Task-Oriented Learning of Structured Probability Distributions

Diane Bouchacourt
Lady Margaret Hall
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Trinity 2017

Dedicated to my sister Flora.

Task-Oriented Learning of Structured Probability Distributions

Diane Bouchacourt

Lady Margaret Hall
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Trinity 2017

Machine learning models automatically learn from historical data to predict unseen events. Such events are often represented as complex multi-dimensional structures. In many cases there is high uncertainty in the prediction process. Research has developed probabilistic models to capture distributions of complex objects, but their learning objective is often agnostic of the evaluation loss. In this thesis, we address the aforementioned deficiency by designing probabilistic methods for structured object prediction that take into account the task at hand.

First, we consider that the task at hand is explicitly known, but there is ambiguity in the prediction due to an unobserved (latent) variable. We develop a framework for latent structured output prediction that unifies existing empirical risk minimisation methods. We empirically demonstrate that for large and ambiguous latent spaces, performing prediction by minimising the uncertainty in the latent variable provides more accurate results.

Empirical risk minimisation methods predict only a pointwise estimate of the output, however there can be uncertainty on the output value itself. To tackle this deficiency, we introduce a novel type of model to perform probabilistic structured output prediction. Our training objective minimises a dissimilarity coefficient between the data distribution and the model's distribution. This coefficient is defined according to a loss of choice, thereby our objective can be tailored to the task loss. We empirically demonstrate the ability of our model to capture distributions over complex objects.

Finally, we tackle a setting where the task loss is implicitly expressed. Specifically, we consider the case of grouped observations. We propose a new model for learning a representation of the data that decomposes according to the semantics behind this grouping, while allowing efficient test-time inference. We experimentally demonstrate that our model learns a disentangled and controllable representation, leverages grouping information when available, and generalises to unseen observations.

Acknowledgements

I owe my deepest gratitude to my advisors, Dr. M Pawan Kumar and Dr. Sebastian Nowozin, for their encouragement and mentoring during those three years. This work would not have been possible without their incredible supervision. They taught me how to do research by being continously involved and interested in my work, while providing me the freedom to explore, learn and progress.

I would also like to thank Dr. Ryota Tomioka for his supervision during my internship at Microsoft Research Cambridge. His comments were certainly very helpful, and thanks to his enthousiasm I spent a fun and instructive three-months project.

I gratefully acknowledge the funding received towards my DPhil from the Microsoft Research PhD Scholarship Programme in EMEA.

I am thankful to the Center for Visual Computing at Ecole Centrale Paris where I pursued my first year of DPhil.

At the University of Oxford, I am very grateful to the members of the Optimization for Vision And Learning group, Torr group and the Visual Geometry Group for creating a great research environment. I especially thank Alban for providing an unlimited access to chewing-gums, and Puneet for his help on writing proper British sentences. I also thank the Oxford-Man Institute for letting me write this manuscript there with my friends.

I enjoyed my time in Oxford thanks to my very good friends there. For the time spent sharing good food and good wine, the laughts we had (sometimes at the last minute), the roads we drove together, I am thankful to my team, Ma (Elmarie) and Dragon (Justin). For our breaks in the park under the sun and our relaxed discussions, I thank Pankaj.

I especially thank my friends from France. To Marie-Estelle, for her cheerful voice messages. For being always enthousiastic about my work and providing excellent comments on this manuscript, I thank Kevin. I also thank Benoît for being there, because he knows. I am very grateful to the Ciscocos (Florine, Franck, Marouane), your jokes might not have always been great, but they would always make me laugh! Thank you for your continous encouragements, and for your warm welcome everytime I visited Paris. For the laughs, the video games, the stupid shows on TV while eating lentils/butternut, for being there the day I moved with my huge backpack and the day I will come back, I thank my Poulet (Manon). To

Kitsune (Kambiz), my travelling buddy who makes tahdig in the middle of Patagonia, my alter-ego, my brother from other parents, I say many thanks. I would not have done this without you.

Finally, I especially thank my sister Flora. You have always held my hand, literally and figuratively, and have led the path for me. You are a motivating force, and since we were little I have tried to follow your example. I would not be who I am without you. Thank you for caring and loving me unconditionally, I love you the same.

Statement of Originality

I confirm that this dissertation is the result of my own work, under supervision explicitly indicated below. I derived the mathematical formulations, wrote the code and performed results analysis. Sections of this work have appeared in publications as described below, of which I am the main author. Part of the figures in this dissertation are taken from these publications, with possible modifications. Where information is derived from other sources, I confirm that this is indicated in the text and that the references given are accurate to the best of my knowledge. I declare that no part of this thesis has been, or is being, submitted for any qualification other than the degree of Doctor of Philosophy at the University of Oxford.

Chapter 3

- The content of Chapter 3 is based on Bouchacourt et al. (2015). I have done this work under the supervision of with my DPhil advisors Dr. M Pawan Kumar and Dr. Sebastian Nowozin.

Chapter 4

- The content of sections 4.1, 4.2, 4.3, and 4.5 is based on Bouchacourt et al. (2016), and the content of section 4.4 is an unpublished in-depth derivation of the theoretical guarantee mentioned in Bouchacourt et al. (2016) that I have done for the manuscript. I have done this work under the supervision of my DPhil advisors Dr. M Pawan Kumar and Dr. Sebastian Nowozin.
- The content of section 4.6 and 4.7 consists of unpublished work. I have done this work under the supervision of my DPhil advisors Dr. M Pawan Kumar and Dr. Sebastian Nowozin.

Chapter 5

- The content of Chapter 5 is based on the work Bouchacourt et al. (2017), currently under review. I have done this work as an intern at Microsoft Research Cambridge, under the supervision of my internship supervisors Dr. Sebastian Nowozin and with Dr. Ryota Tomioka.

Appendices

- Rudy Bunel helped me to code the pseudo-polynomial time algorithm, described in appendix 7.5.3, to perform loss-augmented inference in steps 4 and 5 of Algorithm 3 with the MinMax kernel.

Diane Bouchacourt, September 2017

Contents

List of Theorems, Propositions and Definitions

List of Remarks

List of Figures

List of Tables

Notations

symbol	meaning
1_V	indicator function of the set V
x	scalar or vector
x^i	entry at i-th position of vector x
$\mathbb{R}$	the set of real numbers
$\mathbb{R}^n$	the set of vectors of size n with real entries
$\mathbb{R}^{n\times m}$	the set of matrices of size $n \times m$ with real entries
$\boldsymbol{x}$	scalar-valued or vector-valued random variable
$\boldsymbol{X}$	collection of K random variables $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K)$
$\mathbb{P}$	probability measure on Ω such that $\boldsymbol{x} : \Omega \to \mathcal{X}$
$\mathbb{P}_{\boldsymbol{x}} = \mathbb{P} \circ \boldsymbol{x}^{-1}$	probability measure on $\mathcal{X}$
$\mathbb{P}_{\boldsymbol{x}}(x)$	probability of event x
$\mathcal{P}$	set of probability measures
$P(\boldsymbol{x})$	cumulative distribution function (cdf)
$P(x)$	value of the cdf $P(\boldsymbol{x})$ at $\boldsymbol{x} = x$
$p(\boldsymbol{x})$	probability density function (pdf) or probability mass function (pmf)
$p(x)$	value of the pdf (or pmf) $p(\boldsymbol{x})$ at $\boldsymbol{x} = x$
$\boldsymbol{x} \equiv P$	$\boldsymbol{x}$ follows the distribution with cdf P
$x \sim P(\boldsymbol{x})$	x is sampled from $P(\boldsymbol{x})$
P^K	K-dimensional product cdf built from P
$\mathbb{E}_{x\sim P}[f(x)]$, or $\mathbb{E}_{x\sim P(\boldsymbol{x})}[f(x)]$ or $\mathbb{E}_{x\sim p(\boldsymbol{x})}[f(x)]$ or $\mathbb{E}_{p(\boldsymbol{x})}[f(\boldsymbol{x})]$	expectation of $f(\boldsymbol{x})$ with respect to $\boldsymbol{x} \equiv P$.

Table 1: Notations.

Chapter 1

Introduction

> Better understanding of the natural world not only enhances all of us as human beings, but can also be harnessed for the better good, leading to improved health and quality of life.
>
> *Sir Paul Maxime Nurse, British Scientist*

The words of British geneticist Paul Nurse apply to scientific research as a whole, and find a particular echo in artificial intelligence. In the past decades, artificial intelligence has developed towards replicating the way we, as human beings, naturally understand and evolve in the natural world. To illustrate this, imagine you are in Nice, France, on top of the hill named the "Colline du château".

Figure 1.1: View from the "Colline du château" in Nice, France.

The breeze of the Mediterranean sea that you feel on your skin carries a salty scent. You admire the old town below, with its colorful buildings so typical of the French Riviera. In the old town, there seems to be a market, so you decide to head there afterwards to find a restaurant for lunch. While this intuitive understanding of the scene is straightforward to a human, it is very challenging to reproduce artificially. First, the signals you receive are interdependent, and you process them as such. When the wind brings you a smell of salt, you naturally associate it to the sea. Second, these signals are complex and uncertain. Looking at the old town, you think that there might be a market by the presence of crowds and shops signs, but you cannot directly identify potential restaurants. Indeed, humans are able to understand the world having only a partial knowledge of it. Imagine you grew up in a typical Manhattan tower and have never seen before these colorful and small Provencal homes. While no one has explicitly told you these are habitations, you easily recognise them as such.

Artificial intelligence uses machine learning to design models that automatically replicate this ability to understand the world. Machine learning is concerned with the development of computational and statistical methods to build models that predict future events based on historical data. The *training* or *learning* of a model consists of the learning of its parameters. It is done by optimising an *objective function* on existing data, which itself constitutes the *training data set*. The model's behavior on unseen data is estimated by evaluating its performance on new data, the *testing data set*, according to a loss function associated to the task at hand. In this thesis, we explore the learning of probabilistic distributions of structured objects, while taking into account the intended use of the model, and its associated evaluation loss. We use computer vision problems to demonstrate the efficiency of the proposed methods. We begin by setting the context and motivations of our work.

1.1 Context and motivations

We begin by defining structured objects and their applications in computer vision.

1.1.1 Structured objects and applications in computer vision

We are concerned with *structured* data, that is, multi-dimensional objects that present complex dependencies between their dimensions, as opposed to scalar or real values. Structured data naturally arise in artificial intelligence tasks. For example in computer vision, if we represent an image as a set of pixels, a pixel intensity is not independent of the values of its neighbouring pixels. The task of predicting a structured output given an input is known as *structured output prediction*. For instance, consider the task of predicting, given a depth image of a hand, the pose of the hand in the form of the 3-dimensional (3D) coordinates of the fingers' joints, as shown in Figure 1.2. The pose is a multi-dimensional object, and each joint's coordinates depend on the coordinates of the other joints.

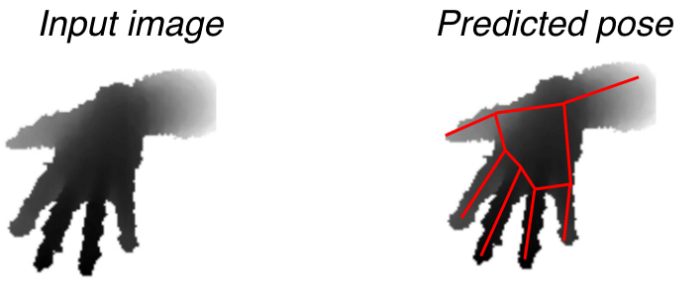


Figure 1.2: Example of a structured output prediction task: hand pose estimation from depth image. The depth image is from the NYU Hand Pose data set (Tompson et al., 2014), preprocessed as in Oberweger et al. (2015b).

Predicting structured objects is a challenging task as such objects present complex structures that simple models cannot capture. Moreover, in most applications structured objects are high-dimensional, making more complicated models

expensive to employ.

1.1.2 The importance of modelling uncertainty

In this thesis, we focus on modelling the probability distribution of structured objects, rather than predicting a pointwise estimate. There are multiple reasons that motivate the use of probabilistic methods. First, in many cases only part of the information, in the form of the *annotation* of the data (or *label*) is available. This is referred to as the *weakly* supervised setting. Consider Figure 1.3: when performing action recognition in images, we may know that a person is doing an action in the image. This is the available annotation. However, the location of the person, in the form of the coordinates of a bounding box containing the action, may be missing. Indeed, it is more expensive to obtain bounding box annotations compared to image level annotations. Probabilistic models allow us to take into account the uncertainty in the missing information.

Figure 1.3: Example of weakly annotated image. The image is from the Pascal VOC 2011 data set (Everingham et al., 2010).

Second, there is often uncertainty in the prediction itself. In the example of hand pose estimation from depth image, shown in Figure 1.2, the pose of the hand is clearly predictable given the image. However, consider that we are given as input the depth image in Figure 1.4 where the hand is fisted. In this image, because the

fingertips are occluded, there are multiple yet plausible predictions of their 3D coordinates. In this case, we will be interested in learning the probability distribution of the joints' coordinates, that is, to perform *probabilistic structured output prediction.*

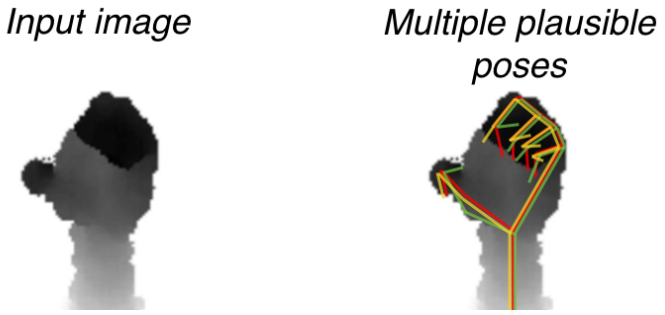


Figure 1.4: There is uncertainty in the pose for a given depth image resulting in multiple yet plausible predictions. The depth image is from the NYU Hand Pose data set (Tompson et al., 2014), preprocessed as in Oberweger et al. (2015b).

