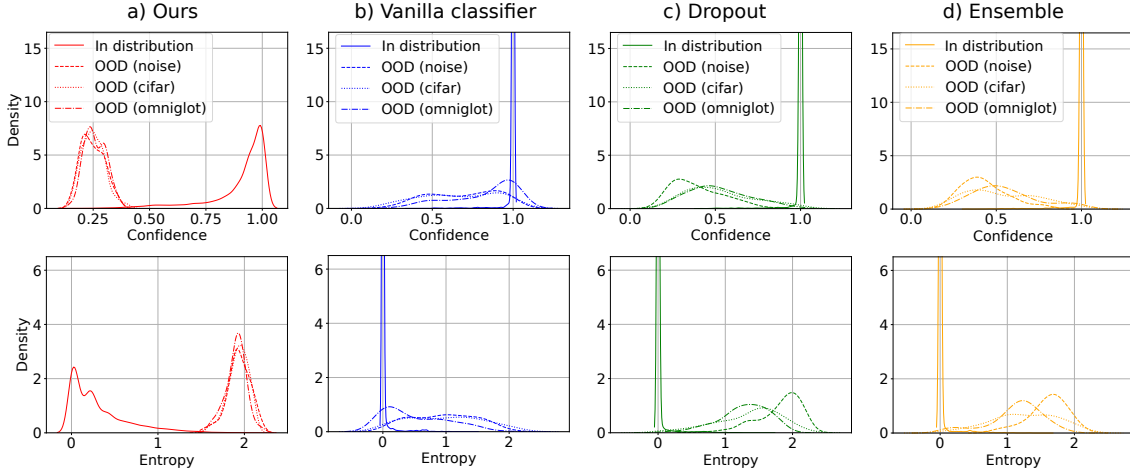


Figure 8.3: Uncertainty estimation for out-of-distribution (OOD) data. Our method keeps the predicted confidence on OOD data low, whereas other approaches tend to make high confidence prediction on OOD data.

Table 8.1: Classification accuracy on a skin lesion detection task (HAM10k test set).

	Ours	Vanilla	Dropout	Ensemble
Accuracy	74.9 ± 0.3	82.0 ± 1.1	82.3 ± 0.5	84.0 ± 0.3

training labels available on the MNIST dataset. As shown in Figure 8.2a, our method gives the best accuracy among all approaches when extremely limited training labels are available (e.g. 55 or 550). This indicates that the generative unit helps our method to derive better representation that compensates the limitation in the amount of the training labels available. Furthermore, our method offers well-calibrated uncertainty estimation that follows the diagonal almost perfectly in Figure 8.2b, whereas the other approaches show various levels of over confidence as they are likely over-fitted to the limited training examples.

We now investigate how different approaches react to OOD data. We train all approaches on the MNIST training data and we plot the confidence and the entropy distribution of all the classifiers on the test set of MNIST (in-distribution) and the test set of three other datasets - random noise, CIFAR10 and OMNIGLOT (OOD). The results are shown in Figure 8.3. All three baseline approaches make high confident prediction (confidence > 0.5 or entropy < 1.0) on a significant distribution of the OOD data. In contrast, our method keeps all the OOD data in the low confidence and high entropy range. This indicates our method do not make over-confident prediction on unfamiliar samples, which is a very desirable property with high-stakes applications such as healthcare.

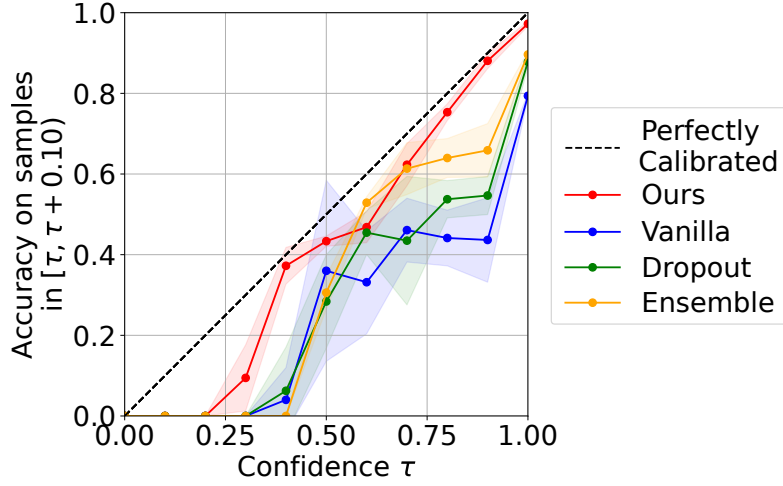


Figure 8.4: Calibration plot for the skin lesion detection task (HAM10k test set). Our method (red line) gives the best calibrated uncertainty prediction.

We now evaluate our method on a medical dataset HAM10k, where the goal is to predict skin lesion type from a dermoscopic image. We train all classifiers until the validation loss has converged. Prior training our latent classifier, we first train the VAE module for 200 epochs when the ELBO objective on the validation set has converged. The test accuracy on all classifiers is reported in Table 8.1, where our performance seems to lag behind other approaches with the ensemble classifier giving the best accuracy.

However, we see a different picture, when we investigate the calibration plot in Figure 8.4. Although reporting a lower accuracy, our method is the best calibrated approach among all. This indicates that the majority of the wrong predictions made by our approach occur at the low confidence regime. As the predicted confidence increases, our method becomes more accurate. The other approaches render a varying degree of over-confidence. Their mis-calibrated accuracy at high confidence prediction is worrying. For example, the accuracy for the ensemble classifier at the confidence bracket $[0.8, 0.9)$ is 65.5%. This indicates 34.5% of these high-confidence predictions are wrong. In applications such as cancer diagnosis, such overly confident wrong prediction can be detrimental.

We show the importance of having a more calibrated confidence prediction in an example illustrated in Figure 8.5. Here we show 64 randomly sampled test images from HAM10k dataset with their predicted lesion class and the prediction confidence using our method in subfigure (a) and the ensemble method in subfigure (b). We rank the samples in an ascending order of the predicted confidence and we highlight the wrong predictions in red boxes. As can be seen, our method make wrong predictions when it is uncertain about this prediction (i.e. at low confidence),

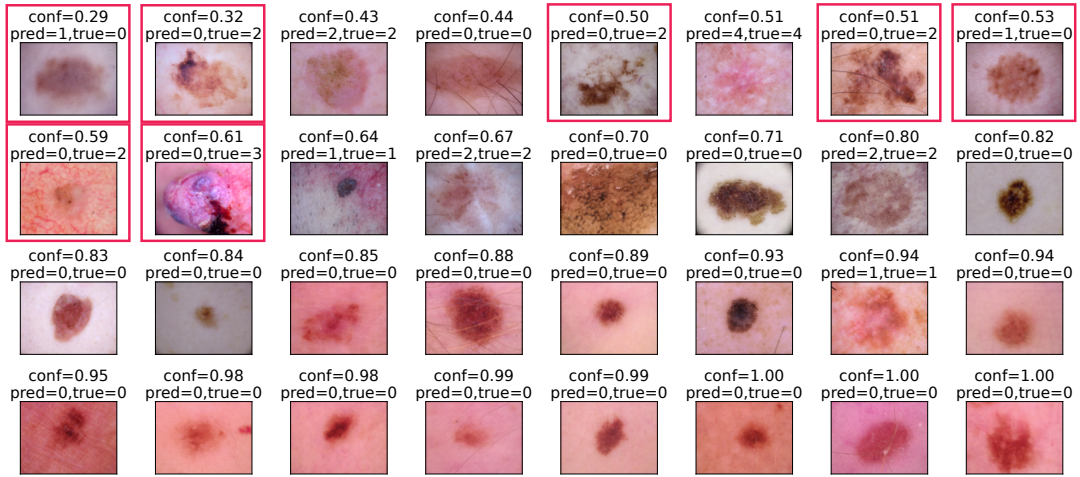
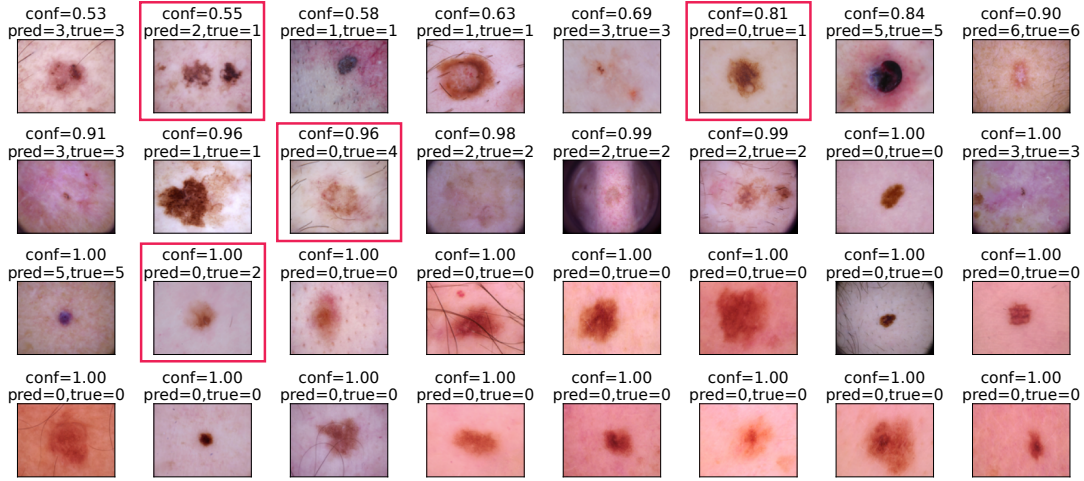
(a) The classification and confidence prediction using **our method**.(b) The classification and confidence prediction using **the ensemble method**.

Figure 8.5: Our method makes wrong prediction when the predicted confidence is low, whereas the ensemble method gives over-confident predictions. (a) Although making more incorrect predictions, our method contains all of the error cases (highlighted in the red boxes) in the low confidence range. (b) In contrast, the ensemble method predicts high confidence values for its error cases, e.g. three of the four error cases have a confidence over 80%.

whereas the ensemble method gives overly confident wrong predictions - 3 out of the 4 wrong predictions are made with a confidence over 80% and 2 are with a confidence over 95%. In a safety-critical application, such as cancer prediction in this case, high confidence mistakes such as the ones shown in Figure 8.5b can be detrimental. Although the ensemble method has a higher accuracy, we believe the prediction made by our method is safer and more trustworthy to be deployed in the real world.

8.4 Summary

In this chapter, we design a new classification system which gives a more calibrated uncertainty estimation. Our system consists of two units. We first map a data sample to a posterior distribution in the low dimensional latent space with a VAE’s probabilistic encoder. We then draw samples from the posterior distribution and pass these samples through a deterministic neural network classifier. The final class distribution is taken as an average of the predictions made by all samples. We add a marginalization treatment on the posterior distributions from the VAE encoder at test time. This ensures the posterior distributions predicted are indicative of the out-of-distribution samples that are likely to be presented during test time. We show that our classifier gives a more calibrated confidence prediction compared to other uncertainty estimation methods, such as MC dropout and deep ensemble in both MNIST (digit and fashion) and a real world skin lesion classification task.

9

Conclusion

In this thesis, we have focused our discussion on an important type of generative models, namely variational auto-encoders (VAEs). In this chapter, we will summarize our contributions and introduce thoughts for future work, following up from the ideas discussed in the thesis.

9.1 Contributions

The thesis provides a self-contained tutorial on the topics of variational inference (VI) and VAEs. The objective of this tutorial is to give sufficient details on how the VI framework can be used to estimate a probabilistic model and how the traditional VI framework is adapted to define a VAE model, important as existing literature is often rather brief regarding these topics. In Chapter 2, we provide a thorough introduction to the Variational Inference (VI) framework. Crucially, we show how the evidence lower bound (ELBO) is derived. We demonstrate how VI can be applied to estimate the parameters in a probabilistic Principal Component Analysis (pPCA) model. In this example, we show that the model parameters can be inferred by optimizing the ELBO objective via gradient descent in a similar way to the method in which the model parameters of VAEs are inferred. We then introduce VAEs as a non-linear extension of the pPCA model in Chapter 3. We introduce key assumptions and techniques, including amortization, Monte Carlo sampling, the reparametrization trick and unbiased estimation via stochastic sampling; all of which are needed to successfully apply the VI framework to VAEs. We believe that the introduction provided in this thesis can help to clarify the key concepts needed to better understand VAEs, and is aimed at readers with limited knowledge of VI or generative modelling.

Although improving the VAE models' generative modelling performance is an active research topic and many works have already been published in this domain, most of the existing literatures are rather brief on explaining why particular issues,

such as low quality data sample being generated from a derived VAE model, are encountered, or the explanation is mathematically heavy and hard to make intuitive sense. In this thesis, we focus on offering illustrative examples to explain a range of challenges that VAE models often face. For example, we visualise the amount of information loss in the derived posterior distribution caused by an over-regularised ELBO in Figure 4.1. We also demonstrate the difference between the unit-Gaussian prior and the derived aggregate posterior in Figure 4.2. These illustrative examples are crucial to relate the abstract mathematical terms to the corresponding data properties. For example, the posterior distribution is a distribution in the latent space that represents the embedding of a single data sample and we illustrate this clearly in the 2D plot of a batch of posteriors in Figure 4.1. We hope the illustration in this thesis helps the readers to build a better understanding of VAE models and their limitations. We also hope the approaches we adopt in this thesis to visualise the representations derived in a VAE model can be helpful for the readers to demonstrate their problems or proposals.