There are many challenges associated with learning the probability distribution of structured objects. First, in most cases we do not have knowledge of the real world distribution. We only have access to it via samples, in the form of the training data set. Second, we often have little prior knowledge on the distribution to capture. Wrongly specified models might obtain poor performance, while unspecified models are more flexible but can lead to intractable computations.

1.1.3 Task-oriented learning

Often, the class of distributions represented by the model does not contain the true distribution of the object to predict. For example this can be due to restricted model capacity. This is referred to as model misspecification. In this case, the choice of the learning objective influences final performance. Similar to Lacoste-Julien et al. (2011), we argue that the learning objective should be tailored to the evaluation loss in order to obtain the best performance with respect to this loss.

We present a simple example to illustrate this point. We consider a data distribution that is a mixture of two bidimensional Normal distributions. We consider two

models capturing the data distribution. Each model is able to represent a bidimensional Normal distribution with diagonal covariance, parametrised by $(\mu_1, \mu_2, \sigma_1, \sigma_2)$ where μ, σ is a mean-variance pair on each dimension. In this case, neither of the models will be able to recover the true data distribution since they do not have the ability to represent a mixture of bidimensional Normal distributions. In other words, we cannot avoid model error, as in the real world scenario. We denote by Δ_{training} the loss function employed during model training, and by Δ_{task} the evaluation loss. Each model uses its own training loss Δ_{training}. Model A employs a loss that emphasises on the first dimension of the data, specified for $x = (x_1, x_2), x' = (x'_1, x'_2) \in \mathbb{R}^2$ by $\Delta_A(x - x') = (10 \times (x_1 - x'_1)^2 + 0.1 \times (x_2 - x'_2)^2)^{1/2}$. Model B does the contrary and employs the loss function $\Delta_B(x - x') = \left(0.1 \times (x_1 - x'_1)^2 + 10 \times (x_2 - x'_2)^2\right)^{1/2}$. Each model performs a grid search over the best parameters values for $(\mu_1, \mu_2, \sigma_1, \sigma_2)$. Details of this illustrative experiment can be found in appendix 7.1.1. Figure 1.5 shows the contours of the data distribution (in black), and the contour of the distribution fitted by each model (Model A in green and Model B in red). As expected, the fitted distributions differ according to Δ_{training} employed.

		Δ_{task}	
		Δ_A	Δ_B
Δ_{training}	Δ_A	**11.6**	13.7
	Δ_B	12.1	**11.0**

Table 1.1: Δ_{task} with respect to $\Delta_{training}$ employed. Best performances in bold.

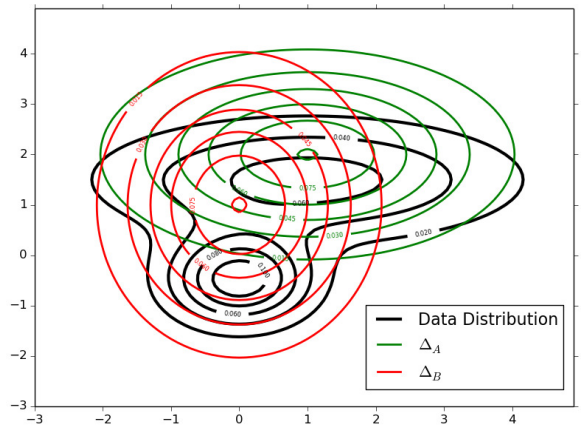


Figure 1.5: Contour lines of the data distribution, and the distribution fitted by each model.

Table 1.1 shows that the loss on the testing set, evaluated with Δ_{task}, is minimised if $\Delta_{\text{training}} = \Delta_{\text{task}}$. This simple example shows the advantage to being

able to tailor the model's training objective to have $\Delta_{\text{training}} = \Delta_{\text{task}}$. However, while we know the evaluation loss, it might be difficult to optimise it at training. For example, the evaluation loss can be non-convex or non-smooth, such as the 0/1 loss between two discrete output values y, y' defined as

$$\Delta(y, y') = \begin{cases} 0 \text{ if } y = y' \\ 1 \text{ if } y \neq y' \end{cases}$$

Research has focused on developing the best approximation of the evaluation loss, see for example the recent works of Bahdanau et al. (2016); Cisse et al. (2017); Osokin et al. (2017). In these cases, learning is done by optimising a *surrogate loss* in place of the evaluation loss.

1.2 Outline and contributions

Our contributions in the topic of probabilistic prediction of structured data are three-fold. As this thesis is organised around them, we present them along with the outline.

1. Chapter 2 introduces the theoretical background necessary to understand the thesis. We review probability theory and scoring rules theory, and describe latent variable models and deep probabilistic models.

2. In Chapter 3, we consider the setting of weakly supervised structured output prediction, with an explicit task loss that we wish to minimise. We show that existing models for weakly supervised structured output prediction can be unified under a common umbrella. This in turn allows us to derive a unique optimisation algorithm and perform a fair comparison between the prediction criterion of each model. Using the proposed method, we empirically show that taking into account the uncertainty in the missing information when

predicting the output leads to better performance. This chapter is based on the work Bouchacourt et al. (2015), published in the proceedings of the International Conference on Computer Vision (ICCV) 2015.

3. In Chapter 4, we move towards the case where there is strong uncertainty in the structured output to predict. We propose a new type of learning objective to perform probabilistic structured output prediction and show theoretical guarantees of our model. In contrast to existing probabilistic models that are agnostic of the evaluation loss, our method can be tailored to the loss related to the task at hand. In the specific case of a discrete output space, our learning procedure is non-differentiable, and we derive an approximation algorithm to learn the model's parameters. Experimental evaluation shows the relevance of our model to capture distributions over complex, structured outputs. This chapter is based on the work Bouchacourt et al. (2016), published in the proceedings of Advances in Neural Information Processing Systems (NIPS) 2016, and also contains unpublished work.

4. In Chapter 5, we continue to consider the prediction of structured probability distributions. However the evaluation loss is not explicitly given. The user's intended application of the model is deduced through a very weak form of supervision. Specifically, we are given a particular grouping of the data, where within a group the data share a common factor of variation. We derive a new model for learning a representation of the data that decomposes according to the semantics embedded in the grouping, while retaining efficient test-time inference over the latent variables. Experimental evaluation shows that the proposed method learns a semantically meaningful disentangled representation and allows control over the latent representation. Moreover, the model generalises to unseen data and is able to reduce uncertainty by leveraging group information. This chapter is based on the work Bouchacourt et al.

(2017) currently under review for the proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) Conference 2018, as well as a patent application to the UK office.

5. Chapter 6 concludes the thesis. We discuss possible research directions for extending the practical and theoretical implications of our work.

Chapter 2

Mathematical preliminaries

This chapter introduces the theoretical background necessary to understand the thesis. We will present the theory of scoring rules, latent variable models and deep probabilistic models. This chapter also sets the notations for the thesis. For a definition of the basics of probability theory we employ, we refer the reader to Appendix 7.3.

2.1 Scoring rules

An important question we will be interested in throughout this thesis is "how good" is our modelling distribution Q, to capture the distribution of interest, denoted P. Therefore, we need a way to evaluate "how close are the distributions P and Q?". In this section we discuss the theory of scoring rules, and how they provide a way to answer this question.

2.1.1 Definition and relation to divergences

We consider a random variable $\boldsymbol{x} : \Omega \to \mathcal{X}$ and use the notation explained in Remark 7.2. That is, we consider the sample space $\mathcal{X}$ and the class of probability measures $\mathcal{P}$ defined on the σ-algebra $\mathcal{B}(\mathcal{X})$, and we associate a measure with its

cdf P and denote $P \in \mathcal{P}$. A probabilistic forecast is any distribution $Q \in \mathcal{P}$. Once the forecast is fixed, a realisation x materialises, and we want to assess the quality of the forecasting distribution. For example, consider a sample space consisting of the outcomes "rain tomorrow" and "not rain tomorrow". An agent makes a weather forecast according to his belief, represented by the distribution Q, and the true distribution of the nature is represented by P. Say the agent predicts "rain tomorrow" and it effectively rains the next day, we would like to positively reward the agent for his correct forecast. Scoring rules provide us a way to evaluate such reward.

The following definitions come from Gneiting and Raftery (2007). A scoring rule is a function $S : \mathcal{P} \times \mathcal{X} \to \mathbb{R} \cup \{-\infty, \infty\}$ such that $S(Q, x)$ is taken to mean the quality of the forecast. Scoring rules can be positively oriented, in which case the scoring rule assigns a higher score to better forecasting distributions, and negatively oriented, in which case a smaller value of the scoring rule means a better forecast. Throughout this thesis we will consider positively oriented scoring rules. The expected score of a forecasting distribution Q given by a scoring rule S under the distribution P is defined as follows.

Definition 2.1 (Expected score)**.** *Given the sample space $\mathcal{X}$, with the σ-algebra $\mathcal{F}$ on $\mathcal{X}$ and a class of probability measures $\mathcal{P}$ on $\mathcal{F}$, the expected score associated with the scoring rule $S : \mathcal{P} \times \mathcal{X} \to \mathbb{R} \cup \{-\infty, \infty\}$ under the distribution $P \in \mathcal{P}$ is*

$$S(Q, P) = \int_{\mathcal{X}} S(Q, x) \mathrm{d}P(x).$$

Taking the example of the weather forecast again, the expected score is the expected reward of the weather forecasting agent under the realisations of the weather, sampled from the true distribution P. We focus on regular scoring rules that assign scores in $\mathbb{R} \cup \{-\infty\}$[1]. Intuitively, we would like to ensure that if the forecasting

[1]The value $-\infty$ for regular scoring rules is allowed only for $Q \neq P$.

distribution Q is perfectly modelling P, the forecaster gets maximal reward. In other words, the expected score should be maximised when $Q = P$. This is ensured if the scoring rule is proper, that is,

Definition 2.2 (Proper scoring rule)**.** *A regular scoring rule is proper when the expected score $S(Q, P)$ is maximised at $Q = P$, and strictly proper if $Q = P$ is the unique maximiser of $S(Q, P)$.*

For proper scoring rule, the value of the expected score when $Q = P$ is defined as the expected score function $G(P) = S(P, P)$, and is also referred to as the generalised entropy of S. Intuitively, we should compare the performance of the forecasting agent to this value. Indeed, if $S(P, P)$ is very high, a forecasting agent should be considered a "good forecaster" only if his reward $S(Q, P)$ is large enough. Therefore, the quantity we are interested in is the divergence corresponding to a proper scoring rule S, defined as follows.

Definition 2.3 (Divergence function associated with a scoring rule)**.** *The divergence function associated with a regular, proper scoring rule S is*

$$d(Q, P) = S(P, P) - S(Q, P) \;\; P, Q \in \mathcal{P}.$$

Since S is proper, then d is non-negative with value 0 if (and only if, in the case of strictly proper) $Q = P$. Furtheremore, if $G(Q)$ is differentiable with respect to Q, then d is a Bregman divergence (Huszár, 2013).

Theorem 1 and Theorem 2 of Gneiting and Raftery (2007) state that a regular scoring rule S is (strictly) proper if and only if the expected score function G is (strictly) convex and the function $S(Q, .) : \mathcal{X} \to \mathbb{R} \cup \{-\infty\}$ is a subtangent of G at the point Q, for all $Q \in \mathcal{P}$. This is an interesting result. Indeed, it means that

from a bounded[2] convex function G, we could generate a proper scoring rule.

2.1.2 Application of scoring rules

We present below several key uses of scoring rules. In this thesis, we will be particularly interested in the use of scoring rules for parameter estimation and model evaluation in Chapter 4.

Probability elicitation. In the context of probability elicitation, proper scoring rules encourage a forecaster to report his true beliefs. Let us take an example from Parmigiani and Inoue (2009). Consider that a bookmaker is reporting odds on a game, where P represents the true beliefs of the bookmaker on the outcome of the game, and Q represents his reported distribution. If the bookmaker knows that his score will be evaluated using a proper scoring rule, he maximises his expected reward by making predictions according to the true beliefs he holds, that is, according to P.

Bayesian model selection. In Bayesian model selection, the marginal log-likelihood of a model depends on a scaling constant determined by the prior on the model, which can be intractable. In this context, Dawid and Musio (2015); Dawid et al. (2017) propose to replace the log-likelihood by proper scoring rules that are insensitive to the scaling constant.

Parameter estimation. In the context of parameter estimation, we will use proper scoring rules to design training objectives. Consider that we want to learn the parameters of a model's distribution $Q(\boldsymbol{x};\theta)$, given a training data set $\mathcal{D}$ consisting of n samples $\mathcal{D} = (x_1, \ldots, x_n)$. We build an estimator of θ by maximising

[2]The scoring rule must be regular in order to be proper, therefore $S(P, P)$ must not take the values $(-\infty, +\infty)$.

the mean score, that is,

$$\hat{\theta} = \arg\max_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} S(Q(\boldsymbol{x}; \theta), x_i).$$

If the model class, that is represented by Θ, contains the true data generating distribution, then this is a consistent estimator. Under model misspecification, using different scoring rules leads to different estimates. In this case, it is interesting to choose a strictly proper scoring rule that is tailored to the problem at hand.