In Chapter 5 and 6, we propose solutions to the commonly encountered problems, including over-regularized posteriors, an inflexible prior model and the encoding drift effect in the derived auto-encoding function. As there is not a universal solution that can address all three problems at once, we propose a separate solution to each of the above problems. All three solutions are general to any VAE based approaches, and respect the variational inference framework and derive a valid lower bound to the data log likelihood as their learning objective. Therefore, the proposed solutions can be applied in combination with any existing techniques that also improve VAE’s performance.

Among the three approaches, the proposal to optimise the variance parameter in the Gaussian decoder under the ELBO objective is the simplest and yet the most effective. Optimising such a variance parameter can significantly increase the test set ELBO value as shown in Chapter 5.1.3. We encourage readers to adopt this technique in the training of their VAE models whenever a Gaussian decoder is adopted. The generative prior proposed in LaDDer is not trivial to implement and the training often requires an iterative procedure that switches between training the outer VAE model and the inner prior VAE model. The benefit of adopting a more flexible prior as such is to have better control in the derived data distribution. With this prior, we will be able to identify the low density region where the data samples are rarely projected and the high density region where the majority of data samples reside. Therefore, we encourage the readers to adopt this technique if their applications require generating data samples from the derived generative model

and/or searching through the derived data distribution to find the embedding of a target data sample. Our proposal that mitigates the encoding drift effect investigates a behaviour that is rarely discussed in the VAE context. According to our evaluation, the proposed approach barely improves the test ELBO metric, but it can effectively mitigate the encoding drift effect. Therefore, we leave this to the readers to decide whether they would like to adopt this technique depending on the problems they have at hand.

In the later chapters of this thesis (Chapter 7 and 8), we demonstrate how VAE models can be applied to solve important machine learning tasks, such as anomaly detection and uncertainty estimation. It is worth noting that in both cases VAEs alone are not enough to provide a full solution. Instead, we adopt VAEs as an effective feature extraction unit that project the high-dimensional raw data to a low-dimensional representation in terms of the posterior distribution. We then apply further operations on the derived latent representations, such as the LSTM unit in the anomaly detection system to allow the system to investigate the signal over a longer span of the raw signal or the classifier unit in the uncertainty estimation system to allow the system to make a more calibrated confidence estimate by averaging the latent samples drawn from the posterior distribution. We encourage the readers to also consider using VAEs as the feature extraction module if such a module is needed in the problems at hand.

In the next section, we outline a few insights and possible future research directions arising from the work done in this thesis.

9.2 Future work

In this thesis, we have proposed three methods that address different issues in the VAE learning scheme. The proposal of inferring the variance parameter in a Gaussian decoder has been applied in our works for anomaly detection and for uncertainty estimation. The VAEs trained with the variance parameter obtains a much higher ELBO and more importantly for these two tasks, the posterior distributions of different data samples become more distinctive and hence serve as a better input for the subsequent modules, such as the LSTM module for anomaly detection and the light-weight classifier for predicting a class distribution. Therefore, the effectiveness of this proposal has been verified in real world applications. On the other hand, the other two proposals, although having been thoroughly evaluated and demonstrated to be effective using the standard datasets, such as MNIST and CelebA, have not been applied to a real world application. Therefore, a direct ex-

tension on these two works will be to demonstrate their effectiveness in a real world application.

For the consistency loss, we can explore its potential in making VAEs more robust to adversarial attacks. Adversarial attacks refer to the situations where a small but intentional change has been applied to the input sample so that the output of the targeted system will be changed [137]. VAEs have been applied to this task but have shown vulnerabilities [138], [139]. An effective attack has been shown in these papers as searching for perturbations of the input sample that will result in a latent representation of the targeted sample. In this sense, a VAE with an encoding function that encodes very different input samples close together in the latent space will render this vulnerability. The consistency loss can potentially help to mitigate such effects, as it encourages the encoding function to infer a posterior that does not shift to the posterior distribution of another sample. It would be interesting to see the impacts of the consistency loss play in this tasks and potentially some modification might be needed to deliver the best improvement.

For LaDDer, there are two potential areas that we would be interested in seeing how LaDDer performs. Firstly, we have seen some exciting progress in the generation quality of VAEs thanks to carefully designed model architectures that aim to generate high-fidelity images [140]. It would be interesting to see if LaDDer coupled with this new architecture can deliver even better generation quality.

Secondly, we have pointed out the inferred prior distribution can be see as a proxy of the data. We can use this prior distribution to set up a constraint for a downstream task that may require searches or optimization in the data distribution. We have given an example of this setup in a simple example of latent space interpolation, where the interpolated samples should be chosen from the data distribution. We would be interested in demonstrating the power of this technique in a more practical scenario. A good example is the camera pose estimation task, as formulated in [1]. In this example, a 6 DoF camera pose and a depth image needs to be estimated from an observation of a pair of input images. The authors propose to train a VAE to represent the distribution of real depth images and they optimize a re-projection error constraint with respect to the camera pose and the latent code of the depth image. Using LaDDer prior, we can add an additional regularization term to the optimization that penalizes the latent code with low likelihood in the prior. This way we can effectively constrain the search of the depth latent code to the ones that correspond to real depth images.

For the work on anomaly detection, we regard a natural extension as the application of our VAE-LSTM detection system to a time series of high-dimensional

input samples, such as video data. We have done some early work in this direction in collaboration with researchers from ABB, focusing on sensor data from their power station. In this project, we simply treat a window of the sensor readings as an input sample for the VAE module and estimate a latent encoding for this window. However, this same technique may not be appropriate for the video data, as a window of images would result in a quite high-dimensional input for a VAE module to process and there is a high-level of redundancy between neighbouring frames which can be removed before passing to the VAE encoder. Therefore, a better strategy can be deploying another VAE module that focuses on encoding the raw images. After converting the video data to a sequence of latent embeddings, we can apply our VAE-LSTM system to this time series of image embeddings.

For the work on uncertainty estimation, we think there are multiple potential research directions that the work can be extended toward. For example, it would be interesting to extend the current work to regression tasks. In addition, with the ability to detect out-of-distribution (OOD) data, our method can draw special attention to the situations where a persistent group of OOD samples are detected. These behaviours can be resulted from adversarial attacks or a domain shift. For both cases, it probably is better to introduce human intervention. Specifically for the case of domain shift, it might be sensible to retrain the classifier model so that it can work with this new domain of inputs. Furthermore, our treatment to the posterior distributions makes it to be representative and indicative of its degree of confidence that the input sample is drawn from the data distribution. This treatment is general and can be applied to any VAE models. We are interested in testing out whether a VAE module with the treatment can be better applied to tasks, such as active learning [141] and reinforcement learning [142] where uncertainty estimation and the ability to detect OOD samples play a key in the performance.

In conclusion, we have investigated multiple areas of research related to VAEs and applications of VAEs. From our experience, VAEs are easy to train and present several desirable properties that make them a perfect component for many challenging tasks. For example, the VAEs' ability to model the normal behaviours in a training data, without labels, and to compress the raw data in a more compact form, plays a key role in our anomaly detection system. Although being widely adopted, there are still open research questions related to VAEs. We have listed a few possible directions as the potential extension for works carried out in this thesis. We hope the thesis can be a useful resource to readers who are interested in VAEs and their applications. We believe that VAEs and their variants will further expand their impacts to a wider domain of applications and contribute to building

AI systems that are robust and interpretable.

Bibliography

- [1] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison, “Codeslam—learning a compact, optimisable representation for dense visual slam,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2560–2568.
- [2] J. Lim, S. Ryu, J. W. Kim, and W. Y. Kim, “Molecular generative model based on conditional variational autoencoder for de novo molecular design,” *Journal of cheminformatics*, vol. 10, no. 1, pp. 1–9, 2018.
- [3] T. R. Shaham, T. Dekel, and T. Michaeli, “Singan: Learning a generative model from a single natural image,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4570–4580.
- [4] M. Huh, P. Agrawal, and A. A. Efros, “What makes imagenet good for transfer learning?,” 2016.
- [5] S. Kornblith, J. Shlens, and Q. V. Le, “Do better imagenet models transfer better?” In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019.
- [6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li, “Imagenet: A large-scale hierarchical image database.,” in *CVPR*, IEEE Computer Society, 2009, pp. 248–255, ISBN: 978-1-4244-3992-8.
- [7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, ser. CVPR ’14, USA: IEEE Computer Society, 2014, pp. 580–587.
- [8] K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos,” in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS’14, Montreal, Canada: MIT Press, 2014, pp. 568–576.
- [9] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik, “Human pose estimation with iterative error feedback,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4733–4742.

- [10] J. Donahue, L. A. Hendricks, M. Rohrbach, *et al.*, “Long-term recurrent convolutional networks for visual recognition and description,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 677–691, Apr. 2017.
- [11] Z. Zhou, J. Shin, L. Zhang, S. Gurudu, M. Gotway, and J. Liang, “Fine-tuning convolutional neural networks for biomedical image analysis: Actively and incrementally,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4761–4772.
- [12] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.
- [13] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, “Generative adversarial nets,” in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS’14, Montreal, Canada: MIT Press, 2014, pp. 2672–2680.
- [14] D. J. Rezende and S. Mohamed, “Variational inference with normalizing flows,” in *International Conference on Machine Learning (ICML)*, 2015, pp. 1530–1538.
- [15] D. P. Kingma, T. Salimans, and M. Welling, “Improving variational inference with inverse autoregressive flow,” *arXiv*, 2016.
- [16] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.
- [17] D. J. Rezende, S. Mohamed, and D. Wierstra, “Stochastic backpropagation and approximate inference in deep generative models,” in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ser. ICML’14, Beijing, China, 2014, II–1278–II–1286.
- [18] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, “An introduction to variational methods for graphical models,” *Mach. Learn.*, vol. 37, no. 2, pp. 183–233, Nov. 1999.
- [19] M. J. Wainwright and M. Jordan, “Graphical models, exponential families, and variational inference,” *Found. Trends Mach. Learn.*, vol. 1, pp. 1–305, 2008.
- [20] J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities,” in *Neurocomputing: Foundations of Research*. Cambridge, MA, USA: MIT Press, 1988, pp. 457–464, ISBN: 0262010976.
- [21] G. E. Hinton, S. Osindero, and Y. W. Teh, “A fast learning algorithm for deep belief nets,” *Neural Computation*, vol. 18, pp. 1527–1554, 2006.
- [22] H. B. Curry, “The method of steepest descent for non-linear minimization problems,” *Quarterly of Applied Mathematics*, vol. 2, pp. 258–261, 1944.

-
- [23] E. K. P. Chong and S. H. Żak, “Introduction to optimization,” *Engineering Optimization Theory and Practice*, 2019.
 - [24] A. Kumar, P. Sattigeri, and A. Balakrishnan, “Variational inference of disentangled latent concepts from unlabeled observations,” in *International Conference on Learning Representations (ICLR)*, 2018.
 - [25] E. Mathieu, T. Rainforth, N. Siddharth, and Y. W. Teh, “Disentangling disentanglement in variational autoencoders,” in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, 2019, pp. 4402–4412.
 - [26] I. Higgins, L. Matthey, A. Pal, *et al.*, “Beta-vae: Learning basic visual concepts with a constrained variational framework,” in *International Conference on Learning Representations (ICLR)*, 2017.
 - [27] C. P. Burgess, I. Higgins, A. Pal, *et al.*, “Understanding disentangling in β -vae,” *CoRR*, vol. abs/1804.03599, 2018.
 - [28] J. Antoran, U. Bhatt, T. Adel, A. Weller, and J. M. Hernández-Lobato, “Getting a {clue}: A method for explaining uncertainty estimates,” in *International Conference on Learning Representations*, 2021.
 - [29] D. Amodei, C. Olah, J. Steinhardt, P. F. Christiano, J. Schulman, and D. Mané, “Concrete problems in ai safety,” *ArXiv*, 2016.
 - [30] B. Lakshminarayanan, A. Pritzel, and C. Blundell, “Simple and scalable predictive uncertainty estimation using deep ensembles,” in *NIPS*, 2017.
 - [31] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, “Autoencoding beyond pixels using a learned similarity metric,” in *International conference on machine learning*, PMLR, 2016, pp. 1558–1566.
 - [32] J. An and S. Cho, “Variational autoencoder based anomaly detection using reconstruction probability,” 2015.
 - [33] X. Wang, Y. Du, S. Lin, P. Cui, Y. Shen, and Y. Yang, “Advae: A self-adversarial variational autoencoder with gaussian anomaly prior knowledge for anomaly detection,” *Knowledge-Based Systems*, vol. 190, 2020.
 - [34] D. Hendrycks, M. Mazeika, and T. Dietterich, “Deep anomaly detection with outlier exposure,” in *International Conference on Learning Representations*, 2019.
 - [35] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan, “Do deep generative models know what they don’t know?” In *International Conference on Learning Representations*, 2019. [Online]. Available: <https://openreview.net/forum?id=H1xwNhCcYm>.

- [36] Y. Gal and Z. Ghahramani, “Dropout as a bayesian approximation: Representing model uncertainty in deep learning,” *ArXiv*, vol. abs/1506.02142, 2016.
- [37] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
- [38] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with lstm,” *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.
- [39] Y. LeCun, “The mnist database of handwritten digits,” <http://yann.lecun.com/exdb/mnist/>, 1998. [Online]. Available: <https://ci.nii.ac.jp/naid/10027939599/en/>.
- [40] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms,” *arXiv preprint arXiv:1708.07747*, 2017.
- [41] Z. Liu, P. Luo, X. Wang, and X. Tang, “Deep Learning Face Attributes in the Wild,” in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015, pp. 3730–3738.
- [42] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, “Unsupervised real-time anomaly detection for streaming data,” *Neurocomputing*, vol. 262, pp. 134–147, 2017.
- [43] P. Tschandl, C. Rosendahl, and H. Kittler, “The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions,” *Scientific Data*, vol. 5, 2018.
- [44] L. Wasserman, *All of statistics : a concise course in statistical inference*. New York: Springer, 2010.
- [45] *Conditional independence — the backbone of bayesian networks*, <https://towardsdatascience.com/conditional-independence-the-backbone-of-bayesian-networks-85710f1b35b>, Accessed: 2022-01-18.
- [46] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.
- [47] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, “Variational inference: A review for statisticians,” *Journal of the American Statistical Association*, vol. 112, pp. 859–877, 2016.
- [48] M. E. Tipping and C. M. Bishop, “Probabilistic principal component analysis,” *Neural Computation*, 1999.
- [49] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, “Stochastic variational inference,” vol. 14, no. 1, 2013, ISSN: 1532-4435.
- [50] *Autograd mechanics*, <https://pytorch.org/docs/stable/notes/autograd.html>, Accessed: 2021-12-06.

-
- [51] *Introduction to gradients and automatic differentiation*, <https://www.tensorflow.org/guide/autodiff>, Accessed: 2021-12-06.
 - [52] P. Dayan, G. E. Hinton, R. M. Neal, and R. S. Zemel, “The helmholtz machine,” *Neural Comput.*, vol. 7, no. 5, pp. 889–904, Sep. 1995.
 - [53] S. J. Gershman and N. D. Goodman, “Amortized inference in probabilistic reasoning,” *Cognitive Science*, vol. 36, 2014.
 - [54] C. Zhang, J. B  t  page, H. Kjellstr  m, and S. Mandt, “Advances in variational inference,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, pp. 2008–2026, 2019.
 - [55] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.
 - [56] D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, Eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986.
 - [57] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>.
 - [58] H. E. Robbins, “A stochastic approximation method,” *Annals of Mathematical Statistics*, vol. 22, pp. 400–407, 2007.
 - [59] J. Kiefer and J. Wolfowitz, “Stochastic estimation of the maximum of a regression function,” *Annals of Mathematical Statistics*, vol. 23, pp. 462–466, 1952.
 - [60] Y. Burda, R. B. Grosse, and R. Salakhutdinov, “Importance weighted autoencoders,” in *4th International Conference on Learning Representations, ICLR*, Y. Bengio and Y. LeCun, Eds., 2016.
 - [61] J. M. Tomczak and M. Welling, “VAE with a VampPrior,” in *Artificial Intelligence*, 2018, pp. 1214–1223.
 - [62] T. Salimans, I. Goodfellow, W. Zaremba, *et al.*, “Improved techniques for training gans,” in *Advances in Neural Information Processing Systems*, vol. 29, 2016.
 - [63] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “Gans trained by a two time-scale update rule converge to a local nash equilibrium,” in *Neural Information Processing Systems (NeurIPS)*, 2017, pp. 6626–6637.
 - [64] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2016.
 - [65] L. Dinh, D. Krueger, and Y. Bengio, “Nice: Non-linear independent components estimation,” *CoRR*, vol. abs/1410.8516, 2014.

- [66] L. Dinh, J. Sohl-Dickstein, and S. Bengio, “Density estimation using real nvp,” in *International Conference on Learning Representations (ICLR)*, 2017.
- [67] I. J. Goodfellow, “NIPS 2016 tutorial: Generative adversarial networks,” 2017.
- [68] L. J. Ratliff, S. A. Burden, and S. S. Sastry, “Characterization and computation of local nash equilibria in continuous games,” in *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2013, pp. 917–924.
- [69] C. Ledig, L. Theis, F. Huszár, *et al.*, “Photo-realistic single image super-resolution using a generative adversarial network,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 105–114.
- [70] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros, “Generative visual manipulation on the natural image manifold,” in *Proceedings of European Conference on Computer Vision (ECCV)*, 2016.
- [71] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5967–5976.
- [72] *Common problems of gans*, <https://developers.google.com/machine-learning/gan/problems>, Accessed: 2022-02-23.
- [73] E. Richardson and Y. Weiss, “On gans and gmms,” in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, Montréal, Canada, 2018, pp. 5852–5863.
- [74] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein generative adversarial networks,” in *Proceedings of the 34th International Conference on Machine Learning*, D. Precup and Y. W. Teh, Eds., ser. Proceedings of Machine Learning Research, vol. 70, PMLR, Jun. 2017, pp. 214–223.
- [75] M. Arjovsky and L. Bottou, “Towards principled methods for training generative adversarial networks,” in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.
- [76] K. Roth, A. Lucchi, S. Nowozin, and T. Hofmann, “Stabilizing training of generative adversarial networks through regularization,” in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017, pp. 2018–2028.
- [77] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, “Unrolled generative adversarial networks,” in *5th International Conference on Learning Representations*

- tations, *ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.
- [78] C. Cremer, X. Li, and D. Duvenaud, “Inference suboptimality in variational autoencoders,” *arXiv*, 2018.
 - [79] L. Mescheder, S. Nowozin, and A. Geiger, “Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks,” in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, Sydney, NSW, Australia, 2017, pp. 2391–2400.
 - [80] F. Huszár, “Variational inference using implicit distributions,” *arXiv preprint arXiv:1702.08235*, 2017.
 - [81] E. T. Nalisnick and P. Smyth, “Stick-breaking variational autoencoders,” in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.
 - [82] P. Goyal, Z. Hu, X. Liang, C. Wang, E. P. Xing, and C. Mellon, “Nonparametric variational auto-encoders for hierarchical representation learning,” in *IEEE International Conference on Computer Vision, ICCV*, 2017.
 - [83] N. Dilokthanakul, P. A. Mediano, M. Garnelo, *et al.*, “Deep unsupervised clustering with gaussian mixture variational autoencoders,” *arXiv preprint arXiv:1611.02648*, 2016.
 - [84] M. J. Johnson, D. Duvenaud, A. B. Wiltschko, R. P. Adams, and S. R. Datta, “Composing graphical models with neural networks for structured representations and fast inference,” in *Neural Information Processing Systems (NeurIPS)*, 2016, pp. 2946–2954.
 - [85] X. Chen, D. P. Kingma, T. Salimans, *et al.*, “Variational lossy autoencoder,” in *International Conference on Learning Representations (ICLR)*, 2017.
 - [86] A. Razavi, A. van den Oord, and O. Vinyals, “Generating Diverse High-Fidelity Images with VQ-VAE-2,” in *Neural Information Processing Systems (NeurIPS)*, 2019, pp. 14 837–14 847.
 - [87] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, “Deep variational information bottleneck,” in *International Conference on Learning Representations (ICLR)*, 2017.
 - [88] A. A. Alemi, B. Poole, I. Fischer, J. V. Dillon, R. A. Saurous, and K. Murphy, “Fixing a broken ELBO,” in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018, pp. 159–168.
 - [89] J. Aneja, A. Schwing, J. Kautz, and A. Vahdat, “A contrastive learning approach for training variational autoencoder priors,” *Advances in Neural Information Processing Systems*, vol. 34, 2021.

- [90] I. O. Tolstikhin, O. Bousquet, S. Gelly, and B. Schölkopf, “Wasserstein auto-encoders,” 2018.
- [91] D. M. Blei, M. I. Jordan, *et al.*, “Variational inference for Dirichlet process mixtures,” *Bayesian analysis*, 2006.
- [92] S. Lin, S. J. Roberts, N. Trigoni, and R. Clark, “Balancing Reconstruction Quality and Regularisation in ElbO for Vaes,” *arXiv*, 2019.
- [93] A. Klushyn, N. Chen, R. Kurle, B. Cseke, and P. van der Smagt, “Learning Hierarchical Priors in Vaes,” in *Neural Information Processing Systems (NeurIPS)*, 2019, pp. 2866–2875.
- [94] H. Kim and A. Mnih, “Disentangling by factorising,” in *International Conference on Machine Learning (ICML)*, PMLR, 2018, pp. 2649–2658.
- [95] D. P. Kingma and P. Dhariwal, “Glow: Generative Flow with Invertible 1x1 Convolutions,” in *Neural Information Processing Systems (NeurIPS)*, 2018.
- [96] A. Radford, L. Metz, and S. Chintala, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.
- [97] D. Berthelot, C. Raffel, A. Roy, and I. Goodfellow, “Understanding and Improving Interpolation in Autoencoders via an Adversarial Regularizer,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.
- [98] D. P. Kingma and J. Ba, “Adam: A method for Stochastic Optimization,” in *International Conference on Learning Representations, ICLR*, 2015.
- [99] S. H. Strogatz, *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. CRC press, 2018.
- [100] *Stability of fixed point of a dynamical system*, <https://adipandas.github.io/posts/2020/02/stable-unstable-fixed-point/>, Accessed: 2022-05-08.
- [101] M. Goldstein and S. Uchida, “A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data,” *PloS one*, 2016.
- [102] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.
- [103] A. K. Ghosh and A. Schwartzbard, “A study in using neural networks for anomaly and misuse detection,” in *USENIX security symposium*, vol. 99, 1999, p. 12.
- [104] T. Wen and R. Keyes, “Time series anomaly detection using convolutional neural networks and transfer learning,” *IJCAI’19 Workshops*, 2019.

- [105] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, “Long short term memory networks for anomaly detection in time series,” in *Proceedings*, Presses universitaires de Louvain, 2015, p. 89.
- [106] S. Suh, D. Chae, H.-G. Kang, and S. Choi, “Echo-state conditional variational autoencoder for anomaly detection,” Jul. 2016, pp. 1015–1022. DOI: 10.1109/IJCNN.2016.7727309.
- [107] Y. Kawachi, Y. Koizumi, and N. Harada, “Complementary set variational autoencoder for supervised anomaly detection,” *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2366–2370, 2018.
- [108] H. Xu, Y. Feng, J. Chen, *et al.*, “Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications,” *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18*, 2018.
- [109] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, “Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding,” *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '18*, 2018. DOI: 10.1145/3219819.3219845. [Online]. Available: <http://dx.doi.org/10.1145/3219819.3219845>.
- [110] P. J. Werbos, “Generalization of backpropagation with application to a recurrent gas market model,” *Neural Networks*, vol. 1, no. 4, pp. 339–356, 1988.
- [111] S. Wang, R. Clark, H. Wen, and N. Trigoni, “Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 2043–2050.
- [112] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni, “Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem,” in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.
- [113] Y. S. Chong and Y. H. Tay, “Abnormal event detection in videos using spatiotemporal autoencoder,” in *International Symposium on Neural Networks*, Springer, 2017, pp. 189–196.
- [114] B. Pincombea, “Anomaly detection in time series of graphs using arma processes,” 2007.
- [115] D. Hendrycks and K. Gimpel, “A baseline for detecting misclassified and out-of-distribution examples in neural networks,” *Proceedings of International Conference on Learning Representations*, 2017.

- [116] S. Liang, Y. Li, and R. Srikant, “Enhancing the reliability of out-of-distribution image detection in neural networks,” 2018.
- [117] L. Zhang, M. Goldstein, and R. Ranganath, “Understanding failures in out-of-distribution detection with deep generative models,” in *International Conference on Machine Learning*, PMLR, 2021.
- [118] Y. Ovadia, E. Fertig, J. Ren, *et al.*, “Can you trust your model’s uncertainty? evaluating predictive uncertainty under dataset shift,” in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32, Curran Associates, Inc., 2019. [Online]. Available: <https://proceedings.neurips.cc/paper/2019/file/8558cb408c1d76621371888657d2eb1d-Paper.pdf>.
- [119] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” in *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2013.
- [120] A. P. Dawid, “The well-calibrated bayesian,” *Journal of the American Statistical Association*, vol. 77, pp. 605–610, 1982.
- [121] M. H. Degroot and S. E. Fienberg, “The comparison and evaluation of forecasters,” *The Statistician*, vol. 32, pp. 12–22, 1983.
- [122] D. J. MacKay, “A practical bayesian framework for backpropagation networks,” *Neural computation*, vol. 4, no. 3, pp. 448–472, 1992.
- [123] R. Neal, “Bayesian learning via stochastic dynamics,” *Advances in neural information processing systems*, vol. 5, 1992.
- [124] R. M. Neal, *Bayesian learning for neural networks*. Springer Science & Business Media, 2012, vol. 118.
- [125] M. Welling and Y. W. Teh, “Bayesian learning via stochastic gradient langevin dynamics,” in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011.
- [126] F. Wenzel, K. Roth, B. S. Veeling, *et al.*, “How good is the bayes posterior in deep neural networks really?” *arXiv preprint arXiv:2002.02405*, 2020.
- [127] R. M. Neal *et al.*, “Mcmc using hamiltonian dynamics,” *Handbook of markov chain monte carlo*, vol. 2, 2011.
- [128] D. J. C. Mackay, “Bayesian methods for adaptive models,” 1992.
- [129] H. Ritter, A. Botev, and D. Barber, “A scalable laplace approximation for neural networks,” in *6th International Conference on Learning Representations (ICLR)*, 2018.