A widely use example of parameter estimation with scoring rules is maximum log-likelihood. When statisticians learn a model by maximum log-likelihood estimation, they actually maximise the expected value of the Logarithmic score, defined as,

$$S(Q, x) = \log q(x).$$

The Logarithmic score is strictly proper. The associated divergence function is the Kullback-Leibler divergence,

$$\begin{aligned} d(Q, P) &= S(P, P) - S(Q, P) \\ &= \int_{\mathcal{X}} p(x) \log p(x) \mathrm{d}x - \int_{\mathcal{X}} p(x) \log q(x) \mathrm{d}x \qquad = \int_{\mathcal{X}} p(x) \log \frac{p(x)}{q(x)} \mathrm{d}x, \end{aligned}$$

and the associated entropy is the negative Shannon entropy. A singular property of the Logarithmic score is that it is local, that is, the score if x realises is computed using only the corresponding predicted density $q(x)$. It does not depend on the probabilities of the outcomes that did not realised. This property is unique to the Logarithmic score, see Bernardo (1979); Parmigiani and Inoue (2009).

Model evaluation. When we will want to perform model evaluation, we will employ divergences associated with strictly proper scoring rules to evaluate the

quality of forecasting models. Let us consider two forecasting distributions Q_1 and Q_2 that model the distribution P, and suppose that Q_1 is a genuinely better model for representing P than Q_2. The scores are evaluated with respect to a data set of samples drawn from P. For example if we consider a data set of iid samples $\mathcal{D} = (x_1, \ldots, x_n)$, we compute

$$S(Q_1, \mathcal{D}) = \frac{1}{n} \sum_{i=1}^{n} S(Q_1, x_i),$$
$$S(Q_2, \mathcal{D}) = \frac{1}{n} \sum_{i=1}^{n} S(Q_2, x_i).$$

If we choose the scoring rule S as evaluation metric, then we say that Q_1 is a better model of P than Q_2, when evaluated on $\mathcal{D}$, if $S(Q_1, \mathcal{D}) > S(Q_2, \mathcal{D})$.

2.1.3 Energy score and Kernel score

In this thesis, we will interest ourselves in two particular scoring rules, the Energy score and the Kernel score.

Energy score When $\mathcal{X} = \mathbb{R}^m$, the Energy score is defined as follows.

Definition 2.4 (Energy score)**.** *Let $\mathcal{P}_\beta, \beta \in (0, 2)$ be the class of probability measures on $\mathcal{B}(\mathbb{R}^m)$ such that $\mathbb{E}_P\left[\|\boldsymbol{x}\|_2^\beta\right]$ is finite, where $\|.\|_2$ denotes the Euclidean norm. The Energy score is defined as*

$$S(Q, x) = \frac{1}{2}\mathbb{E}_{x', x'' \sim Q}\left[\|x' - x''\|_2^\beta\right] - \mathbb{E}_{x' \sim Q}\left[\|x' - x\|_2^\beta\right], Q \in \mathcal{P}_\beta.$$

where x', x'' are two samples, independently drawn, of the random variable $\boldsymbol{x}$ with distribution $Q \in \mathcal{P}_\beta$.

Theorem 5 in Gneiting and Raftery (2007) guarantees the strict propriety of the Energy score for $\beta \in (0, 2)$.

Kernel score The term Kernel score was introduced by Dawid (2007), and is a proper scoring rule, as shown by Theorem 4 in Gneiting and Raftery (2007).

Definition 2.5 (Kernel score)**.** *Let $\mathcal{X}$ be a Hausdorff space and let K be a non-negative, continuous negative definite kernel on $\mathcal{X} \times \mathcal{X}$, and $\mathcal{P}$ be the space of probability measures on $\mathcal{B}(\mathcal{X})$. The kernel scoring rule*

$$S(Q, x) = \frac{1}{2}\mathbb{E}_{x',x''\sim Q}[K(x', x'')] - \mathbb{E}_{x'\sim Q}[K(x, x')], Q \in \mathcal{P},$$

where x', x'' are two samples, independently drawn, of the random variable $\boldsymbol{x}$ with distribution $Q \in \mathcal{P}$.

With $\mathcal{X} = \mathbb{R}^m, \beta \in (0, 2), K(x, x') = \|x - x'\|_2^\beta$, the Kernel score recovers the Energy score. Theorem 4 in Gneiting and Raftery (2007) only states propriety (and not *strict* propriety) of the Kernel score for non-negative, continuous negative definite kernel (in which case K measures the dissimilarity between two samples). By relating negatively oriented scoring rules to Maximum Mean Discrepancy (MMD) (Gretton et al., 2012), Huszár (2013) shows that the scoring rule is *strictly* proper whenever the kernel K is characteristic. In this case K measures the similarity between two samples. A kernel is characteristic if the associated kernel mean embedding is injective (Muandet et al., 2016). Intuitively, the injectivity property ensures that "all the information" about two compared distributions P and Q is preserved by the kernel mean embedding. Therefore if the distributions have the same embeddings they are the same, ensuring the strict propriety.

In the case of finite domains, any strictly positive definite (SPD) kernel is universal, hence characteristic (Borgwardt et al., 2006). Therefore, in the case of a finite sample space $\mathcal{X}$, the kernel K being SPD is sufficient to ensure strict propriety of the Kernel score. We derive a proof using scoring rule theory that when $\mathcal{X}$ is finite, the negative of the Kernel score defined with a SPD kernel K is strictly proper.

The conclusion of our Theorem 2.1 is not new, but we show it directly using strictly proper scoring rule theory. Note that, compared to Huszár (2013), we consider the negative of the Kernel score, as we consider positively oriented scoring rules.

Theorem 2.1 (Strict propriety of the Negative Kernel score with SPD kernels)**.** *We assume $\mathcal{X} \subseteq \mathbb{R}^d$ is countable and finite. Let $K(x, x') : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a strictly positive definite kernel on $\mathcal{X} \times \mathcal{X}$, and $\mathcal{P}$ be the space of probability measures on $\mathcal{B}(\mathcal{X})$. We denote x', x'' two samples, independently drawn, of the random variable $\boldsymbol{x}$ with distribution $Q \in \mathcal{P}$. The Negative Kernel score*

$$S(Q, x) = \mathbb{E}_{x' \sim Q}[K(x, x')] - \frac{1}{2}\mathbb{E}_{x', x'' \sim Q}[K(x', x'')] \, , Q \in \mathcal{P},$$

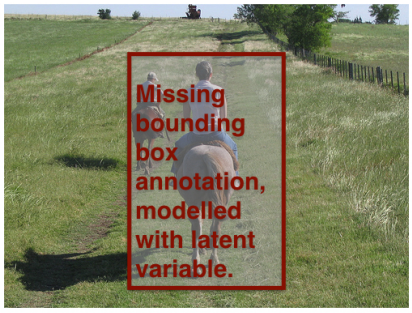
is strictly proper relative to the class of the probability measures on $\mathcal{B}(\mathcal{X})$ for which $\mathbb{E}_{x', x'' \sim Q}[K(x', x'')]$ is finite.

Proof. See appendix 7.3.3. □

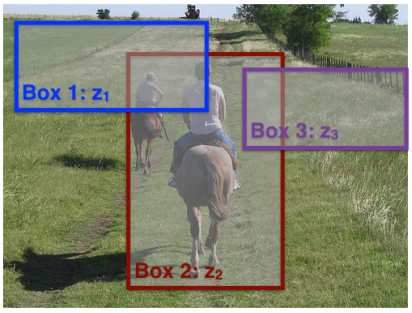
2.2 Latent variable models

In the context of statistical estimation we only have samples from the distribution of interest in the form of our training data set $\mathcal{D}$. However, some random variables are never observed. For example they can correspond to physical processes that are not measured for practical reasons. Consider the weakly supervised image from Figure 1.3 in our introduction. Latent variable models represent the missing information, that is, the coordinates of the bounding box containing the action, with a latent (or hidden) variable. This is shown in Figure 2.1. Latent variables are random variables that are not directly observed but their values are inferred from values of observed variables.

Two types of methods have been proposed to deal with latent variables. We



(a) Weakly annotated image.



(b) The latent variable $\boldsymbol{z}$ represents the missing information.

Figure 2.1: Example of weakly annotated image with a latent variable. The image is from the Pascal VOC 2011 data set (Everingham et al., 2010).

refer to the first type as probabilistic methods, where the parameters of the model are estimated by maximising the marginalised log-likelihood. The second type of method learns the model parameters by minimising the regularised empirical risk. Both types of methods have been extensively studied and apply to the case of continuous or discrete random variables. For clarity, we present each type of method in the context of its use in this thesis.

2.2.1 Probabilistic methods and variational inference

We consider a data set $\mathcal{D}$ consisting of samples drawn from an observed random variable $\boldsymbol{x}$. For clarity, we will consider a data set consisting of a single sample $\mathcal{D} = \{x\}$. We consider a latent variable denoted by $\boldsymbol{z}$, and a probabilistic model $P(\boldsymbol{x}, \boldsymbol{z}; \theta)$ where θ are the parameters of the model. Probabilistic methods estimate θ by maximising the log-likelihood, marginalised with respect to the latent variable. The marginalised log-likelihood (or evidence) of the model is

$$\log p(x; \theta) = \log \int_{\mathcal{Z}} p(x, z; \theta) \mathrm{d}z. \tag{2.1}$$

We see that maximising the evidence requires computing $p(x, z; \theta)$ for all possible values z of $\boldsymbol{z}$. This can be difficult to optimise. Consider the case when the space of possible values of the latent variable $\boldsymbol{z}$ is large, computing the evidence requires summing over all possible values. In this setting, rather than directly maximising the evidence, probabilistic methods maximise a lower bound on the evidence $\log p(x; \theta)$, as follows,

$$\begin{aligned}
\log p(x;\theta) &= \log \int_{\mathcal{Z}} p(x, z; \theta) \mathrm{d}z \\
&= \log \int_{\mathcal{Z}} q(z) \frac{p(x, z; \theta)}{q(z)} \mathrm{d}z \\
&\geq \int_{\mathcal{Z}} q(z) \log \frac{p(x, z; \theta)}{q(z)} \mathrm{d}z \\
&= \int_{\mathcal{Z}} q(z) \log p(x, z; \theta) \mathrm{d}z + \int_{\mathcal{Z}} q(z) \log q(z) \mathrm{d}z \\
&= \mathbb{E}_{q(\boldsymbol{z})}[\log p(x, \boldsymbol{z}|\theta)] - \mathbb{E}_{q(\boldsymbol{z})}[\log q(\boldsymbol{z})] := \mathcal{L}(x; q, \theta),
\end{aligned}$$

where we use Jensen's inequality. The resulting inequality is true for any distribution $q(\boldsymbol{z})$[3]. The bound $\mathcal{L}(x; q, \theta)$ is referred to as the Evidence Lower Bound (ELBO). Let us look at the difference between the evidence and the ELBO.

$$\begin{aligned}
\log p(x;\theta) - \mathcal{L}(x; q, \theta) &= \log p(x;\theta) - \int_{\mathcal{Z}} q(z) \log \frac{p(x, z; \theta)}{q(z)} \mathrm{d}z \\
&= \log p(x;\theta) - \int_{\mathcal{Z}} q(z) \log \frac{p(z|x;\theta) p(x;\theta)}{q(z)} \mathrm{d}z \\
&= -\log \int_{\mathcal{Z}} q(z) \log \frac{p(z|x;\theta)}{q(z)} \mathrm{d}z \\
&= \mathrm{KL}(q(\boldsymbol{z}) || p(\boldsymbol{z}|x;\theta)),
\end{aligned}$$

where $\mathrm{KL}(q(\boldsymbol{z})||p(\boldsymbol{z}|x;\theta))$ is the Kullback-Leibler (KL) divergence between $q(\boldsymbol{z})$ and $p(\boldsymbol{z}|x;\theta)$. The KL-divergence is non-negative, and equals zero only

[3] subject to $q(z) > 0$ whenever $p(z|x;\theta) > 0$.

when $q(\boldsymbol{z}) = p(\boldsymbol{z}|x;\theta)$. Therefore, the bound is tight when $q(\boldsymbol{z}) = p(\boldsymbol{z}|x;\theta)$. Once the ELBO equals the evidence, we maximise the ELBO with respect to the model parameter θ to increase the evidence.