-
- [130] A. Graves, “Practical variational inference for neural networks,” in *NIPS*, 2011.
 - [131] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, “Weight uncertainty in neural networks,” *ArXiv*, vol. abs/1505.05424, 2015.
 - [132] M. Dusenberry, G. Jerfel, Y. Wen, *et al.*, “Efficient and scalable bayesian neural nets with rank-1 factors,” in *International conference on machine learning*, 2020, pp. 2782–2792.
 - [133] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, “On calibration of modern neural networks,” *ArXiv*, vol. abs/1706.04599, 2017.
 - [134] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation,” 1986.
 - [135] Y. A. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” 1998.
 - [136] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.
 - [137] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” 2015.
 - [138] J. Kos, I. Fischer, and D. Song, “Adversarial examples for generative models,” in *2018 IEEE security and privacy workshops (spw)*, IEEE, 2018, pp. 36–42.
 - [139] G. Gondim-Ribeiro, P. Tabacof, and E. Valle, “Adversarial attacks on variational autoencoders,” *arXiv preprint arXiv:1806.04646*, 2018.
 - [140] A. Vahdat and J. Kautz, “Nvae: A deep hierarchical variational autoencoder,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 19 667–19 679, 2020.
 - [141] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, “Active learning with statistical models,” *Journal of artificial intelligence research*, vol. 4, pp. 129–145, 1996.
 - [142] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
 - [143] K. B. Petersen and M. S. Pedersen, *The matrix cookbook*, Nov. 2012.

A

Appendix A

A.1 Matrix derivatives

$$\frac{\partial \mathbf{x}^T \mathbf{a}}{\partial \mathbf{x}} = \frac{\partial \mathbf{a}^T \mathbf{x}}{\partial \mathbf{x}} = \mathbf{a} \quad (\text{A.1})$$

$$\frac{\partial \mathbf{x}^T \mathbf{A}}{\partial \mathbf{x}} = (\mathbf{A} + \mathbf{A}^T) \mathbf{x} \quad (\text{A.2})$$

$$\frac{\partial \log |\mathbf{X}|}{\partial \mathbf{X}} = (\mathbf{X}^{-1})^T = (\mathbf{X}^T)^{-1} = \mathbf{X}^{-T} \quad (\text{A.3})$$

$$\frac{\partial \mathbf{a}^T \mathbf{X}^{-1} \mathbf{b}}{\partial \mathbf{X}} = -\mathbf{X}^{-T} \mathbf{a} \mathbf{b}^T \mathbf{X}^{-T} \quad (\text{A.4})$$

A.2 Expectation for vectors and matrices

Assuming $\mathbf{u}$ is a random variable with mean $\mathbf{c}$ and covariance matrix Σ , we have the following identities [143]:

$$\mathbb{E}_{p(\mathbf{u})} [\mathbf{A} \mathbf{u}] = \mathbf{A} \mathbf{c} \quad (\text{A.5})$$

$$\mathbb{E}_{p(\mathbf{u})} [\mathbf{u}^T \mathbf{A} \mathbf{u}] = \text{Tr}(\mathbf{A} \Sigma) + \mathbf{c}^T \mathbf{A} \mathbf{c} \quad (\text{A.6})$$

$$\mathbb{E}_{p(\mathbf{u})} [\mathbf{u}^T \mathbf{u}] = \text{Tr}(\Sigma) + \mathbf{c}^T \mathbf{c}. \quad (\text{A.7})$$

A.3 Variational inference derivations

We show below how the evidence lower bound objective is derived from the original objective of $D_{\text{KL}}(q_\phi(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x}))$:

$$D_{\text{KL}}(q_\phi(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x})) = \mathbb{E}_{q_\phi(\mathbf{z})} \left[\log \frac{q_\phi(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})} \right] \quad (\text{A.8})$$

$$= \mathbb{E}_{q_\phi(\mathbf{z})} [\log q_\phi(\mathbf{z})] - \mathbb{E}_{q_\phi(\mathbf{z})} [\log p(\mathbf{z}|\mathbf{x})] \quad (\text{A.9})$$

$$= \mathbb{E}_{q_\phi(\mathbf{z})} [\log q_\phi(\mathbf{z})] - \mathbb{E}_{q_\phi(\mathbf{z})} \left[\log \frac{p_\theta(\mathbf{z}, \mathbf{x})}{p(\mathbf{x})} \right] \quad (\text{A.10})$$

$$= \mathbb{E}_{q_\phi(\mathbf{z})} [\log q_\phi(\mathbf{z})] - \mathbb{E}_{q_\phi(\mathbf{z})} [\log p_\theta(\mathbf{z}, \mathbf{x})] + \mathbb{E}_{q_\phi(\mathbf{z})} [\log p(\mathbf{x})] \quad (\text{A.11})$$

$$= \mathbb{E}_{q_\phi(\mathbf{z})} [\log q_\phi(\mathbf{z})] - \mathbb{E}_{q_\phi(\mathbf{z})} [\log p_\theta(\mathbf{z}, \mathbf{x})] + \log p(\mathbf{x}). \quad (\text{A.12})$$

Here we show how the mean-field assumption simplifies the computation of the

entropy term $\mathbb{E}_{q_\phi(\mathbf{z})}[\log q_\phi(\mathbf{z})]$ in ELBO.

$$\mathbb{E}_{q_\phi(\mathbf{z})}[\log q_\phi(\mathbf{z})] = \int q_\phi(\mathbf{z}) \log q_\phi(\mathbf{z}) d\mathbf{z} \quad (\text{A.13})$$

$$= \int \left(\prod_{j=1}^M q_{\phi_j}(\mathbf{z}^{(j)}) d\mathbf{z}^{(j)} \right) \left(\log \prod_{i=1}^M q_{\phi_i}(\mathbf{z}^{(i)}) \right) \quad (\text{A.14})$$

$$= \int \left(\prod_{j=1}^M q_{\phi_j}(\mathbf{z}^{(j)}) d\mathbf{z}^{(j)} \right) \left(\sum_{i=1}^M \log q_{\phi_i}(\mathbf{z}^{(i)}) \right) \quad (\text{A.15})$$

$$= \int \left(\prod_{j=1}^M q_{\phi_j}(\mathbf{z}^{(j)}) d\mathbf{z}^{(j)} \right) \left(\log q_{\phi_1}(\mathbf{z}^{(1)}) + \sum_{i=2}^M \log q_{\phi_i}(\mathbf{z}^{(i)}) \right) \quad (\text{A.16})$$

$$= \int \left(\prod_{j=2}^M q_{\phi_j}(\mathbf{z}^{(j)}) d\mathbf{z}^{(j)} q_{\phi_1}(\mathbf{z}^{(1)}) d\mathbf{z}^{(1)} \right) \left(\log q_{\phi_1}(\mathbf{z}^{(1)}) + \sum_{i=2}^M \log q_{\phi_i}(\mathbf{z}^{(i)}) \right) \quad (\text{A.17})$$

$$= \int_{\mathbf{z}_{j \geq 2}} \left(\prod_{j=2}^M q_{\phi_j}(\mathbf{z}^{(j)}) d\mathbf{z}^{(j)} \right) \left(\int_{\mathbf{z}^{(1)}} q_{\phi_1}(\mathbf{z}^{(1)}) d\mathbf{z}^{(1)} \left(\log q_{\phi_1}(\mathbf{z}^{(1)}) + \sum_{i=2}^M \log q_{\phi_i}(\mathbf{z}^{(i)}) \right) \right) \quad (\text{A.18})$$

$$= \int_{\mathbf{z}_{j \geq 2}} \left(\prod_{j=2}^M q_{\phi_j}(\mathbf{z}^{(j)}) d\mathbf{z}^{(j)} \right) \left(\int_{\mathbf{z}^{(1)}} q_{\phi_1}(\mathbf{z}^{(1)}) d\mathbf{z}^{(1)} \log q_{\phi_1}(\mathbf{z}^{(1)}) + \sum_{i=2}^M \log q_{\phi_i}(\mathbf{z}^{(i)}) \right) \quad (\text{A.19})$$

$$= \int_{\mathbf{z}^{(1)}} q_{\phi_1}(\mathbf{z}^{(1)}) d\mathbf{z}^{(1)} \log q_{\phi_1}(\mathbf{z}^{(1)}) + \int_{\mathbf{z}_{j \geq 2}} \left(\prod_{j=2}^M q_{\phi_j}(\mathbf{z}^{(j)}) d\mathbf{z}^{(j)} \right) \left(\sum_{i=2}^M \log q_{\phi_i}(\mathbf{z}^{(i)}) \right) \quad (\text{A.20})$$

$$= \mathbb{E}_{q_{\phi_1}(\mathbf{z}^{(1)})}[\log q_{\phi_1}(\mathbf{z}^{(1)})] + \int_{\mathbf{z}_{j \geq 2}} \left(\prod_{j=2}^M q_{\phi_j}(\mathbf{z}^{(j)}) d\mathbf{z}^{(j)} \right) \left(\sum_{i=2}^M \log q_{\phi_i}(\mathbf{z}^{(i)}) \right) \quad (\text{A.21})$$

$$= \sum_{j=1}^M \mathbb{E}_{q_{\phi_j}(\mathbf{z}^{(j)})}[\log q_{\phi_j}(\mathbf{z}^{(j)})]. \quad (\text{A.22})$$

Equations (A.13) - (A.21) show how the expectation of the first latent variable $\mathbf{z}_1$ can be separated from the rest terms. Applying this same trick for all M latent variables, we derive the final expression in Equation (A.22).

Here we show how the i.i.d assumption can be applied to simplify the log of the

joint probability $\log p_\theta(\mathbf{z}, \mathbf{x})$.

$$\log p_\theta(\mathbf{z}, \mathbf{x}) = \log \prod_{i=1}^N p_\theta(\mathbf{z}_i, \mathbf{x}_i) \quad (\text{A.23})$$

$$= \log \prod_{i=1}^N \left(p_\theta(\mathbf{x}_i | \mathbf{z}_i) p(\mathbf{z}_i) \right) \quad (\text{A.24})$$

$$= \sum_{i=1}^N \log \left(p_\theta(\mathbf{x}_i | \mathbf{z}_i) p(\mathbf{z}_i) \right) \quad (\text{A.25})$$

$$= \sum_{i=1}^N \left(\log p_\theta(\mathbf{x}_i | \mathbf{z}_i) + \log p(\mathbf{z}_i) \right) \quad (\text{A.26})$$

$$= \sum_{i=1}^N \log p_\theta(\mathbf{x}_i | \mathbf{z}_i) + \sum_{i=1}^N \log p(\mathbf{z}_i) \quad (\text{A.27})$$

Here we show the mean-field assumption can be used to simplify the expectation computation in the following term:

$$\sum_{i=1}^N \mathbb{E}_{q_\phi(\mathbf{z})} \left[\log p_\theta(\mathbf{x}_i | \mathbf{z}_i) \right] = \sum_{i=1}^N \mathbb{E}_{\prod_{j=1}^N q_\phi(\mathbf{z}_j)} \left[\log p_\theta(\mathbf{x}_i | \mathbf{z}_i) \right] \quad (\text{A.28})$$

$$= \sum_{i=1}^N \mathbb{E}_{q_\phi(\mathbf{z}_i) \prod_{j=1, j \neq i}^N q_\phi(\mathbf{z}_j)} \left[\log p_\theta(\mathbf{x}_i | \mathbf{z}_i) \right] \quad (\text{A.29})$$

$$= \sum_{i=1}^N \mathbb{E}_{q_\phi(\mathbf{z}_i)} \mathbb{E}_{\prod_{j=1, j \neq i}^N q_\phi(\mathbf{z}_j)} \left[\log p_\theta(\mathbf{x}_i | \mathbf{z}_i) \right] \quad (\text{A.30})$$

$$= \left(\sum_{i=1}^N \mathbb{E}_{q_\phi(\mathbf{z}_i)} \left[\log p_\theta(\mathbf{x}_i | \mathbf{z}_i) \right] \right) \left(\mathbb{E}_{\prod_{j=1, j \neq i}^N q_\phi(\mathbf{z}_j)} \right) \quad (\text{A.31})$$

$$= \sum_{i=1}^N \mathbb{E}_{q_\phi(\mathbf{z}_i)} \left[\log p_\theta(\mathbf{x}_i | \mathbf{z}_i) \right]. \quad (\text{A.32})$$

A.4 KL divergence between two Gaussians

Here we derive the KL divergence $D_{\text{KL}}(p(\mathbf{x}) \| q(\mathbf{x}))$ between two multivariate Gaussian distributions $p(\mathbf{x})$ and $q(\mathbf{x})$. Recall that multivariate Gaussian distributions have a parametric form introduced in Equation (2.11). In our case, we have:

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p) = \frac{1}{(2\pi)^{\frac{D}{2}}} \frac{1}{|\boldsymbol{\Sigma}_p|^{\frac{1}{2}}} \exp \left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_p)^T \boldsymbol{\Sigma}_p^{-1} (\mathbf{x} - \boldsymbol{\mu}_p) \right), \quad (\text{A.33})$$

$$q(\mathbf{x}) = \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q) = \frac{1}{(2\pi)^{\frac{D}{2}}} \frac{1}{|\boldsymbol{\Sigma}_q|^{\frac{1}{2}}} \exp \left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_q)^T \boldsymbol{\Sigma}_q^{-1} (\mathbf{x} - \boldsymbol{\mu}_q) \right). \quad (\text{A.34})$$

Also recall that the KL divergence is defined in Equation (2.35). Therefore, we have:

$$D_{\text{KL}}(p(\mathbf{x}) \parallel q(\mathbf{x})) = \mathbb{E}_p \left[\log \frac{p}{q} \right] = \mathbb{E}_p[\log p] - \mathbb{E}_p[\log q]. \quad (\text{A.35})$$

In the above equation, we have omitted the variable $\mathbf{x}$ for brevity and we will do so for the rest of derivation wherever appropriate.

Substituting the probability distributions for p and q from Equation (A.33) and (A.34) to (A.35), we have:

$$\mathbb{E}_p[\log p] = \mathbb{E}_p \left[\log \frac{1}{(2\pi)^{\frac{D}{2}}} \frac{1}{|\Sigma_p|^{\frac{1}{2}}} - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_p)^T \Sigma_p^{-1}(\mathbf{x} - \boldsymbol{\mu}_p) \right] \quad (\text{A.36})$$

$$= -\frac{D}{2} \log 2\pi - \frac{1}{2} \log |\Sigma_p| - \frac{1}{2} \mathbb{E}_p \left[(\mathbf{x} - \boldsymbol{\mu}_p)^T \Sigma_p^{-1}(\mathbf{x} - \boldsymbol{\mu}_p) \right]. \quad (\text{A.37})$$

Similarly, for the second expectation in Equation (A.35), we have:

$$\mathbb{E}_p[\log q] = -\frac{D}{2} \log 2\pi - \frac{1}{2} \log |\Sigma_q| - \frac{1}{2} \mathbb{E}_p \left[(\mathbf{x} - \boldsymbol{\mu}_q)^T \Sigma_q^{-1}(\mathbf{x} - \boldsymbol{\mu}_q) \right]. \quad (\text{A.38})$$

Bringing the two expressions together, we have the expression for the overall KL divergence as:

$$D_{\text{KL}}(p(\mathbf{x}) \parallel q(\mathbf{x})) = \frac{1}{2} \log \frac{|\Sigma_q|}{|\Sigma_p|} - \frac{1}{2} \mathbb{E}_p \left[(\mathbf{x} - \boldsymbol{\mu}_p)^T \Sigma_p^{-1}(\mathbf{x} - \boldsymbol{\mu}_p) \right] \quad (\text{A.39})$$

$$+ \frac{1}{2} \mathbb{E}_p \left[(\mathbf{x} - \boldsymbol{\mu}_q)^T \Sigma_q^{-1}(\mathbf{x} - \boldsymbol{\mu}_q) \right]. \quad (\text{A.40})$$

We can simplify the expectation terms using the Identity (A.6), giving:

$$\mathbb{E}_p \left[(\mathbf{x} - \boldsymbol{\mu}_p)^T \Sigma_p^{-1}(\mathbf{x} - \boldsymbol{\mu}_p) \right] = (\boldsymbol{\mu}_p - \boldsymbol{\mu}_p)^T \Sigma_p^{-1}(\boldsymbol{\mu}_p - \boldsymbol{\mu}_p) + \text{Tr}(\Sigma_p^{-1} \Sigma_p) \quad (\text{A.41})$$

$$= D. \quad (\text{A.42})$$

$$\mathbb{E}_p \left[(\mathbf{x} - \boldsymbol{\mu}_q)^T \Sigma_q^{-1}(\mathbf{x} - \boldsymbol{\mu}_q) \right] = (\boldsymbol{\mu}_p - \boldsymbol{\mu}_q)^T \Sigma_q^{-1}(\boldsymbol{\mu}_p - \boldsymbol{\mu}_q) + \text{Tr}(\Sigma_q^{-1} \Sigma_p). \quad (\text{A.43})$$

Therefore, we have the final expression for the KL divergence as:

$$D_{\text{KL}}(p(\mathbf{x}) \parallel q(\mathbf{x})) = \frac{1}{2} \left[\log \frac{|\Sigma_q|}{|\Sigma_p|} - D + (\boldsymbol{\mu}_p - \boldsymbol{\mu}_q)^T \Sigma_q^{-1}(\boldsymbol{\mu}_p - \boldsymbol{\mu}_q) + \text{Tr}(\Sigma_q^{-1} \Sigma_p) \right]. \quad (\text{A.44})$$

If $q(\mathbf{x})$ is zero-mean, unit-variance Gaussian, i.e. $\mathcal{N}(\mathbf{0}, \mathbf{I})$, then the KL divergence

can be simplified to:

$$D_{\text{KL}}(p(\mathbf{x}) \parallel q(\mathbf{x})) = \frac{1}{2} \left[\boldsymbol{\mu}_p^{\text{T}} \boldsymbol{\mu}_p + \text{Tr}(\boldsymbol{\Sigma}_p) - D - \log |\boldsymbol{\Sigma}_p| \right]. \quad (\text{A.45})$$

A.5 Likelihood for a VAE model

Here we derive the log likelihood $\log p(\mathbf{x}^*)$ on a test data sample $\mathbf{x}^*$ for a VAE model, where the joint probability $p_{\phi, \theta}(\mathbf{x}, \mathbf{z}) = p_{\theta}(\mathbf{x}|\mathbf{z})p(\mathbf{z}) = q_{\phi}(\mathbf{z}|\mathbf{x})p(\mathbf{x})$ is estimated through two parametric models $p_{\theta}(\mathbf{x}|\mathbf{z})$ and $q_{\phi}(\mathbf{z}|\mathbf{x})$.

$$p(\mathbf{x}^*) = \frac{p(\mathbf{x}^*)}{\int q_{\phi}(\mathbf{z}|\mathbf{x}) \, d\mathbf{z}} \quad (\text{A.46})$$

$$= \frac{p(\mathbf{x}^*)}{\int q_{\phi}(\mathbf{z}|\mathbf{x}) \frac{p_{\phi, \theta}(\mathbf{x}, \mathbf{z})}{p_{\phi, \theta}(\mathbf{x}, \mathbf{z})} \, d\mathbf{z}} \quad (\text{A.47})$$

$$= \int \frac{p(\mathbf{x}^*)}{p_{\phi, \theta}(\mathbf{x}, \mathbf{z})} \frac{p_{\phi, \theta}(\mathbf{x}, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} \, d\mathbf{z} \quad (\text{A.48})$$

$$= \int q_{\phi}(\mathbf{z}|\mathbf{x}) \frac{p_{\phi, \theta}(\mathbf{x}, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} \, d\mathbf{z} \quad (\text{A.49})$$

$$= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left[\frac{p_{\phi, \theta}(\mathbf{x}, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} \right]. \quad (\text{A.50})$$

B

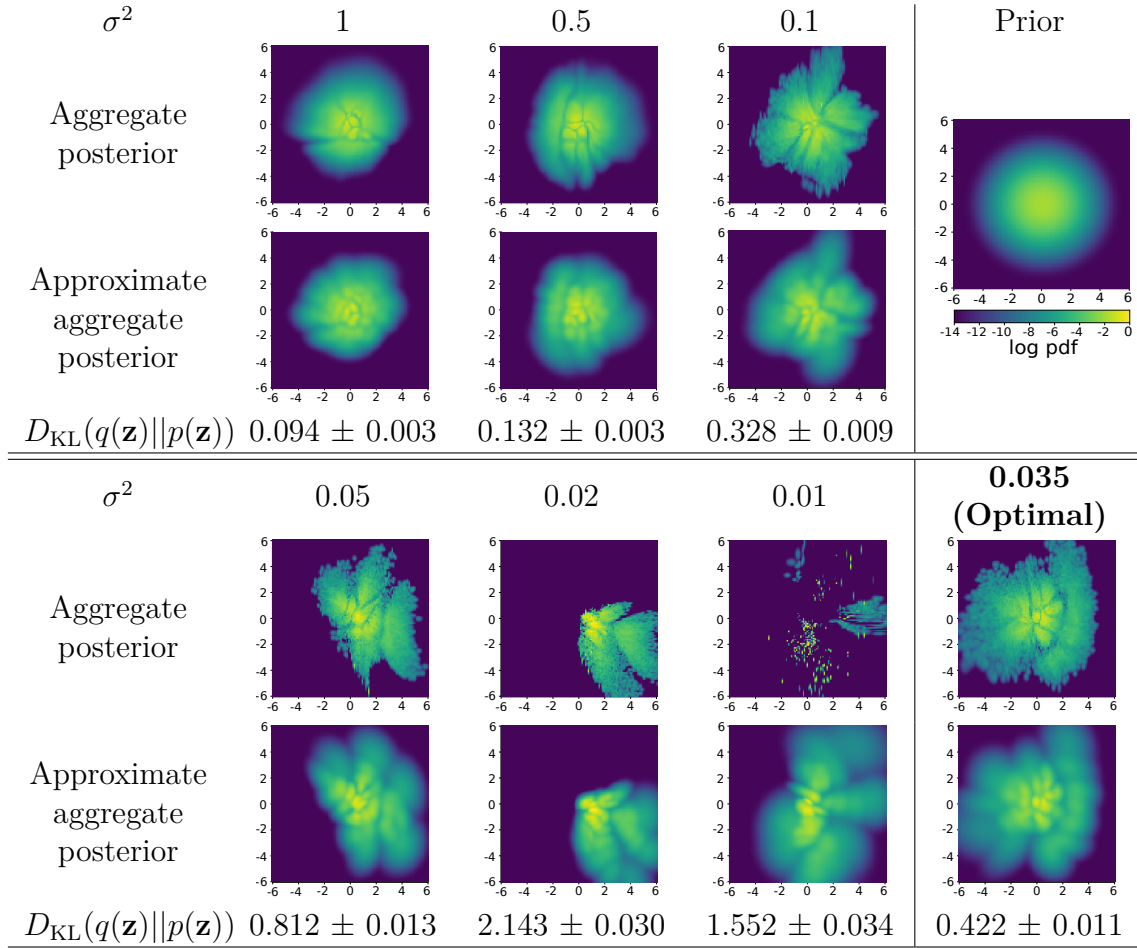
Appendix B

B.1 Additional results for Section 5.1

B.1.1 More Visualisation of MNIST 2D Aggregate Posteriors

Visualisation of the marginal latent distribution for MNIST dataset with a 2D latent space is very helpful to understand the impact of different variance values on the learnt marginal distributions. We showed plots for 4 different variance values in the main texts. Here we list more variance values to illustrate more clearly the transition of the learnt marginal posterior (or aggregate posterior - the data distribution in the latent space) from close to prior distribution to quite different when σ^2 decreases from 1 to very small value. The optimal σ^2 (0.035, found by maximising ELBO) gives an auto-encoding procedure that maintains the best balance between reconstruction error and prior regularisation.

Table B.1: Comparison of the aggregate posterior (data distribution in the latent space) and its Gaussian mixture approximation (32 mixtures) for models learnt under different σ^2 values for MNIST dataset. $D_{\text{KL}}(\hat{q}(\mathbf{z})||p(\mathbf{z}))$ denotes the gap between the approximate aggregate posterior and the prior distribution and is evaluated by Monte Carlo estimation of 10k samples (10 runs). The data distribution is clearly different from the prior distribution: low density regions exist in areas near zero code whereas prior does not contain such low density regions. Hence, naively using the prior distribution to represent the data distribution is sub-optimal. Our GMM approximation can capture data distribution for almost all σ^2 values except very small σ^2 . However, we don't have to worry about it, as our α -VAE learning algorithm can successfully avoid the small σ^2 .



B.2 Uncertainty Estimation

We can conveniently use the learnt variance prediction to estimate the uncertainty in the generated or reconstructed samples. We showed some examples in the main text and here we give more results for all datasets in Figure B.1-B.6.



Figure B.1: Generated samples (1st row) for MNIST dataset given by our learning method α -VAE and the corresponding pixel-wise variances (2nd row), which can be used to indicate regions of high uncertainty in the generated samples. The generated samples look realistic and the high variances correspond to areas where other possibilities of generation can occur.