The Expectation-Maximisation (EM) algorithm (Dempster et al., 1977; Rolf, 1974) does this in a two-step procedure. The first step proceeds as follows. Given a current value of the parameters θ^{old}, we first "push" the ELBO $\mathcal{L}(x; q, \theta^{\text{old}})$ so that is equals the quantity we are interested in, that is, the evidence $\log p(x;\theta)$. Specifically, the first step does:

$$\text{Set } q(\boldsymbol{z}) = \arg\max_{\tilde{q}} \mathcal{L}(x; \tilde{q}, \theta^{\text{old}}).$$

In the EM algorithm, we consider that we can set the bound to be tight, that is, the solution to first step achieves $q(\boldsymbol{z}) = p(\boldsymbol{z}|x;\theta^{\text{old}})$. In this case, the ELBO becomes:

$$\begin{aligned}\mathcal{L}(x; q, \theta^{\text{old}}) &= \mathbb{E}_{q(\boldsymbol{z})}[\log p(x, \boldsymbol{z}; \theta)] - \mathbb{E}_{q(\boldsymbol{z})}[\log q(\boldsymbol{z})] \\ &= \mathbb{E}_{p(\boldsymbol{z}|x;\theta^{\text{old}})}[\log p(x, \boldsymbol{z}; \theta)] - \mathbb{E}_{p(\boldsymbol{z}|x;\theta^{\text{old}})}\left[\log p(\boldsymbol{z}|x;\theta^{\text{old}})\right].\end{aligned}$$

The distribution q is now kept fixed at $p(\boldsymbol{z}|x;\theta^{\text{old}})$. Note that the term $\mathbb{E}_{p(\boldsymbol{z}|x;\theta^{\text{old}})}\left[\log p(\boldsymbol{z}|x;\theta^{\text{old}})\right]$ in the ELBO is independent of θ, but only depends on the current value θ^{old}. Therefore maximising the ELBO with respect to θ is equivalent to maximising $\mathbb{E}_{p(\boldsymbol{z}|x;\theta^{\text{old}})}[\log p(x, \boldsymbol{z}; \theta)]$. In other words, the first step allows us to find the posterior $p(\boldsymbol{z}|x;\theta^{\text{old}})$ so that we can compute this expectation. Therefore, the first step is referred to as *Expectation*-step (E-step). Now we maximise the ELBO with respect to θ, which will in turn increase the evidence of at least as much as the ELBO does. This is done with second step, that is, the *Maximisation*-step (M-step):

$$\text{Set } \theta^{\text{new}} = \arg\max_{\tilde{\theta}} \mathbb{E}_{p(\boldsymbol{z}|x;\theta^{\text{old}})}\left[\log p(x, \boldsymbol{z}; \tilde{\theta})\right].$$

Because q is kept fixed at $p(\boldsymbol{z}|x;\theta^{\text{old}})$, the KL-divergence between q used to compute the expectation and the new exact posterior $p(\boldsymbol{z}|x;\theta^{\text{new}})$ will not be 0. In other words, the bound will not be tight. Therefore the EM algorithm alternates between the two steps until convergence.

We see that the EM algorithm evaluates $p(\boldsymbol{z}|x;\theta)$. Indeed, computing the posterior distribution of the latent variable given the observation is a key task in the probabilistic framework. The advantage of the EM algorithm arises when we can set the ELBO to equal the evidence, by performing exact evaluation of the posterior. However, in some cases the posterior distribution $p(\boldsymbol{z}|x;\theta)$ is intractable, in other words, we cannot set $q(\boldsymbol{z}) = p(\boldsymbol{z}|x;\theta)$. This happens for example when we want to employ complicated posterior distributions. One way to circumvent this issue is to use variational inference methods (MacKay, 2002). The main idea in variational methods is to employ a posterior q parametrised by ϕ and use $q(\boldsymbol{z}|x;\phi)$ to approximate the exact posterior $p(\boldsymbol{z}|x;\theta)$. The lower bound to optimise is now a function of (ϕ, θ, x). Specifically,

$$\begin{aligned}\log p(x;\theta) &= \log \int_{\mathcal{Z}} p(x,z;\theta) \\ &\geq \int_{\mathcal{Z}} q(z|x;\phi)\log p(x,z;\theta)\mathrm{d}z + \int_{\mathcal{Z}} q(z|x;\phi)\log q(z|x;\phi)\mathrm{d}z \\ &= \mathbb{E}_{z\sim q(\boldsymbol{z}|x;\phi)}[\log p(x|z,\theta)] - \mathrm{KL}(q(\boldsymbol{z}|x;\phi)||p(\boldsymbol{z};\theta)) := \mathcal{L}(x;\phi,\theta). \quad (2.2)\end{aligned}$$

The ELBO $\mathcal{L}(x;\phi,\theta)$ is composed of two terms, the first one is the expected log-likelihood of the data under the model. The second one is the KL-divergence between the variational posterior $q(\boldsymbol{z}|x;\phi)$ and the prior distribution of the latent variable, $p(\boldsymbol{z};\theta)$. Again, the gap between the ELBO and the evidence is the KL divergence between the variational posterior and the model's posterior, $\mathrm{KL}(q(\boldsymbol{z}|x;\phi)||p(\boldsymbol{z}|x;\theta))$. The parameters θ, ϕ are learned by maximising $\mathcal{L}(x;\phi,\theta)$. We will employ variational methods in Chapter 5 when we will

want to manipulate the latent variable in order to perform a variety of tasks.

2.2.2 Empirical risk minimisation methods

The second type of method for dealing with latent variable models that we consider in this thesis is Empirical Risk Minimisation (ERM) methods. We start by explaining ERM methods when all variables are observed, then present the most commonly used ERM method for latent variable models. We discuss ERM methods in the context of structured output prediction where the goal is to predict the value of a structured output variable $\boldsymbol{y}$ taking values in $\mathcal{Y}$, corresponding to an input random variable $\boldsymbol{x}$ taking values in $\mathcal{X}$, with joint data distribution $\tau(\boldsymbol{x}, \boldsymbol{y})$.

Specifically, we want to learn a function $f : \mathcal{X} \to \mathcal{Y}$ mapping an input value to an output value. We also assume that we are given a non-negative loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ that measures the discrepancy between two outputs. In this thesis, we will be interested in ERM methods in the setting of discrete output space $\mathcal{Y}$. The associated risk is defined as the expectation of the loss function, that is,

$$R(f) = \mathbb{E}_{\tau(\boldsymbol{x},\boldsymbol{y})}[\Delta(f(\boldsymbol{x}), \boldsymbol{y})] \,.$$

In the context of statistical estimation, we consider a training data set $\mathcal{D}$. Again, for clarity, we consider a single input-output training pair, $\mathcal{D} = \{(x, y)\}$. ERM methods learn the function f by minimising an estimator of $R(f)$, the *empirical risk* $\hat{R}(f, \mathcal{D})$, computed with the training data, as follows,

$$f^*(\mathcal{D} = \{(x, y)\}) = \arg\min_{f \in \mathcal{F}} \hat{R}(f, \mathcal{D}) = \arg\min_{f \in \mathcal{F}} \Delta(f(x), y). \tag{2.3}$$

The value $\Delta(f(x), y)$ measures the discrepancy between the predicted output $f(x)$ and the output y paired with x in the training data set, referred to as the *ground-truth* output.

Structured prediction with Structural Support Vector Machine. The most commonly used ERM method is the Structural Support Vector Machine (SSVM) model (Taskar et al., 2003; Tsochantaridis et al., 2004). The SSVM model extends the Support Vector Machine (Vapnik, 1995) model to the case of structured data. The original SSVM model learns a linear prediction function of the form

$$f(x) = \arg\max_{\tilde{y} \in \mathcal{Y}} \theta^T \phi(x, \tilde{y}),$$

where θ are the parameters of the model and ϕ is a joint feature vector that describes the relationship between the input and output and can be computed beforehand. Consider the task of action classification on the weakly annotated example of Figure 1.3. The different values for the latent variable correspond to possible bounding box containing the action performed, and the feature vector could consist of activation specific activation scores extracted for each possible bounding box. The use of the arg max operator renders the minimisation of the empirical risk in equation (2.3) discontinuous with respect to θ. SSVM model overcomes this difficulty by minimising an upper bound on the loss value at each sample, as follows,

$$\Delta(y_\theta(x), y) \leq \max_{\tilde{y} \in \mathcal{Y}} [\theta^T \phi(x, \tilde{y}) + \Delta(\tilde{y}, y)] - \theta^T \phi(x, y).$$

This results in the following learning objective

$$\min_{\theta \in \Theta} \frac{1}{2} ||\theta||^2 + \max_{\tilde{y} \in \mathcal{Y}} [\theta^T \phi(x, \tilde{y}) + \Delta(\tilde{y}, y) - \theta^T \phi(x, y)], \tag{2.4}$$

where the term $\frac{1}{2}||\theta||^2$ regularises over the parameters of the model θ to avoid overfitting. The learning objective is now convex and continuous with respect to the model parameters, but it remains non-differentiable. Thereby, it is reformulated with the introduction of a variable ξ, referred to as the "slack" variable, which represents

the value of the maximum. This leads to the reformulation of Equation (2.4) as

$$\begin{aligned} &\min_{\theta\in\Theta} \frac{1}{2}||\theta||^2 + \xi, \\ &s.t.\ \max_{\tilde{y}\in\mathcal{Y}}[\theta^T\phi(x,\tilde{y}) + \Delta(\tilde{y},y) - \theta^T\phi(x,y)] \leq \xi. \end{aligned} \tag{2.5}$$

In order to work with linear constraint, another reformulation is as follows,

$$\begin{aligned} &\min_{\theta\in\Theta} \frac{1}{2}||\theta||^2 + \xi, \\ &s.t.\ \theta^T\phi(x,\tilde{y}) + \Delta(\tilde{y},y) - \theta^T\phi(x,y) \leq \xi, \forall y \in \mathcal{Y}. \end{aligned} \tag{2.6}$$

While the number of constraints has increased from 1 to $|\mathcal{Y}|$ by reformulating equation (2.5) to equation (2.6), the constraints in equation (2.6) are linear in θ. To efficiently solved the resulting optimisation problem (Taskar et al., 2003; Tsochantaridis et al., 2004) propose a cutting-plane algorithm that exploits the sparseness of the problem by keeping a set of potentially active constraints and updating it iteratively.

Latent Structural Support Vector Machine. In the presence of latent variables, SSVM have been extended to Latent SSVM (LSSVM) (Felzenszwalb et al., 2008; Yu and Joachims, 2009). In the LSSVM model the joint feature vector is extended with an extra argument z to $\phi(x,y,z)$ to describe the relation among input, output, and latent variable value z. The LSSVM model learns a linear prediction function of the form

$$f(x) = (y_\theta(x), z_\theta(x)) = \arg\max_{\tilde{y}\in\mathcal{Y},\tilde{z}\in\mathcal{Z}} \theta^T\phi(x,\tilde{y},\tilde{z}).$$

where we denote $y_\theta(x), z_\theta(x)$ to emphasise that these values are computed for a given θ and for the input value x. With the introduction of the latent variable, the

empirical risk now takes the following form,

$$\begin{cases} \hat{R}(f, \mathcal{D} = \{(x, y)\}) = \Delta(f(x), y), \\ f(x) = (y_\theta(x), z_\theta(x)). \end{cases}$$

Similarly to the original SSVM model, the LSSVM model upper bounds the loss value at each sample, as follows,

$$\Delta(y_\theta(x), z_\theta(x), y) \leq \max_{\tilde{y} \in \mathcal{Y}, \tilde{z} \in \mathcal{Z}} [\theta^T \phi(x, \tilde{y}, \tilde{z}) + \Delta(\tilde{y}, \tilde{z}, y)] - \max_{\tilde{z} \in \mathcal{Z}} \theta^T \phi(x, y, \tilde{z}). \quad (2.7)$$

This upper bound is not convex but is a difference of convex functions, that is, a sum of a convex function $f(\theta)$ and a concave function $g(\theta)$. This allows the LSSVM model to learn the parameters θ using the concave-convex procedure (CCCP) (Yuille and Rangarajan, 2002). The CCCP procedure first computes the gradient of the concave part at the current parameters θ^t, that is, $\nabla_\theta g|_{\theta^t}$ and obtains the next update θ^{t+1} by solving $\nabla_\theta f + \nabla_\theta g|_{\theta^t} = 0$. In the case of LSSVM, the first step of the CCCP procedure approximates the concave function by a linear upper bound. This corresponds to the assignment of values for the latent variable, as follows,

$$z = \arg\max_{\tilde{z} \in \mathcal{Z}} \theta^{tT} \phi(x, y, \tilde{z}).$$

The second step of the CCCP procedure results in solving the SSVM optimisation problem of equation (2.6), with the latent variable now considered as observed.

We will be interested in ERM methods in the weakly supervised setting in Chapter 3, and explore the strengths and weaknesses of existing models.

2.3 Deep probabilistic models

Deep neural networks are a very powerful tool to build highly expressive models that are able to capture complex distributions of structured data. Deep neural networks, and in particular, Convolutional Neural Networks (CNNs) are comprised of several convolutional layers, followed by one or more fully connected (dense) layers, interleaved by non-linear function(s) and (optionally) pooling. Since the first successful application of CNNs (LeCun et al., 1998), their use has greatly increased and diversified. In the probabilistic setting, where the goal is to capture the distribution $P(\boldsymbol{x})$ of the data via the model distribution $Q(\boldsymbol{x};\theta)$, deep neural networks are employed to represent the model's parameters θ. We refer to these models as deep probabilistic models.

We observe that deep probabilistic models separate into two types. While *explicit* models provide samples from a distribution and allow for the computation of the log-likelihood function, *implicit* models provide samples but the log-likelihood is not available. Both implicit and explicit models can be generative or discriminative. *Generative* models learn to capture the probability distribution of a random variable $P(\boldsymbol{x})$. We refer to *discriminative* models in the case where the distribution of interest is a conditional distribution $P(\boldsymbol{y}|x)$ of an output variable $\boldsymbol{y}$ given an input variable value x.

2.3.1 Law of the Unconscious Statistician (LOTUS)

Before diving into deep probabilistic models, we describe a key component of their learning procedure, the Law of the Unconscious Statistician (LOTUS) (Ross, 1970, p. 2). Let us consider a random variable $\boldsymbol{z}$ with distribution $P_z(\boldsymbol{z})$, and a measurable function g of $\boldsymbol{z}$. In this setting, $\boldsymbol{y} = g(\boldsymbol{z})$ is also a random variable, with distribution denoted $Q(\boldsymbol{y})$. The LOTUS states that we can compute the expected value of $g(\boldsymbol{z})$

as the integral of $g(\boldsymbol{z})$ with respect to $P_z(\boldsymbol{z})$, in other words:

$$\mathbb{E}_{y\sim Q(\boldsymbol{y})}[y] = \mathbb{E}_{z\sim P_z(\boldsymbol{z})}[g(z)]\,.$$

Let us now consider a random variable $\boldsymbol{y}$, which distribution $Q(\boldsymbol{y};\theta)$ is parametrised by some parameters θ. We assume that $\boldsymbol{y}$ can be expressed as a function g of $\boldsymbol{z}$ and θ, $\boldsymbol{y} = g(\boldsymbol{z};\theta)$, where $\boldsymbol{z}$ follows a simple, known distribution such as a uniform distribution between 0 and 1, denoted $\mathcal{U}([0,1])$. For example, if the distribution $Q(\boldsymbol{y};\theta)$ has an invertible cdf, then with g the inverse cdf of $Q(\boldsymbol{y};\theta)$ and $\boldsymbol{z} \sim \mathcal{U}([0,1])$ we can write $\boldsymbol{y} = g(\boldsymbol{z};\theta)$. Using the LOTUS, we have for any measurable function f:

$$\mathbb{E}_{y\sim Q(\boldsymbol{y};\theta)}[f(y)] = \mathbb{E}_{z\sim P_z(\boldsymbol{z})}[f(g(z;\theta))]\,.$$

If we need to compute the gradient with respect to θ, we note that the distribution $\boldsymbol{z}$ is independent of θ. Therefore one can easily compute the gradient as follows,

$$\nabla_\theta \mathbb{E}_{y\sim Q(\boldsymbol{y};\theta)}[f(y)] = \nabla_\theta \mathbb{E}_{z\sim P_z(\boldsymbol{z})}[f(g(z;\theta))] = \mathbb{E}_{z\sim P_z(\boldsymbol{z})}[\nabla_\theta f(g(z;\theta))]\,.$$

This is the key component to learn the parameters of deep probabilistic models. Deep probabilistic models employ an estimator of the gradient $\nabla_\theta \mathbb{E}_{y\sim Q(\boldsymbol{y};\theta)}[f(y)]$ computed using L samples from $P_z(\boldsymbol{z})$:

$$\widehat{\nabla_\theta \mathbb{E}_{y\sim Q(\boldsymbol{y};\theta)}[f(y)]} := \frac{1}{L}\sum_{l=1}^{L} \nabla_\theta f(g(z_l;\theta)).$$

Note that this reparametrisation means that in order to compute the estimator of the gradient with respect to the model parameters, we need to derive the gradient of $g(\boldsymbol{z};\theta)$. This poses a problem when the function g is a discontinuous funtion

mapping from a continuous random variable $\boldsymbol{z}$ to a discrete random variable $\boldsymbol{y}$. We address this challenge for deep probabilistic models in section 4.6.

2.3.2 Implicit deep probabilistic models

Implicit models do not explicitly compute the probability distribution of interest. These models allow the user to sample from this distribution by sampling a noise vector z from a known distribution $P_z(\boldsymbol{z})$. Among such models, Generative Adversarial Networks (GAN) (Goodfellow et al., 2014) are very popular and have been used in several computer vision applications (Denton et al., 2015; Radford et al., 2015; Springenberg, 2016). A GAN model consists of two neural networks, simultaneously trained in an adversarial manner. A generative model, referred to as the *Generator* G with parameters θ_G, is trained to replicate the data. Specifically, from a noise sample z the Generator outputs a data sample $x = G(z; \theta_G) \sim P_{\theta_G}$, where P_{θ_G} denotes the distribution of the samples generated by G. An adversarial discriminative model, referred to as the *Discriminator* D with parameters θ_D, is trained to identify whether a sample comes from the data distribution or has been generated by G. Specifically, the *Discriminator* returns the probability that a sample comes from the real data, rather than G. The GAN's training objective is based on a minimax game between the two networks, written as follows,

$$\min_{\theta_G} \max_{\theta_D} \mathbb{E}_{x\sim\tau(\boldsymbol{x})}[\log D(x;\theta_D)] + \mathbb{E}_{x\sim P_{\theta_G}(\boldsymbol{x})}[\log(1 - D(x;\theta_D))]\,,$$

where $\tau(\boldsymbol{x})$ is the data generating distribution. The GAN's objective can be rewritten, using the LOTUS discussed in section 2.3.1, as follows,

$$\min_{\theta_G} \max_{\theta_D} \mathbb{E}_{x\sim\tau(\boldsymbol{x})}[\log D(x;\theta_D)] + \mathbb{E}_{z\sim P_z(\boldsymbol{z})}[\log(1 - D(G(z,\theta_G);\theta_D))]\,.$$

The reparametrisation allows the parameters of both networks to be learned with gradient descent.

Another class of implicit model (Dziugaite et al., 2015; Li et al., 2015) trains generative deep networks with an objective function based on Maximum Mean Discrepancy (MMD) (Gretton et al., 2012), which measures the divergence or distance between two distributions. The MMD is used in statistical hypothesis testing to compare probability distributions. In this context, the MMD test can been seen as playing the role of an adversary.

The advantage of implicit models is that they do not require the density function of the distribution Q to have an analytic expression therefore are more flexible compared to explicit models. However, this means that the likelihood of the observations under the model is not directly computable.

2.3.3 Explicit deep probabilistic models

Contrary to implicit models, explicit deep probabilistic models require to access an analytic expression of the density function of the model's distribution. The most commonly employed explicit probabilistic model based on deep neural networks is the Variational Autoencoder (VAE) model (Kingma and Welling, 2014; Rezende et al., 2014). In the probabilistic framework, the VAE model assumes that the observed variable $\boldsymbol{x}$ is generated by a latent variable $\boldsymbol{z}$, via the *decoding* distribution $p(\boldsymbol{x}|\boldsymbol{z};\theta)$, but the posterior $p(\boldsymbol{z}|\boldsymbol{x};\theta)$ might be intractable. The prior distribution on the latent variable $p(\boldsymbol{z})$ is fixed and independent of θ. Using variational inference methods described in section 2.2.1, the VAE model approximates the intractable posterior with the *encoding* distribution $q(\boldsymbol{z}|\boldsymbol{x};\phi)$, where ϕ are the variational parameters. The VAE model performs amortised variational inference, that is, the observations parametrise the posterior distribution of the latent variable, and all observations share a single set of parameters ϕ. At test-time, this

allows efficient inference over the latent variable value for a new observation x by using the learned parameters ϕ. Both the encoding and decoding distributions are parametrised by neural networks. The VAE model is an explicit model and the distributions $q(\boldsymbol{z}|\boldsymbol{x};\phi), p(\boldsymbol{x}|\boldsymbol{z};\theta), p(\boldsymbol{z})$ have an analytic expression.

In the context of statistical estimation, we access the data distribution only via the training data set $\mathcal{D}$ consisting of n samples $\mathcal{D} = (x_1, \ldots, x_n)$. The VAE model assumes that the observed variables are independently, identically distributed (iid). The parameters θ, ϕ are learned by maximising the average Evidence Lower Bound (ELBO), presented in equation (2.2), of a single sample of the dataset $\mathcal{D}$, that is,

$$\mathcal{L}(\mathcal{D};\phi,\theta) := \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{z_i\sim q(\boldsymbol{z}_i|x_i;\phi)}[\log p(x_i|z_i;\theta)] - \mathrm{KL}(q(\boldsymbol{z}_i|x_i;\phi)||p(\boldsymbol{z}_i)).$$

In order to learn the parameters θ, ϕ the VAE model computes an estimate of the gradient of $\mathcal{L}(\mathcal{D};\phi,\theta)$. While the gradient with respect to θ is straightforward to compute, the gradient with respect to ϕ is problematic. Indeed, the expectations are computed with respect to $q(\boldsymbol{z}_i|x_i;\phi)$. In order to alleviate this issue, Kingma and Welling (2014); Rezende et al. (2014) propose the "reparametrisation trick". The idea is to reparametrise the variational approximation by a deterministic function g_ϕ which takes as input an auxiliary noise variable $\boldsymbol{\epsilon}$ with fixed distribution, and the observed variable $\boldsymbol{x}$. In order to draw a sample z from $q(\boldsymbol{z}_i|x_i;\phi)$, one draws a sample ϵ from $p(\boldsymbol{\epsilon})$ and computes

$$z_i = g_\phi(\epsilon, x_i). \tag{2.8}$$

Using the LOTUS, expectations with respect to $q(\boldsymbol{z}_i|x_i;\phi)$ can be rewritten as expectations with respect to $p(\boldsymbol{\epsilon})$. Similar to GAN, the VAE model has been extended in many research directions, several of which we will describe in this thesis. Moreover, the two models have been combined to retain the advantages of both methods,

that is efficient test-time inference of the VAE model and model flexibility of implicit models, see for example Makhzani et al. (2015); Mescheder et al. (2017). In this thesis, we will use implicit models in Chapter 4 to capture, given an input value, the probability distribution of an output. This will allow us to draw, for a given input, multiple samples from the modelled distribution. In Chapter 5 we will want to have an analytic expression for the modelled distribution, therefore we will employ a VAE-based model.

Chapter 3

A Unified Framework for latent structured output prediction

In this chapter, we focus on the problem of structured output prediction in the weakly supervised setting. We study several generalisations of the popular Latent Structural Support Vector Machine (LSSVM) model (Felzenszwalb et al., 2008; Yu and Joachims, 2009) presented in section 2.2.2. These generalisations all fall under the umbrella of Empirical Risk Minimisation (ERM) methods, but they differ in their prediction criterion, that is, the function returning the output given the input and model's parameters. Specifically, their prediction criteria can be divided into two categories. The first category uses marginalisation over the latent variable, while the second category uses entropy as an uncertainty measure on the value of the latent variable. Our goal in this chapter is to perform a fair comparison between these methods to evaluate the benefits of taking into account the uncertainty in the latent variable. However, a comparison between the prediction criterion of each method is not directly feasible, as each of them has its own optimisation procedure. Therefore, we propose a novel prediction criterion, which includes as special cases

previous prediction criteria that have been used in the literature, and employ a unique optimisation algorithm to learn the parameters of our model. Specifically, our framework's prediction criterion minimises the Aczél and Daròczy entropy of the output. The main contributions of our method are (i) We design a learning objective that provides a Unified Framework (UF) for structured output prediction in the presence of latent variables; (ii) We develop a single optimisation algorithm, which in turns allows us to compare the prediction criteria of the aforementioned generalisations by discounting any variability that may arise due to differences in their optimisation algorithm; (iii) We empirically show that the proposed algorithm is as effective as the more complex approaches that have been previously employed for latent structured output prediction; (iv) We provide empirical evidence that lends support to prediction via the minimisation of the latent space uncertainty.

3.1 Introduction

Structured output prediction methods (BakIr et al., 2007; Nowozin and Lampert, 2011) have gained popularity in computer vision due to their ability to provide an elegant formulation for systems that perform various important visual tasks such as object detection (Blaschko and Lampert, 2008) or semantic segmentation (Kumar et al., 2011). However, in the supervised setting these methods assume that training data are fully labelled.

In real applications it can be very difficult to obtain the full annotation of a data set. Latent variable models handle the missing information with the introduction of latent variables. We describe in section 2.2 two types of method to perform structured output prediction in the presence of latent variables, namely probabilistic methods and empirical risk minimisation methods. With probabilistic methods the parameters are estimated by maximising the marginalised data log-likelihood.

One famous example is the Expectation-Maximization (EM) algorithm (Dempster et al., 1977; Rolf, 1974). EM and its variants (Gelman et al., 1995; Neal and Hinton, 1998; Salojärvi et al., 2005) have been widely used to learn the parameters of latent variable models. In our case, we are primarily interested in the second type of method, that is, Empirical Risk Minimisation (ERM) methods, where the parameters are learned by minimising the regularised empirical risk. The risk is measured by a user-specified loss function. Indeed, our goal throughout this thesis is to perform a task, in this case predict an output, while taking into account its associated evaluation loss.

We present in section 2.2.2 the most commonly employed ERM method for structured output prediction in presence of latent variables, that is, the Latent Structural Support Vector Machine (LSSVM) model (Felzenszwalb et al., 2008; Yu and Joachims, 2009). Its prediction criterion maximises the joint posterior probability over the output and latent variables values. Its learning objective is an upper bound on a user-specified loss function that provides a measure of the prediction risk. Due to its simplicity and computational efficiency, the LSSVM formulation has gained popularity among the computer vision community with several applications such as object detection (Felzenszwalb et al., 2008; Zhu et al., 2010), image segmentation (Kumar et al., 2011), indoor scene interpretation (Wang et al., 2010) and action classification (Behl et al., 2014). Recently, several generalisations of the LSSVM model have been proposed. While the generalised frameworks share the common characteristic that their parameters are estimated by minimising the prediction risk, they differ from each other in the prediction criterion. Specifically, these methods can be separated into two categories. The first category predicts the output while marginalising over the latent variable or setting it to its most likely value, and the second performs prediction while minimising the uncertainty in the latent variable by using an entropy-based uncertainty measure.

To the best of our knowledge, there is no existing fair evaluation of the strengths and weaknesses of the LSSVM framework and its generalisations. We propose a natural Unified Framework (UF) that contains the aforementioned loss-based latent structured output prediction frameworks as special cases. Our model performs inference over the output variable while minimising the uncertainty in the latent variable as measured by the Aczél and Daròczy (AD) entropy. We derive an optimisation procedure for learning the parameters of the UF based on the concave convex procedure (CCCP) (Yuille and Rangarajan, 2002). This allows us to compare the prediction criterion of LSSVM and its generalisations by discounting any variability that may arise due to differences in their optimisation algorithm. We test the UF, LSSVM, and LSSVM's generalisations on the task of multi-class gesture recognition from video sequences and action recognition in images. Our experiments convincingly demonstrate that, for large and ambiguous latent spaces, an entropy-based prediction criterion provides more accurate results.