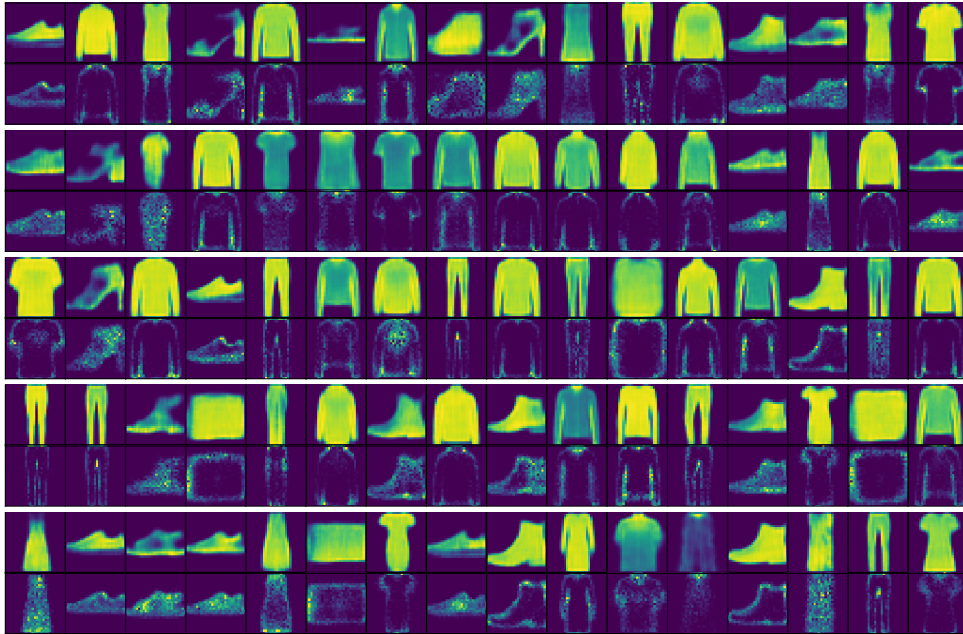


Figure B.2: Generated samples (1st row) for Fashion MNIST dataset given by our learning method α -VAE and the corresponding pixel-wise variances (2nd row), which can be used to indicate regions of high uncertainty in the generated samples. The generated samples look realistic and the high variances correspond to areas where other possibilities of generation can occur.



Figure B.3: Generated samples (1st row) for CelebA dataset given by our learning method α -VAE and the corresponding pixel-wise variances (2nd row), which can be used to indicate regions of high uncertainty in the generated samples. Note that the high variance regions often correspond to areas of eyes, mouth, hair and background where high variation in the dataset exists. This indicates that the model is aware that there are multiple valid options to generate those regions and the current generation is only one of such options.

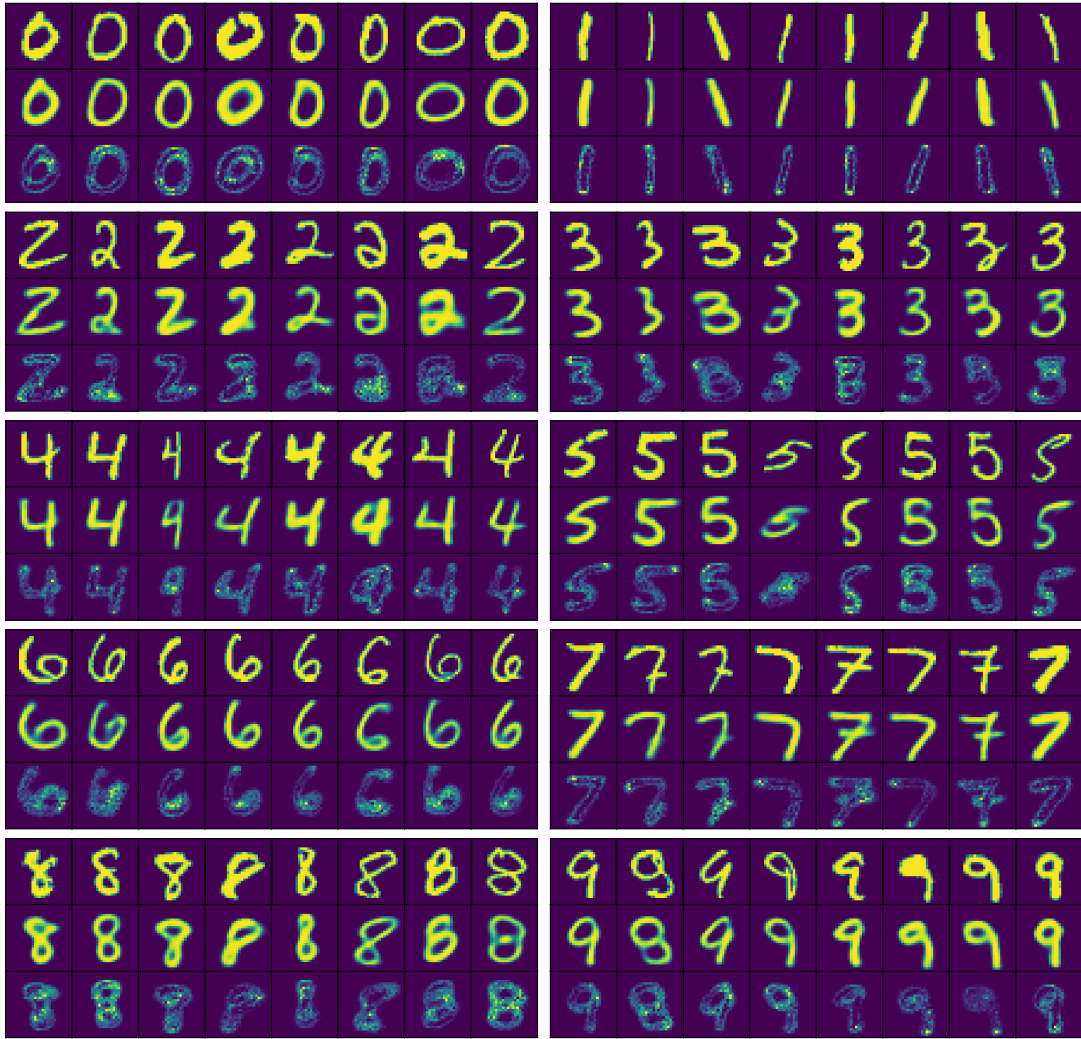


Figure B.4: Original and reconstructed samples of MNIST dataset (top and middle row for each digit block) and the corresponding variance values (bottom row) indicating regions of high uncertainty in the reconstruction. Note the high variance regions correspond very well to the areas where the reconstruction is different from the original image. This indicates that the model is aware the imperfection in its reconstruction and the location of such imperfection.

B.3 Additional results for Section 5.2

B.3.1 Model Architectures and Training Parameters

B.3.1.1 MNIST and Fashion MNIST

We use CNN architectures for both MNIST and Fashion MNIST encoder and decoder network (with additional depth to space operations for the decoder), as detailed below. We use the exactly same network for both datasets.

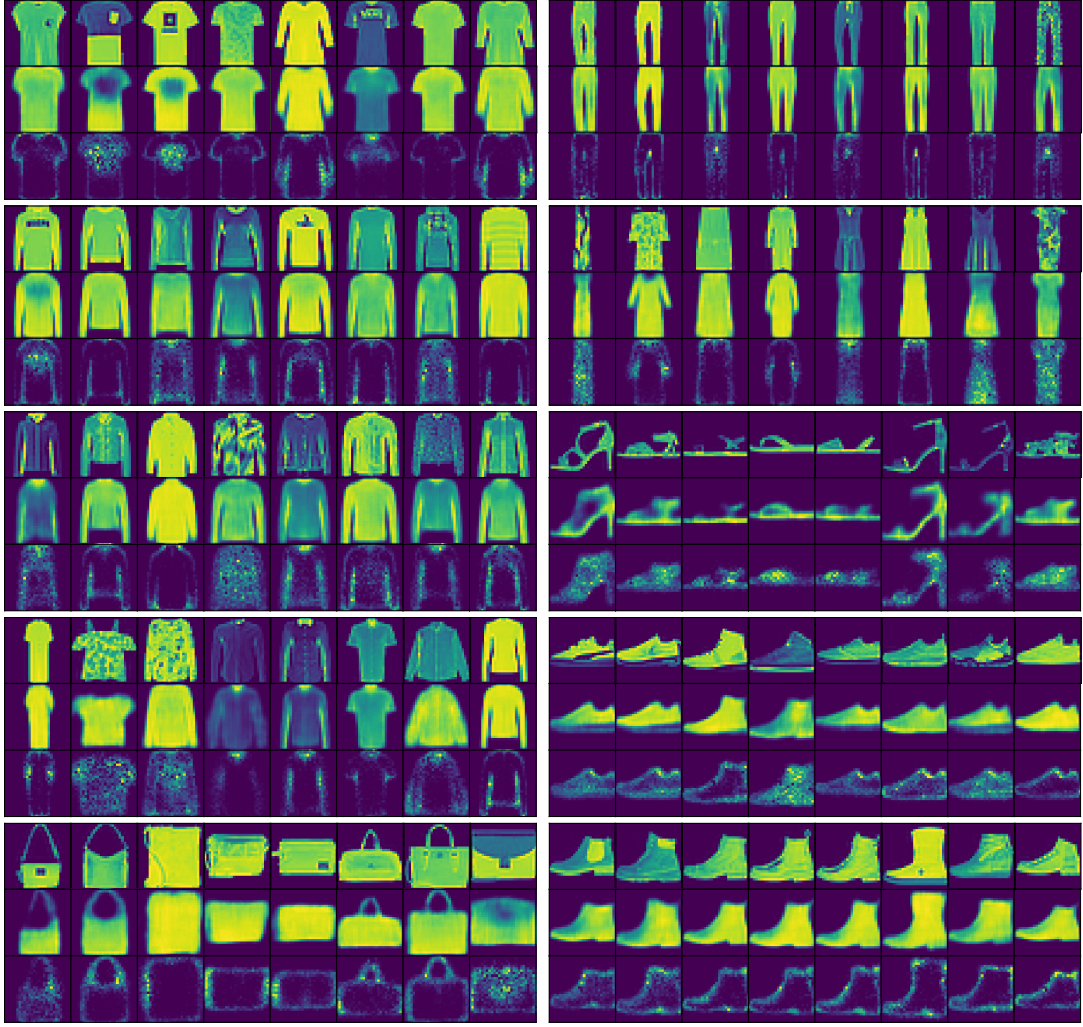


Figure B.5: Original and reconstructed samples of Fashion MNIST dataset (top and middle row for each digit block) and the corresponding variance values (bottom row) indicating regions of high uncertainty in the reconstruction. Note the high variance regions correspond very well to the areas where the reconstruction is different from the original image. This indicates that the model is aware the imperfection in its reconstruction and the location of such imperfection.

Encoder network:

$$\begin{aligned}
 \mathbf{x} \in \mathcal{R}^{28 \times 28 \times 1} &\rightarrow \text{Symmetric padding} \rightarrow \mathcal{R}^{32 \times 32 \times 1} \rightarrow \text{Conv}_{16}^{3 \times 3} (\text{stride} = 2) \rightarrow \text{ReLU} \\
 &\rightarrow \text{Conv}_{64}^{3 \times 3} (\text{stride} = 2) \rightarrow \text{ReLU} \\
 &\rightarrow \text{Conv}_{256}^{3 \times 3} (\text{stride} = 2) \rightarrow \text{ReLU} \\
 &\rightarrow \text{FC}_{64} \rightarrow \text{FC}_{d_z} \Rightarrow \mu \in \mathcal{R}^{d_z} \\
 &\quad \searrow \\
 &\quad \text{FC}_{d_z} \rightarrow \text{ReLU} \Rightarrow \sigma \in \mathcal{R}^{d_z}
 \end{aligned}$$



Figure B.6: Original and reconstructed samples of CelebA dataset (top and middle row for each block) and the corresponding variance values (bottom row) indicating regions of high uncertainty in the reconstruction. Note the high variance regions correspond very well to the areas where the reconstruction is different from the original image. This indicates that the model is aware the imperfection in its reconstruction and the location of such imperfection.

Decoder network:

$$\begin{aligned}
 \mathbf{z} &= \mu + \epsilon \odot \sigma \rightarrow \text{FC}_{4096} \rightarrow \text{ReLU} \rightarrow \text{Reshape} \rightarrow \mathcal{R}^{1 \times 1 \times 4096} \\
 &\rightarrow \text{depth to space}_4 \rightarrow \mathcal{R}^{4 \times 4 \times 256} \rightarrow \text{Conv}_{256}^{3 \times 3} (\text{stride} = 1) \rightarrow \text{ReLU} \\
 &\rightarrow \text{depth to space}_2 \rightarrow \mathcal{R}^{8 \times 8 \times 64} \rightarrow \text{Conv}_{64}^{3 \times 3} (\text{stride} = 1) \rightarrow \text{ReLU} \\
 &\rightarrow \text{depth to space}_2 \rightarrow \mathcal{R}^{16 \times 16 \times 16} \rightarrow \text{Conv}_{16}^{3 \times 3} (\text{stride} = 1) \rightarrow \text{ReLU} \\
 &\rightarrow \text{depth to space}_2 \rightarrow \mathcal{R}^{32 \times 32 \times 4} \rightarrow \text{Conv}_1^{5 \times 5} (\text{stride} = 1, \text{padding} = \text{valid}) \xrightarrow{\text{XIII}} \\
 &\rightarrow \text{Sigmoid} \rightarrow \text{Reshape} \Rightarrow \hat{\mathbf{x}} \in \mathcal{R}^{28 \times 28 \times 1}
 \end{aligned}$$

Prior VAE network:

$$\begin{aligned}
 \mathbf{z} &\rightarrow (\text{FC}_{512} \rightarrow \text{Leaky ReLU}) \times 3 \rightarrow \text{FC}_{d_t} \rightarrow \mu_t \in \mathcal{R}^{d_t} \\
 &\searrow \\
 &\text{FC}_{d_t} \rightarrow \text{ReLU} \Rightarrow \sigma_t \in \mathcal{R}^{d_t} \\
 \mathbf{t} &= \mu_t + \epsilon \odot \sigma_t \rightarrow (\text{FC}_{512} \rightarrow \text{Leaky ReLU}) \times 3 \\
 &\rightarrow \text{FC}_{d_z} \rightarrow \hat{\mathbf{z}} \in \mathcal{R}^{d_z}
 \end{aligned}$$

d_z is the dimension of z -space, d_t is the latent dimension of our prior VAE, μ is code mean and σ is decoder variance parameter introduced in Section ??, $\epsilon \sim \mathcal{N}(0, \mathcal{I})$ and unless otherwise specified the padding in the convolution operation is 'same'.

We use the following hyper-parameters to train the network:

Batch size	d_z	d_t	N epoch	Optimizer	Learning rate	Padding
256	16	2	50	Adam	5×10^{-4}	SAME

B.3.1.2 CelebA

For CelebA dataset, we implement a CNN encoder and a Style-GAN generator, as detailed below. Encoder network:

$$\begin{aligned}
 \mathbf{x} \in \mathcal{R}^{128 \times 128 \times 3} &\rightarrow \text{Conv}_{32}^{3 \times 3} (\text{stride} = 2) \rightarrow \text{Leaky ReLU} \\
 &\rightarrow \text{Conv}_{64}^{3 \times 3} (\text{stride} = 2) \rightarrow \text{Leaky ReLU} \\
 &\rightarrow \text{Conv}_{128}^{3 \times 3} (\text{stride} = 2) \rightarrow \text{Leaky ReLU} \\
 &\rightarrow \text{Conv}_{256}^{3 \times 3} (\text{stride} = 2) \rightarrow \text{Leaky ReLU} \\
 &\rightarrow \text{Conv}_{512}^{3 \times 3} (\text{stride} = 2) \rightarrow \text{Leaky ReLU} \\
 &\rightarrow \text{Conv}_{1024}^{3 \times 3} (\text{stride} = 2, \text{padding} = \text{valid}) \rightarrow \text{Leaky ReLU} \\
 &\rightarrow \text{FC}_{512} \rightarrow \text{FC}_{d_z} \Rightarrow \mu \in \mathcal{R}^{d_z} \\
 &\searrow \\
 &\text{FC}_{d_z} \Rightarrow \sigma \in \mathcal{R}^{d_z}
 \end{aligned}$$

Decoder network which is inspired by the Style-GAN architecture is given below. The Style Module refers to a fully connected a layer that first doubles the channel dimensions and then merges the channel dimension back to the original dimension by summing the first half channels with the original channel value weighted second half channels. This operation is modified from the original Style-GAN generator to

fit with a VAE model.

$$\begin{aligned}
\mathbf{z} = \mu + \epsilon \odot \sigma &\rightarrow (\text{FC}_{512} \rightarrow \text{Leaky ReLU}) \times 8 \rightarrow \text{Reshape} \rightarrow \mathcal{R}^{1 \times 1 \times 512} \\
&\rightarrow \text{Resize}_{128}^{2 \times 2} \rightarrow \text{CNN}_{512}^{3 \times 3} \rightarrow \text{Instance Norm} \rightarrow \text{Style Module} \rightarrow \mathcal{R}^{2 \times 2 \times 512} \\
&\rightarrow \text{Resize}_{128}^{4 \times 4} \rightarrow \text{CNN}_{512}^{3 \times 3} \rightarrow \text{Instance Norm} \rightarrow \text{Style Module} \rightarrow \mathcal{R}^{4 \times 4 \times 512} \\
&\rightarrow \text{Resize}_{128}^{8 \times 8} \rightarrow \text{CNN}_{512}^{3 \times 3} \rightarrow \text{Instance Norm} \rightarrow \text{Style Module} \rightarrow \mathcal{R}^{8 \times 8 \times 512} \\
&\rightarrow \text{Resize}_{128}^{16 \times 16} \rightarrow \text{CNN}_{256}^{3 \times 3} \rightarrow \text{Instance Norm} \rightarrow \text{Style Module} \rightarrow \mathcal{R}^{16 \times 16 \times 256} \\
&\rightarrow \text{Resize}_{64}^{32 \times 32} \rightarrow \text{CNN}_{256}^{3 \times 3} \rightarrow \text{Instance Norm} \rightarrow \text{Style Module} \rightarrow \mathcal{R}^{32 \times 32 \times 256} \\
&\rightarrow \text{Resize}_{64}^{64 \times 64} \rightarrow \text{CNN}_{128}^{3 \times 3} \rightarrow \text{Instance Norm} \rightarrow \text{Style Module} \rightarrow \mathcal{R}^{64 \times 64 \times 128} \\
&\rightarrow \text{Resize}_{32}^{128 \times 128} \rightarrow \text{CNN}_{128}^{3 \times 3} \rightarrow \text{Instance Norm} \rightarrow \text{Style Module} \rightarrow \mathcal{R}^{128 \times 128 \times 128} \\
&\rightarrow \text{CNN}_{128}^{3 \times 3} \rightarrow \text{Leaky ReLU} \rightarrow \text{CNN}_3^{3 \times 3} \rightarrow \mathcal{R}^{128 \times 128 \times 3}
\end{aligned}$$

Prior VAE network:

$$\begin{aligned}
\mathbf{z} &\rightarrow (\text{FC}_{512} \rightarrow \text{Leaky ReLU}) \times 5 \rightarrow \text{FC}_{d_t} \rightarrow \mu_t \in \mathcal{R}^{d_t} \\
&\quad \searrow \\
&\quad \text{FC}_{d_t} \rightarrow \text{ReLU} \Rightarrow \sigma_t \in \mathcal{R}^{d_t} \\
\mathbf{t} &= \mu_t + \epsilon \odot \sigma_t \rightarrow (\text{FC}_{512} \rightarrow \text{Leaky ReLU}) \times 5 \\
&\rightarrow \text{FC}_{d_z} \rightarrow \hat{\mathbf{z}} \in \mathcal{R}^{d_z}
\end{aligned}$$

We use the following hyper-parameters to train the network:

Batch	d_z	d_t	N epoch	Optimizer	Learning	Padding
size					rate	
256	256	32	75	Adam	5×10^{-4}	SAME

B.3.2 More Experimental Results

Now we show more results that supplement the results from the main texts. We give more generated samples from each VAE model, FID scores on the MNIST and fashion-MNIST datasets, an ablation study on the impacts of the prior VAE’s latent dimension on the overall modelling performance and finally more examples of latent space interpolation using the derived latent distribution.

B.3.2.1 Generated Samples from Inferred Prior Distribution

Here we generate samples from the inferred prior distribution of each VAE model. The sample quality indicates how well the prior distribution matches the learnt latent data distribution. As can be seen, our method and GMM prior produce

the best quality samples. However, in the higher dimensional latent space which is often needed for more complex datasets such as CelebA, our method outperforms the GMM prior. It is also very clear that the standard VAE with a normal prior fails to represent the inferred data distribution at all. This calls for caution in using the unit Gaussian prior if the inferred latent distribution is actually needed to represent the realistic data samples in later tasks.

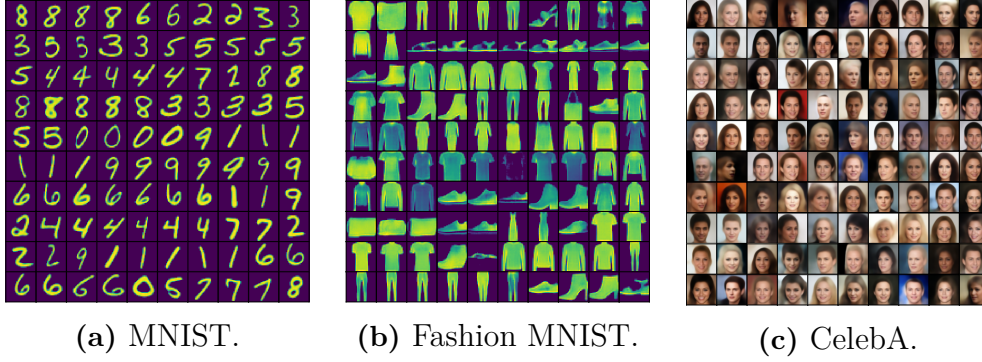


Figure B.7: 100 images generated from the inferred prior distribution in our LaD-Der model for each of the 3 datasets.

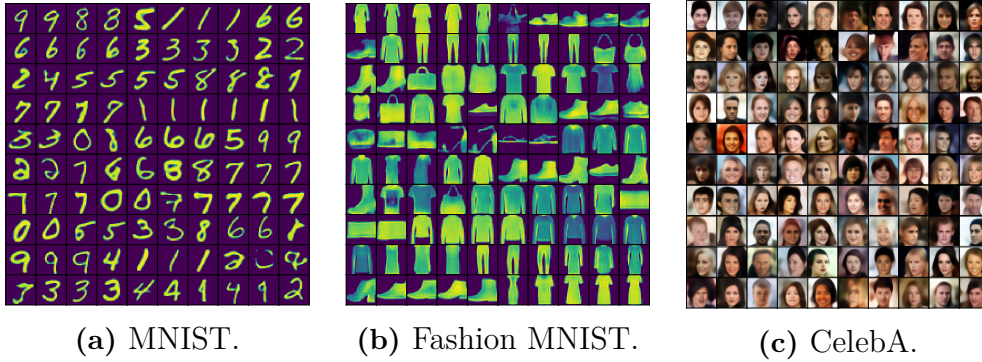


Figure B.8: 100 images generated from the inferred prior distribution in a GMM-VAE model for each of the 3 datasets.

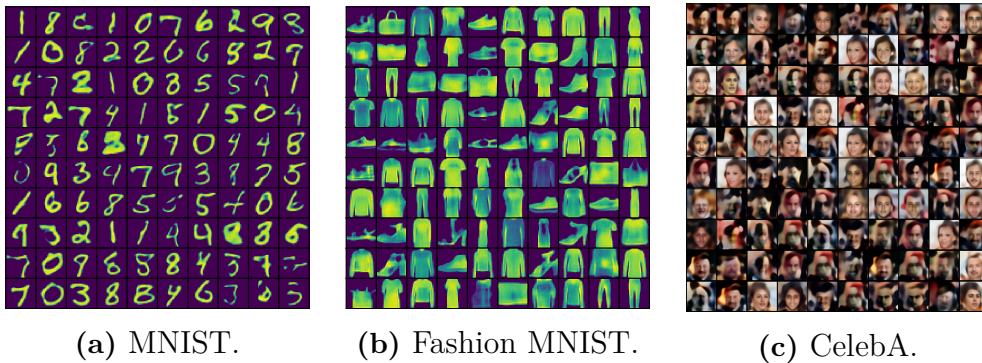


Figure B.9: 100 images generated from the inferred prior distribution in VampPrior model for each of the 3 datasets.

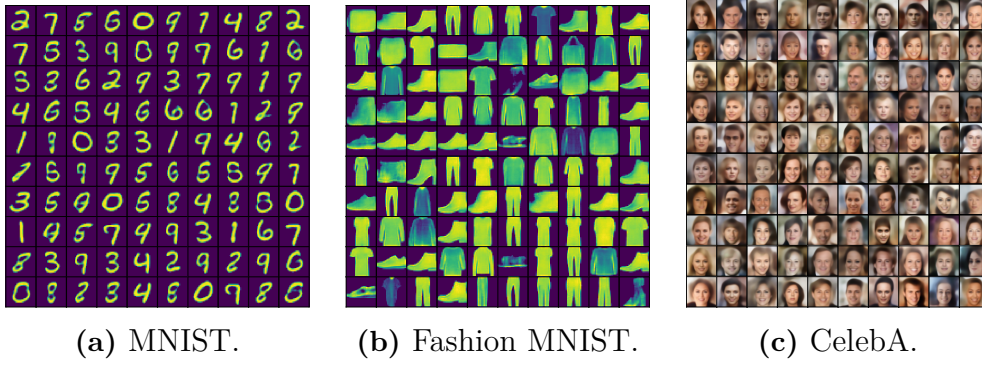


Figure B.10: 100 images generated from the inferred prior distribution in a VAE with a hierarchical prior for each of the 3 datasets.