3.2 Generalisations of the LSSVM model

Recent years have witnessed the development of several generalisations of LSSVM. While all these frameworks estimate the parameters by minimising an upper bound on the empirical risk, they differ greatly in their prediction criterion and can be divided into two categories. The first category marginalises the joint probability of the output and latent variables, over the latent variable. Schwing et al. (2012) introduce a temperature parameter $\epsilon \in (0, 1]$ and propose a family of models which we refer to as the ϵ-framework. The prediction criterion of the ϵ-framework ranges from maximising the joint probability of the output and latent variables (by setting ϵ to the limit value of 0, denoted $\epsilon \searrow 0$, recovering the LSSVM model) to the marginalisation over the latent variable values (by setting $\epsilon = 1$). Ping et al. (2014) present

marginal structural support vector machines (MSSVM). Similar to the ϵ-framework, the MSSVM prediction criterion marginalises the joint probability to estimate the probability of the output variable. Ping et al. (2014) also describe a more general framework with temperature parameters that includes LSSVM, MSSVM and the ϵ-framework as special cases. The second category uses entropy as an uncertainty measure on the value of the latent variable. Specifically, Miller et al. (2012) propose the max-margin min-entropy (M3E) family of models. M3E models account for uncertainty in the latent variable by predicting the output value with minimum Rényi entropy.

There have been limited experimental results reported in the literature that compare the merits of each of these methods. Moreover, even in the limited experiments, it is difficult to assess whether the reported improvements of one method over the other are due to a better learning objective or a better optimisation algorithm. We define a simple prediction criterion that allows us to construct a Unified Framework (UF) for loss-based latent structured output prediction. Our UF recovers all the aforementioned frameworks that are LSSVM, MSSVM, the ϵ-framework and M3E as special cases. Furthermore, it also exposes an extra-degree of freedom that has not yet been explored. We develop an optimisation procedure and propose a single simple algorithm for learning the parameters of the UF. With this setting we are able to compare the prediction criterion of the cited models by discounting the differences that could arise from using different optimisation algorithms.

3.3 The Unified Framework

In order to facilitate our exploration of the performance of the various generalisations of LSSVM, we would like to obtain a unified criterion for prediction that captures the previously used criteria. In other words, our criterion should include

as special cases, (i) the maximisation over the output and latent variables values, to recover the LSSVM model; (ii) the maximisation over the output variable value after marginalising the joint probability over the latent variable, to recover the MSSVM and ϵ-framework; and (iii) the maximisation over the output variable while minimising the uncertainty in the distribution of the latent variable, to include the M3E models. Furthermore, we would also like to design a learning objective that minimises an upper bound on the empirical risk based on the prediction criterion, where the risk is measured by a user-defined loss function. We begin by showing that the Aczél and Daròczy (AD) entropy provides a suitable prediction criterion that meets the aforementioned requirements. The identification of the AD entropy as our prediction criterion will allow us to develop a suitable learning objective.

3.3.1 Preliminaries

Notations. We denote the input variable by $\boldsymbol{x}$, taking values in $\mathcal{X}$, the output variable by $\boldsymbol{y}$, taking values in $\mathcal{Y}$ and the latent variable by $\boldsymbol{z}$, taking values in $\mathcal{Z}$. We further assume that $\boldsymbol{y}$ and $\boldsymbol{z}$ are discrete, that is, the spaces $\mathcal{Y}$ and $\mathcal{Z}$ are finite and countable. The value of $\boldsymbol{x}$ is known during both training and testing, the value of $\boldsymbol{y}$ is known only during training and the value of $\boldsymbol{z}$ is unknown during both training and testing. Recall the example of action recognition in images in Figure 2.1. We have access to the input image x both at training and testing, we know the ground-truth label of the action performed y only at training, and we never have knowledge of the coordinates of the bounding box containing the action. This is the weakly supervised setting, and the latent variable $\boldsymbol{z}$ is introduced to represent the missing annotation on the bounding box. We assume $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ follows a conditional model:

$$p(\boldsymbol{y} = y, \boldsymbol{z} = z | \boldsymbol{x} = x; \theta) := p(y, z | x; \theta) \propto \exp\big(\frac{1}{\epsilon_z} \theta^T \phi(x, y, z)\big), \tag{3.1}$$

where $\phi : (\mathcal{X}, \mathcal{Y}, \mathcal{Z}) \to \mathbb{R}^D$ refers to the joint feature vector of the input, the output and the latent variables, and $\theta \in \mathbb{R}^D$ are the model parameters. We introduce the temperature parameter $\epsilon_z \in (0, 1]$ in the expression of the joint probability of the output and latent variables (note that ϵ_z is a scalar parameter of the model, and does not depend on z). The use of ϵ_z allows our prediction criterion to range from assigning the latent variable to its most likely value when $\epsilon_z \searrow 0$ to the marginalisation over the latent variable when $\epsilon_z = 1$. It follows that

$$p(\boldsymbol{y} = y|\boldsymbol{x} = x; \theta) := p(y|x; \theta) \propto \sum_{z \in \mathcal{Z}} \exp\big(\frac{1}{\epsilon_z}\theta^T \phi(x, y, z)\big).$$

We use the following shorthand notations:

- The notation p_x refers to $p(\boldsymbol{y}, \boldsymbol{z}|x; \theta)$, the joint conditional probability of the output and latent variables $(\boldsymbol{y}, \boldsymbol{z})$ given a value of the input variable $\boldsymbol{x} = x$ and model parameters θ.
- The notation q_x^y refers to $p(y, \boldsymbol{z}|x; \theta)$, that is, $p(\boldsymbol{y}, \boldsymbol{z}|x; \theta)$ for a specific value $\boldsymbol{y} = y$, therefore q_x^y is a probability distribution of the latent variable $\boldsymbol{z}$. Note that q_x^y is a generalised distribution, that is, it needs not sum to one but to $p(y|x; \theta)$. In other words, q_x^y is not normalised by the normalisation constant $p(y|x; \theta) = \sum_{z \in \mathcal{Z}} p(y, z|x; \theta) = \sum_{z \in \mathcal{Z}} q_x^y(z)$.
- The notation p_x^y refers to $p(\boldsymbol{z}|y, x; \theta)$, the conditional distribution of the latent variable, given the input and output variables values and the model parameters.

Aczél and Daròczy entropy. To measure uncertainty we will use the Aczél and Daròczy (AD) entropy (Aczél and Daróczy, 1963), parametrised by the scalars α

and β. The AD entropy of the generalised distribution is

$$H_{\alpha,\beta}(q_x^y;\theta) = \frac{1}{1-\alpha}\log\Big(\frac{\sum\limits_{z\in\mathcal{Z}} p(y,z|x;\theta)^{\alpha+\beta-1}}{\sum\limits_{z\in\mathcal{Z}} p(y,z|x;\theta)^{\beta}}\Big),$$

with $\alpha \geq 0, \alpha \neq 1, \beta \geq 0, \alpha+\beta-1 \geq 0$. The AD entropy has been studied by Aczél and Daròczy (Aczél and Daróczy, 1963) as a natural generalisation of the Rényi entropy with an extra scalar parameter β. The AD entropy has since been shown to create a natural family of uncertainty measures, also recovering other existing entropy functions as special cases (Esteban and Morales, 1995; Mathai and Rathie, 1975).

Let us take a closer look at some interesting special cases of the AD entropy. Specifically when $\beta = 1$, the AD entropy is equivalent to the Rényi entropy used in the M3E models,

$$H_{\alpha,1}(q_x^y;\theta) = \frac{1}{1-\alpha}\log\Big(\frac{\sum\limits_{z\in\mathcal{Z}} p(y,z|x;\theta)^{\alpha}}{\sum\limits_{z\in\mathcal{Z}} p(y,z|x;\theta)}\Big).$$

When $\beta = 1$ and $\alpha \to \infty$, the AD entropy is equivalent to the minimum entropy,

$$H_{\infty,1}(q_x^y;\theta) = -\log\max_{z\in\mathcal{Z}} p(y,z|x;\theta).$$

As $\beta = 1$ the AD entropy is equivalent to the Rényi entropy, when $\alpha \to 1$ the AD entropy is equivalent to the commonly used Shannon entropy (see Bromiley et al. (2010) for a proof),

$$H_{1,1}(q_x^y;\theta) = -\frac{\sum\limits_{z\in Z} p(y,z|x;\theta)\log p(y,z|x;\theta)}{\sum\limits_{z\in\mathcal{Z}} p(y,z|x;\theta)}.$$

When $\beta = 0$ and $\alpha = 2$, the AD entropy is equivalent to the marginalisation

over the latent variable,

$$H_{2,0}(q_x^y;\theta) = -\log \sum_{z\in\mathcal{Z}} p(y,z|x;\theta) + K,$$

where K is a constant. We will use these special cases to show how we recover existing prediction criteria in the next section.

3.3.2 A common prediction criterion

We propose to predict the output variable for an input x by taking the output value with minimum AD entropy of q_x^y, that is,

$$y(\theta) = \arg\min_{y\in\mathcal{Y}} H_{\alpha,\beta}(q_x^y;\theta). \tag{3.2}$$

Again, let's look at special cases of this prediction procedure. When $\beta = 1$, the prediction recovers the prediction task of M3E models,

$$y(\theta) = \arg\min_{y\in\mathcal{Y}} H_{\alpha,1}(q_x^y;\theta).$$

Since M3E itself generalises LSSVM, it follows that our prediction criterion also includes LSSVM as a special case. Specifically, with $\beta = 1$ and $\alpha \to \infty$, the prediction recovers the prediction task of LSSVM by performing maximum a posteriori prediction, that is,

$$y(\theta) = \arg\max_{y\in\mathcal{Y}} \max_{z\in\mathcal{Z}} \log p(y,z|x;\theta).$$

Similarly when $\beta = 0$ and $\alpha = 2$, the prediction task is equivalent to the maximisation, over the output variable, of the marginalised joint probability. This is the

prediction setting of MSSVM and ϵ-framework, that is,

$$y(\theta) = \arg\max_{y \in \mathcal{Y}} \log \sum_{z \in \mathcal{Z}} p(y, z|x; \theta). \tag{3.3}$$

To illustrate the prediction process of each existing model, we consider again the weakly annotated image of Figures 1.3 and 2.1. As shown in Figure 3.1, the value y of the output variable is the class of the action, that is, "riding horse". The value z of the latent variable represents the bounding box location, with the associated joint probability $p(y, z|x; \theta)$ (not including the temperature parameter). Recall that these are generalised probabilities hence do not have to sum to one, but to $p(y|x; \theta)$. In this example, to predict y, the LSSVM model computes the log-probability of the highest scoring bounding box, that is, $\log(0.5)$. The MSSVM model computes the logarithm of the marginalised probability, that is, $\log(0.5 + 0.2 + 0.1)$. The ϵ-framework does the same with the temperature parameter ϵ, that is, $\log(0.5^{1/\epsilon} + 0.2^{1/\epsilon} + 0.1^{1/\epsilon})$. The M3E model prediction criterion

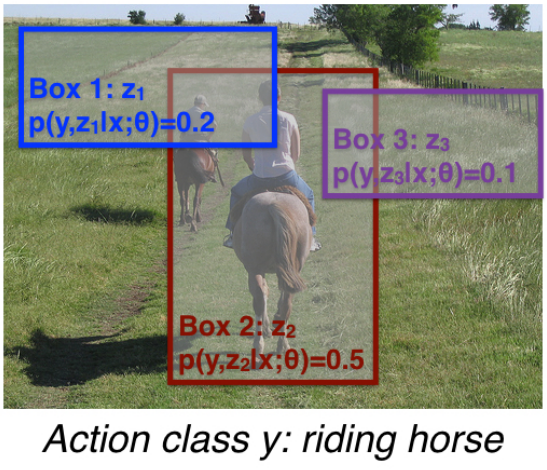


Figure 3.1: The latent variable $\boldsymbol{z}$ represents the missing information, with corresponding joint probabilities. The image is from the Pascal VOC 2011 data set (Everingham et al., 2010).

requires the computation of the Rényi entropy, that is,

$$\frac{1}{1-\alpha}\log\frac{0.5^{\alpha}+0.2^{\alpha}+0.1^{\alpha}}{0.5+0.2+0.1}.$$

Our UF prediction criterion computes the AD entropy, with the temperature parameter ϵ_z, that is,

$$\frac{1}{1-\alpha}\log\frac{0.5^{(\alpha+\beta-1)/\epsilon_z}+0.2^{(\alpha+\beta-1)/\epsilon_z}+0.1^{(\alpha+\beta-1)/\epsilon_z}}{0.5^{\beta/\epsilon_z}+0.2^{\beta/\epsilon_z}+0.1^{\beta/\epsilon_z}}.$$

The following proposition sheds further light on our prediction criterion.

Proposition 3.1 (Decomposition of the AD entropy)**.** *Given an input x, the AD entropy of q_x^y, the generalised distribution of $\boldsymbol{z}$ for a value y, can be written as the sum of the negative log-likelihood of y and the AD entropy of p_x^y, the conditional distribution of the latent variable given the output and input,*

$$H_{\alpha,\beta}(q_x^y;\theta) = -\log p(y|x;\theta) + H_{\alpha,\beta}(p_x^y;\theta). \tag{3.4}$$

Proof. See appendix 7.4.1. □

Proposition 3.1 shows that performing prediction by minimising the AD entropy is equivalent to predicting the output y which (i) has a high probability, and (ii) minimises the uncertainty in the latent variable $\boldsymbol{z}$. Specifically when $\beta = 0$ and $\alpha = 2$, the term $H_{\alpha,\beta}(p_x^y;\theta)$ in equation (3.4) disappears. This means that we do not minimise the uncertainty in the distribution of $\boldsymbol{z}$ and equation (3.2) and equation (3.3) are equivalent. This is another way to see how we recover the prediction procedure of the MSSVM and ϵ-framework models.

3.3.3 A common learning objective

In the context of statistical estimation, we consider a training data set of input-output pairs $\mathcal{D} = \{(x_i, y_i), i = 1, \ldots, n\}$. We wish to learn the parameters of the model θ, to be able to predict the output for any input value x. We introduce the loss function $\Delta(y_i, y),\ (y_i, y) \in \mathcal{Y} \times \mathcal{Y}$ with $\Delta(y_i, y_i) = 0$ that compares the risk of making the prediction y for the input x_i with ground-truth output y_i. The parameters of the model are learned by minimising the following objective function.

$$\min_{\theta \in \Theta} \frac{1}{2}||\theta||^2 + \frac{C}{n}\sum_{i=1}^{n}\Big[\epsilon_y \log \sum_{y \in \mathcal{Y}} \exp \frac{1}{\epsilon_y}\Big(\Delta(y_i, y) - \epsilon_z H_{\alpha,\beta}(q^y_{x_i}; \theta)\Big) + \epsilon_z H_{\alpha,\beta}(q^{y_i}_{x_i}; \theta)\Big]. \tag{3.5}$$

We introduce regularisation over the parameters of the model θ to avoid overfitting the parameters to the training data. We employ the log-sum-exp function and we introduce the temperature parameter $\epsilon_y \in (0, 1]$ in order to recover the objective function of LSSVM and its extension, as shown below.

Recovering existing models. Specifically when $\beta = 1, \epsilon_y \searrow 0, \epsilon_z = 1$ we retrieve the objective function used in training of the M3E models, which maximises the margin between the entropy of the ground-truth output value and all other outputs,

$$\min_{\theta \in \Theta} \quad \frac{1}{2}||\theta||^2 + \frac{C}{n}\sum_{i=1}^{n}\max_{y \in \mathcal{Y}}\Big(\Delta(y_i, y) - H_{\alpha,1}(q^y_{x_i}; \theta)\Big) + H_{\alpha,1}(q^{y_i}_{x_i}; \theta).$$

From the M3E models, taking $\alpha \rightarrow \infty$ recovers LSSVM. Thus when $\beta = 1, \epsilon_y \searrow 0, \epsilon_z = 1, \alpha \rightarrow \infty$ our learning procedure is equivalent to

the learning problem of the LSSVM,

$$\min_{\theta\in\Theta} \quad \frac{1}{2}||\theta||^2 + \frac{C}{n}\sum_{i=1}^{n} \max_{y,z\in\mathcal{Y}\times\mathcal{Z}} \Big(\Delta(y_i,y) + \theta^T\phi(x_i,y,z)\Big) - \frac{C}{n}\sum_{i=1}^{n} \max_{z\in\mathcal{Z}} \theta^T\phi(x_i,y_i,z).$$

When $\beta = 0, \alpha = 2, \epsilon_y \searrow 0, \epsilon_z = 1$, the log-sum-exp function over y approximates the maximum function and we maximise over the values of the output y while marginalising over the latent variable values z. Thus equation (3.5) becomes equivalent to the optimisation problem of the MSSVM,

$$\begin{aligned}\min_{\theta\in\Theta} \quad & \frac{1}{2}||\theta||^2 + \frac{C}{n}\sum_{i=1}^{n} \max_{y\in\mathcal{Y}} \Big(\Delta(y_i,y) + \log\sum_{z\in\mathcal{Z}} \exp(\theta^T\phi(x_i,y,z))\Big) \\ & - \frac{C}{n}\sum_{i=1}^{n} \log\sum_{z\in\mathcal{Z}} \exp\big(\theta^T\phi(x_i,y_i,z)\big).\end{aligned}$$

Similarly when $\beta = 0, \alpha = 2, \epsilon_y = \epsilon_z$ we recover the same objective to minimise as in the ϵ-framework, that is

$$\begin{aligned}\min_{\theta\in\Theta} \quad & \frac{1}{2}||\theta||^2 + \frac{C}{n}\sum_{i=1}^{n} \epsilon\log\sum_{y,z\in\mathcal{Y}\times\mathcal{Z}} \exp\Big(\frac{\Delta(y_i,y) + \theta^T\phi(x_i,y,z)}{\epsilon}\Big) \\ & - \frac{C}{n}\sum_{i=1}^{n} \epsilon\log\sum_{z\in\mathcal{Z}} \exp\Big(\frac{\theta^T\phi(x_i,y_i,z)}{\epsilon}\Big).\end{aligned}$$

Figure 3.2 shows that our framework gathers all models presented in section 3.2 with specific parameters values. Figure 7.1 in appendix 7.4.4 shows the equivalence in more details.

A max-margin formulation. When $\epsilon_y \searrow 0$, the log-sum-exp in equation (3.5) results in the following maximisation

$$\max_{y\in\mathcal{Y}} \Big(\Delta(y_i,y) - \epsilon_z H_{\alpha,\beta}(q^y_{x_i};\theta)\Big) + \epsilon_z H_{\alpha,\beta}(q^{y_i}_{x_i};\theta),$$

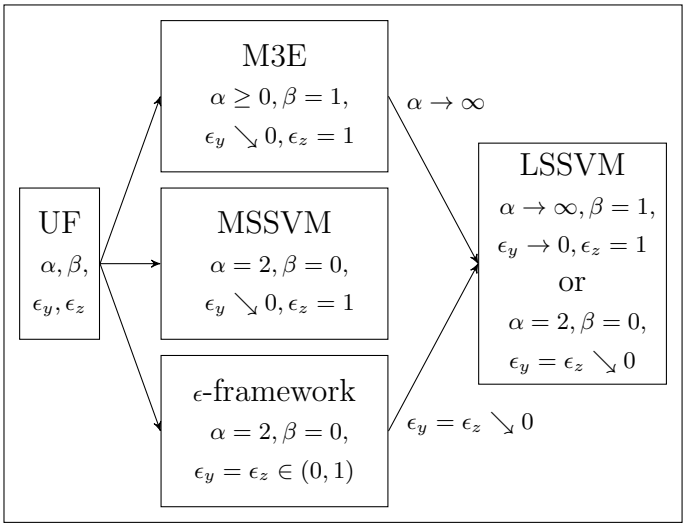


Figure 3.2: Equivalence of the UF with existing ERM methods.

or equivalently

$$\max_{y \in \mathcal{Y}} \Big(\Delta(y_i, y) - \epsilon_z H_{\alpha,\beta}(q^y_{x_i}; \theta) + \epsilon_z H_{\alpha,\beta}(q^{y_i}_{x_i}; \theta) \Big).$$

Note that we assume $\Delta(y_i, y) \geq 0$, $\Delta(y_i, y_i) = 0$, therefore we have $\max_{y \in \mathcal{Y}} \Big(\Delta(y_i, y) - \epsilon_z H_{\alpha,\beta}(q^y_{x_i}; \theta) + \epsilon_z H_{\alpha,\beta}(q^{y_i}_{x_i}; \theta) \Big) \geq 0$. By minimising the objective of equation (3.5), we enforce that this maximum is attained at 0. This boils down to the following requirement, $\forall y \in \mathcal{Y},\ y \neq y_i$,

$$\forall y \in \mathcal{Y},\ y \neq y_i,\ \Delta(y_i, y) - \epsilon_z H_{\alpha,\beta}(q^y_{x_i}; \theta) + \epsilon_z H_{\alpha,\beta}(q^{y_i}_{x_i}; \theta) \leq 0,$$

or equivalently,

$$\forall y \in \mathcal{Y},\ y \neq y_i,\ \Delta(y_i, y) \leq \epsilon_z H_{\alpha,\beta}(q^y_{x_i}; \theta) - \epsilon_z H_{\alpha,\beta}(q^{y_i}_{x_i}; \theta). \tag{3.6}$$

Equation (3.6) enforces that the AD entropy of the ground-truth output, $H_{\alpha,\beta}(q^{y_i}_{x_i}; \theta)$ is smaller than the AD entropy of all other outputs, by a margin defined by $\Delta(y_i, y)$. This margin approach is also presented in the LSSVM model. For other values of ϵ_y, the objective function replaces the maximisation by a softmax (with the log-sum-exp function).

An upper-bound on the loss. Our objective function in equation (3.5) naturally derives from our prediction procedure, and it allows us to upper bound the user-defined loss. Indeed, Proposition 3.2 shows that the optimisation problem of equation (3.5) minimises an upper bound on the user-defined loss.

Proposition 3.2 (Upper bound on the loss)**.** *The optimisation problem of equation (3.5) minimises an upper bound on the loss* $\frac{C}{n}\sum_{i=1}^{n}\Delta(y_i, y_i(\theta))$ *where* y_i *is the ground-truth output for the training input* x_i *and* $y_i(\theta)$ *is the predicted output for* x_i*. This upper bound is tightest when* $\epsilon_y \searrow 0$*.*

Proof. See appendix 7.4.2 □

3.4 Optimisation procedure

We propose to use a common algorithm for all methods covered by the UF. Our algorithm is as computationally efficient as the existing procedures employed by the models we unify. This allows us to compare the prediction criterion of all these models independently of their specific method for solving their optimisation problem. We derive an optimisation method for learning the parameters of the UF. This procedure works for every setting of parameters of the UF. In other words it works for every model presented in section 3.2.

The optimisation function in equation (3.5) is not convex. In order to obtain an approximate solution, we write our objective function as a difference of convex (DC) functions (Yuille and Rangarajan, 2002). To this end, it would be helpful to introduce the following shorthand notations in the case $\alpha > 1$,

- $F^{+}_{\alpha,\beta}(y,\theta) = -\frac{\epsilon_z}{1-\alpha}\log\sum_{z\in\mathcal{Z}} p(y,z|x_i;\theta)^{\alpha+\beta-1}$,
- $F^{-}_{\alpha,\beta}(y,\theta) = -\frac{\epsilon_z}{1-\alpha}\log\sum_{z\in\mathcal{Z}} p(y,z|x_i;\theta)^{\beta}$,
- $G^{+}_{\alpha,\beta}(y_i,\theta) = -\frac{\epsilon_z}{1-\alpha}\log\sum_{z\in Z} p(y_i,z|x_i;\theta)^{\beta}$

- $G^{-}_{\alpha,\beta}(y_i,\theta) = -\dfrac{\epsilon_z}{1-\alpha}\log\sum_{z\in\mathcal{Z}} p(y_i,z|x_i;\theta)^{\alpha+\beta-1}$.

We show in the proof of Proposition 3.3 in appendix 7.4.3) that given our conditional model of equation (3.1), the functions $F^{+}_{\alpha,\beta}(y,\theta), F^{-}_{\alpha,\beta}(y,\theta), G^{+}_{\alpha,\beta}(y_i,\theta), G^{-}_{\alpha,\beta}(y_i,\theta)$ are convex with respect to θ. In the case $\alpha < 1$, their definitions will be similar (see Proof of Proposition 3.3 in appendix 7.4.3).

Proposition 3.3 (Optimisation problem as a DC)**.** *The optimisation problem in equation (3.5) can be equivalently written as a difference of convex (DC) functions for any values of* $\alpha \geq 0, \alpha \neq 1, \beta \geq 0, \alpha+\beta-1 \geq 0$ *using the following formulation,*

$$\begin{aligned}\min_{\theta\in\Theta} \quad & \frac{1}{2}||\theta||^2 + \frac{C}{n}\sum_{i=1}^{n}\Big[\epsilon_y \log\sum_{y\in\mathcal{Y}}\exp\frac{1}{\epsilon_y}\Big(\Delta(y_i,y) \\ & + F^{+}_{\alpha,\beta}(y,\theta) - F^{-}_{\alpha,\beta}(y,\theta)\Big) \\ & + G^{+}_{\alpha,\beta}(y_i,\theta) - G^{-}_{\alpha,\beta}(y_i,\theta)\Big],\end{aligned}$$

with $F^{+}_{\alpha,\beta}(y,\theta), F^{-}_{\alpha,\beta}(y,\theta), G^{+}_{\alpha,\beta}(y_i,\theta), G^{-}_{\alpha,\beta}(y_i,\theta)$ *all convex with respect to* θ*.*

Proof. See appendix 7.4.3 □

This results in the Algorithm 1 for training the UF. Algorith 1 is similar to standard concave convex procedure (CCCP) (Yuille and Rangarajan, 2002). During step 3 of Algorithm 1, we solve the convex optimisation problem of equation (3.7):

$$\begin{aligned}\theta^{t+1} = \arg\min_{\theta} & \frac{1}{2}||\theta||^2 + \frac{C}{n}\sum_{i=1}^{n}\Big[\epsilon_y \log\sum_{y\in\mathcal{Y}}\exp\frac{1}{\epsilon_y}\Big(\Delta(y_i,y) \\ & + F^{+}_{\alpha,\beta}(y,\theta) - T^{F^{-}_{\alpha,\beta}}_{y,\theta^t}(\theta)\Big) \\ & + G^{+}_{\alpha,\beta}(y_i,\theta) - T^{G^{-}_{\alpha,\beta}}_{y_i,\theta^t}(\theta)\Big],\end{aligned} \tag{3.7}$$

where $F^{-}_{\alpha,\beta}(y,\theta)$ and $G^{-}_{\alpha,\beta}(y_i,\theta)$ are replaced by their first order Taylor expansion:

$$T^{F^{-}_{\alpha,\beta}}_{y,\theta^t}(\theta) = F_{\alpha,\beta}(y,\theta^t)^{-} + (\theta-\theta^t)^T\nabla_\theta F^{-}_{\alpha,\beta}(y,\theta)|_{\theta^t},$$
$$T^{G^{-}_{\alpha,\beta}}_{y_i,\theta^t}(\theta) = G_{\alpha,\beta}(y_i,\theta^t)^{-} + (\theta-\theta^t)^T\nabla_\theta G^{-}_{\alpha,\beta}(y_i,\theta)|_{\theta^t}.$$

We denote by $\nabla_\theta F^{-}_{\alpha,\beta}(y,\theta)|_{\theta^t}$ the gradient of $F^{-}_{\alpha,\beta}(y,\theta)$ with respect to θ estimated at θ^t and similarly $\nabla_\theta G^{-}_{\alpha,\beta}(y_i,\theta)|_{\theta^t}$. We solve the optimisation problem of equation (3.7) in step 3 of Algorithm 1 by performing gradient descent, with a step size found by line search procedure shown in Algorithm 5 in appendix 7.4.5. We stop the training when the objective has converged, which is estimated by the stopping criterion $0 \leq \delta_{obj} < C\lambda$ ($\lambda = 0.01$ in our experiments).