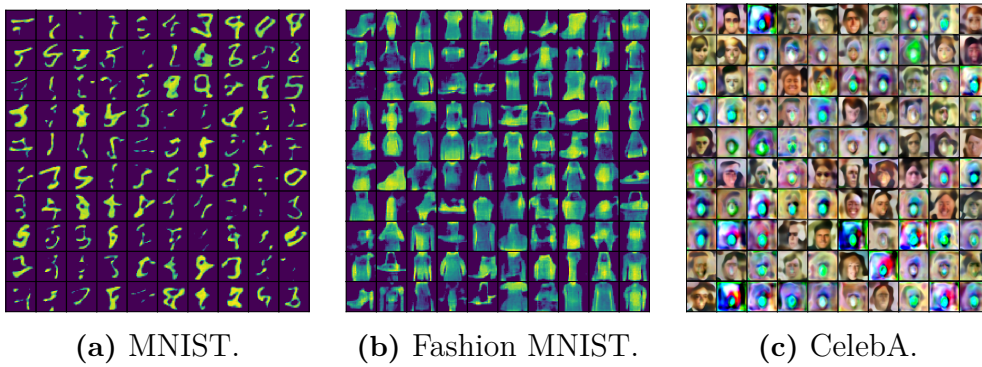


Figure B.11: 100 images generated from the inferred prior distribution in standard VAE (normal prior) for each of the 3 datasets.

B.3.2.2 FID Scores on MNIST and fashion-MNIST

In Section 4.1 from the main texts, we demonstrate that our generative prior can accurately model the inferred latent data distribution by evaluating the FID scores of both generated and reconstructed samples. We pay special attention to the difference in the visual quality between the generation and reconstruction. This is because the reconstruction samples quality of a VAE indicates the best sample quality the VAE can generate (if the prior 100% matches the inferred data manifold point-by-point). We give the FID score for CelebA dataset in the main texts. Here we also give the FID scores for MNIST and fashion-MNIST datasets, which tell a very similar story that our LaDDer model can generate very good quality samples and the gap between generation and reconstruction is small.

An interesting effect shown here is that the VAE model with a GMM prior also achieves very good performance. We reckon this is because the MNIST datasets are relatively simple and hence a GMM fitting in a relatively low-dimensional z -space can match the overall data distribution fairly well. However, when the dataset becomes more complex, such as the CelebA dataset, our generative prior is more flexible and outperforms the GMM prior, as shown in the main texts Section 4.1.

Table B.2: FID scores (lower is better) of generated and reconstructed samples from 5 VAE models with different priors. Our generative prior and GMM hyper prior achieves best sample quality in both generation and reconstruction and the minimal gap between the two.

Dataset	Prior model	Normal	VampPrior	GMM	Hierarchical	Ours	Real image
MNIST digit	Generation	401.4 $\pm$ 4.4	47.2 $\pm$ 3.6	2.3 $\pm$ 0.5	52.4 $\pm$ 4.7	6.4 $\pm$ 1.3	2.3 $\pm$ 0.4
	Reconstruction	2.1 $\pm$ 0.8	1.4 $\pm$ 0.4	2.5 $\pm$ 0.8	5.4 $\pm$ 1.4	2.4 $\pm$ 0.4	
	Difference	399.3 $\pm$ 5.2	45.8 $\pm$ 4.0	-0.2 $\pm$ 1.3	47.0 $\pm$ 3.3	4.0 $\pm$ 1.7	
MNIST fashion	Generation	266.1 $\pm$ 2.4	23.6 $\pm$ 1.2	6.0 $\pm$ 1.0	243.8 $\pm$ 2.9	5.2 $\pm$ 0.9	0.4 $\pm$ 0.1
	Reconstruction	3.2 $\pm$ 0.4	3.0 $\pm$ 0.2	2.9 $\pm$ 0.4	1.8 $\pm$ 0.2	1.0 $\pm$ 0.4	
	Difference	262.9 $\pm$ 2.8	20.6 $\pm$ 1.4	3.1 $\pm$ 1.4	242.0 $\pm$ 3.1	4.2 $\pm$ 1.3	

B.3.2.3 Ablation Study: the Impact of Hyper Latent Dimension on Performance

Here we carry out an ablation study on the impact of the latent dimension of our prior VAE model on the overall modelling performance, specifically on the generation quality. Over all three datasets, the generation quality improves as we allow the prior VAE to adopt a higher latent dimension. However, the benefit of increasing the latent dimension diminishes as the dimension reaches certain level. This indicates that the dimension reduction achieved by the prior VAE module is still valid and does not limit the model’s performance as long as its latent dimension is not set to be overly small.

Table B.3: FID scores (lower is better) of generated samples from our generative prior with different latent dimension d_t . It is clear with higher latent dimension, the generation improves. However, the benefits diminish as the dimension increases.

Dataset	Ours					Real image
	2d	4d	8d	16d	32d	
MNIST digit	58.7 $\pm$ 1.8	35.1 $\pm$ 8.1	6.4 $\pm$ 1.3	6.9 $\pm$ 0.8	N/A	2.3 $\pm$ 0.4
MNIST fashion	25.3 $\pm$ 1.6	9.9 $\pm$ 1.2	5.5 $\pm$ 0.7	5.2 $\pm$ 0.9	N/A	0.4 $\pm$ 0.1
CelebA	175.7 $\pm$ 0.5	148.4 $\pm$ 0.6	138.4 $\pm$ 0.5	133.5 $\pm$ 0.6	132.7 $\pm$ 0.5	5.9 $\pm$ 0.5

B.3.2.4 Latent Space Interpolation Through Our Shortest Likelihood Path

In the main texts, we demonstrate how to use the estimated latent distribution to facilitate further tasks through a latent space interpolation example. We show that the derived latent distribution can inform the interpolation to only step on high likelihood regions in the latent space. As a result, the inferred samples remain realistic throughout the path and render smooth transitions into the target image. Here, we show more examples on the interpolation task and we also show our shortest likely path traversal on two other VAE models to indicate it can be generally applied

to other methods. Across the interpolation results, the interpolation informed with the learnt latent distribution generates better samples. The exception is the VAE with a normal prior and this is because the normal prior poorly models the true data distribution.

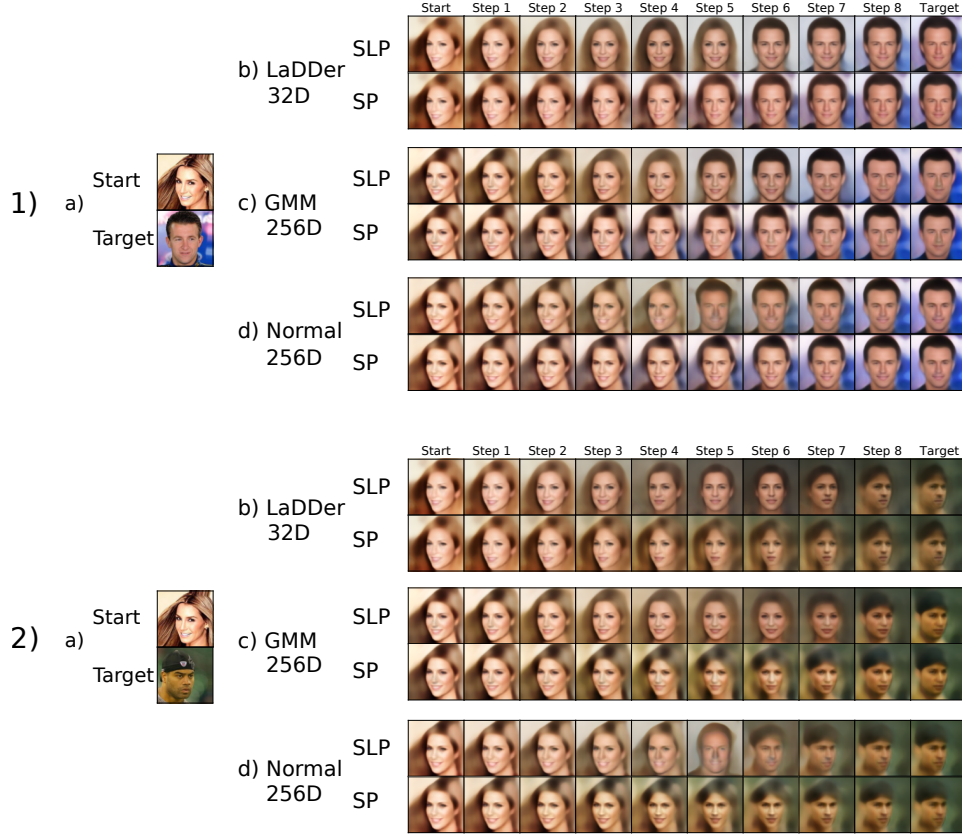


Figure B.12: Latent space interpolation between a pair of CelebA images. We show the interpolated images for 3 VAE models: (b) our proposed LaDDer, (c) a GMM prior and (d) a normal prior. For each method, top row gives the interpolation of our shortest likely path (SLP) method and bottom row gives the interpolation of the shortest path (SP) method.

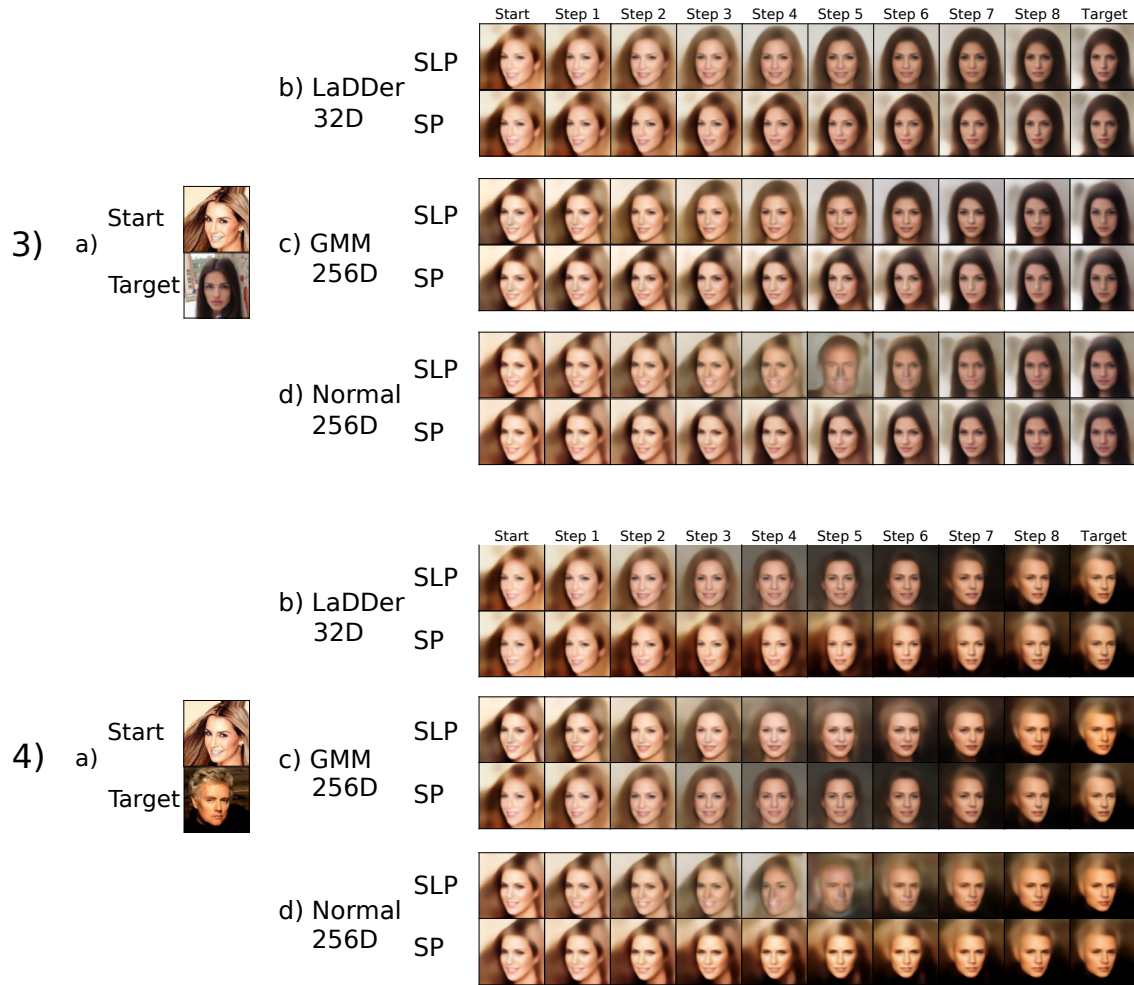


Figure B.13: Latent space interpolation between a pair of CelebA images. We show the interpolated images for 3 VAE models: (b) our proposed LaDDer, (c) a GMM prior and (d) a normal prior. For each method, top row gives the interpolation of our shortest likely path (SLP) method and bottom row gives the interpolation of the shortest path (SP) method.