Algorithm 1: Unified Framework training algorithm.

1 Initialise $\theta = \theta_0, t = 0$.
2 Define $\text{obj}(\theta_0) = \infty$ and

$$\text{obj}(\theta,\theta^t) = \frac{1}{2}||\theta||^2 + \frac{C}{n}\sum_{i=1}^{n}\Big[\epsilon_y \log\sum_{y\in\mathcal{Y}}\exp\frac{1}{\epsilon_y}\Big(\Delta(y_i,y) + F^{+}_{\alpha,\beta}(y,\theta) - T^{F^{-}_{\alpha,\beta}}_{y,\theta^t}(\theta)\Big) + G^{+}_{\alpha,\beta}(y_i,\theta) - T^{G^{-}_{\alpha,\beta}}_{y_i,\theta^t}(\theta)\Big].$$

while $t \leq T$ *and not* $0 \leq \delta_{obj} < C\lambda$ **do**
3 $\theta^{t+1} \leftarrow \arg\min_{\theta} \text{obj}(\theta,\theta^t)$ by gradient descent.
4 $\delta_{obj} \leftarrow \text{obj}(\theta^t,\theta^{t-1}) - \text{obj}(\theta^{t+1},\theta^t)$.
5 $t \leftarrow t+1$.
end

3.5 Experiments

We perform comparison between the UF, LSSVM, MSSVM, the ϵ-framework, M3E models and the replications of these models by the UF. By replication we mean that we employ the UF and its optimisation procedure with parameters $\alpha, \beta, \epsilon_z$

set to replicate the existing models. In all figures, the sign $\sim$ means that the UF's parameters were set to replicate an existing model. Recall that we separate the models in two categories, one category using marginalisation over the latent variable and one category predicting the output with minimum entropy over the latent variable. Our goal is to assess which is the most accurate prediction criterion between the two categories, regardless of their specific optimisation algorithm. To this end we perform two tasks. The first task is binary action classification with a small latent space size, with little uncertainty in the latent variable value. The second task is multi-class gesture recognition in video sequences, in this case the uncertainty in the value of the latent variable is high and the latent space is large. For the UF, we employ $\epsilon_y \searrow 0$, that is, maximisation over the output variable value, which provides the tightest bound. Note that when we train the ϵ-framework with its own optimisation algorithm we employ $\epsilon \neq 0$.

3.5.1 Binary action classification

We start our empirical comparison on an example where the latent space size is small and the uncertainty in the latent variable is reduced. Specifically, we perform the following experiment of binary action classification over 10 classes of the PASCAL VOC 2011 data set.

Pascal VOC data set. We use the "trainval" data set of the PASCAL VOC 2011 (Everingham et al., 2010) action classification data set, consisting of 2424 images depicting 10 action classes. For each image we are provided the bounding boxes of the persons in the image and its action class. However, in our experiments, we discard the bounding box information and instead model it using a latent variable.

Modeling and features. The score of a bounding box in the image x is $\theta^T \phi(x, y, z) = \theta_y^T \phi(x, z)$ where θ_y are the parameters that correspond to the label y

and $\phi(x, z)$ is the feature vector extracted from the bounding box z. Similar to Behl et al. (2014), we consider the bounding boxes of the image with the top 20 scores found by a standard person detector (Felzenszwalb et al., 2010). Thus we reduce the uncertainty in the latent space since we take only the top scoring bounding boxes and we work with a small latent space size. The feature vector $\phi(x, z)$ consists of the standard poselet-based feature vector (Maji et al., 2011), that is, a 2404-dimensional vector consisting of 2400 activation scores of action specific poselets and 4 object activation scores. We added the score given by the person detector, making $\phi(x, z)$ a 2405-dimensional feature vector. The loss Δ is the standard 0-1 loss.

Experimental setting. We split the data set into 1940 training images and 484 validation images. Hyper parameters are chosen via 5 folds cross-validation. We use four random seeds values in order to mitigate possible effects of the initialisation; this is to ensure that the non-convexity of the optimisation objective does not lead to poor results by local minima. For each fold, we report the test error corresponding to the seed with the lowest training objective value. We refer the reader to appendix 7.4.6 for additional details regarding the experimental setup and the results.

Results. Figure 3.3 shows the mean loss on the testing set over the 10 classes of each model with best cross-validated parameters (averaged over the 5 folds).

We see that the performance of the UF as a replication and the corresponding existing model are similar. Predicting the output by marginalising the output and latent variables as done by MSSVM is the less accurate criterion. Our UF and M3E models provide slightly better performance than LSSVM, MSSVM and the ϵ-framework. We do not report the best parameters chosen by cross-validation but they are available in Tables 7.2 and 7.3. In most cases the set of UF parameters chosen by cross-validation boils down to a prediction criterion that maximises over the output and latent variables (see our analysis in appendix 7.4.6). Similarly, the

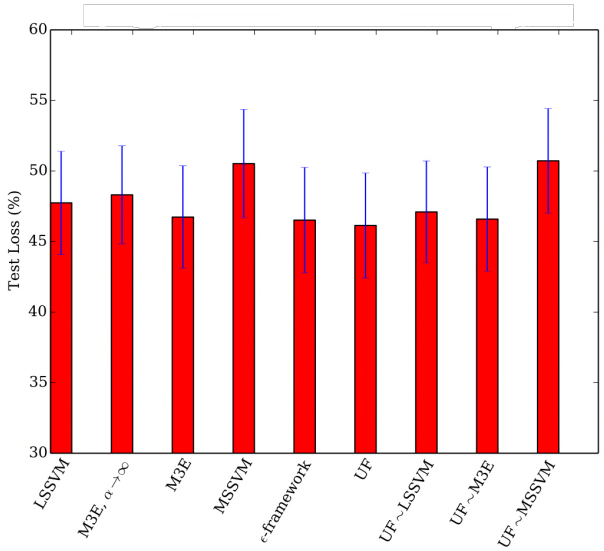


Figure 3.3: Test loss mean on the 10 classes (in %) ± standard error of the mean (in %) with cross-validated parameters on the PASCAL VOC 11 data set, after averaging on the 5 folds.

best α chosen for the M3E models is of high value. In other words, M3E recovers LSSVM. The ϵ-framework chooses a small value of ϵ, which also approximates LSSVM. These results show that when the size of the latent space is small (we use 20 bounding boxes) and that uncertainty in the latent variable is reduced (we consider top scoring bounding boxes), there is a small gain to be made from taking into account the uncertainty in the latent variable. This comes at the cost of using a more complicated model with more parameters to cross-validate.

3.5.2 Multi-class gesture recognition

We now explore an experiment where the uncertainty in the value of the latent variable is high and the latent space is large. To do so, we tested the different models on the task of gesture recognition in video sequences. Given a set of video sequences, our goal is to learn to classify, among c classes of gestures, the gesture performed in a video sequence.

MSRC-12. The MSRC-12 data set (Fothergill et al., 2012) contains 594 sequences of motion capture data, recording the 3D world position of 20 joints in the human body using a Kinect sensor from a population of 30 individuals. In each sequence the actor performs one type of gesture repeatedly, typically 10 times, and each instance of the gesture is marked at a typical frame. The original purpose of the data set was to enable research into low latency gesture detection (Nowozin and Shotton, 2012). However, it has since been used to classify the sequence as a whole (Fothergill et al., 2012; Lehrmann et al., 2014). We also perform sequence classification but do not use the individual frame-level annotations. Instead, as training data we use only the sequence class label. This is a realistic assumption in an interactive setting when the user is instructed by a system to perform a gesture. We know that a gesture of a given class will be performed in the video but not when the user will effectively perform it.

Noisy MSRC-12 data set. In order to evaluate the behavior of the different models in presence of noisy data, we corrupt the MSRC-12 data set by adding random Gaussian noise with zero mean and various standard deviations. In particular we add noise to each element of all frames' feature vectors. We use three values of standard deviations: $\sigma = [1\text{cm}, 5\text{cm}, 8\text{cm}]$.

Modeling and features. The input variable $\boldsymbol{x}$ consists of 3D world position of a person performing a gesture. The output variable $\boldsymbol{y}$ is the class of the gesture performed in the video sequence. In a video sequence only a few percentage of the video frames effectively contains the person performing the gesture. We consider the training data as weakly labelled, that is, the video sequences are only labelled at the sequence level and each frame is not individually annotated as containing the gesture or not. We use the latent variable $\boldsymbol{z}$ to model this lack of information. The size of the latent space is the number of frames per video sequence, that is, on aver-

age 1200 frames per sequence. The score of a specific frame z in a video sequence x is $\theta^T\phi(x, y, z) = \theta_y^T\phi(x, z)$ where θ_y are the parameters that correspond to the label y and $\phi(x, z)$ is the feature vector extracted from the frame z in the sequence x. We take the same features derived from 3D joint locations as in Fothergill et al. (2012); Lehrmann et al. (2014); Nowozin and Shotton (2012), obtaining a feature vector $\phi(x, z)$ of dimension 130. The loss Δ is the standard 0/1 loss.

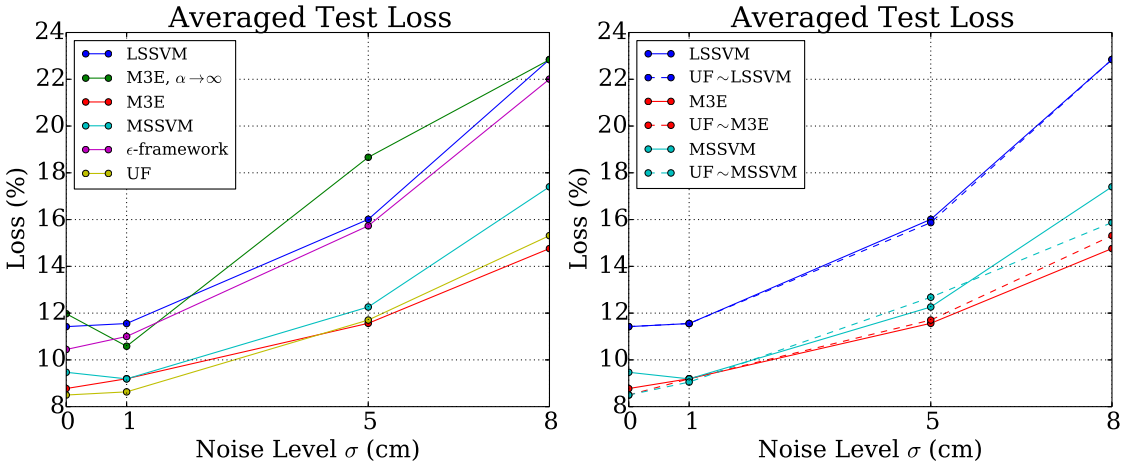
Experimental setting. We use the 594 sequences from the MSRC-12 data set. We divide this data set into five folds, each fold containing 20% of the trainval data set for testing and 80% for training, using stratified sampling over class labels in order to ensure a uniform distribution of classes. As in the action classification experiment, hyper parameters are cross-validated over 5 folds and we use 4 random seeds for initialisation and for each fold, we report the test error corresponding to the seed with the lowest training objective value. Additional details regarding the experimental setup and the results can be found in appendix 7.4.7.

Results. Table 3.1 reports the best parameters chosen by cross-validation for the UF. It shows that the best parameters combination $(\epsilon_z, \alpha, \beta)$ always take in account the AD entropy of the latent variable and the term $H_{\alpha,\beta}(p_x^y; \theta)$ in equation (3.4) is never set to 0. This would be the case if we had $(\alpha = 2, \beta = 0)$ or $(\alpha \to \infty, \beta = 1)$. Therefore, the UF is never boiling down to either LSSVM, MSSVM or the ϵ-framework. Moreover, except for $\sigma = 1$cm, the UF recovers M3E models, that is $\epsilon_z = 1$ and $\beta = 1$. Recall that the UF recovers all prediction criteria using the same optimisation algorithm. This means that regardless of the algorithm used to solve the minimisation problem during learning, predicting the output variables by minimising the uncertainty in the latent variable is a more relevant method than marginalising the latent variable or estimating its most likely value.

Figure 3.4a shows the average loss on the testing set for each model with respect

$\sigma = 0$cm	$\sigma = 1$cm	$\sigma = 5$cm	$\sigma = 8$cm
$\epsilon_z = 1$, $\alpha = 0.01$, $\beta = 1$	$\epsilon_z = 1$, $\alpha = 2$, $\beta = 0.5$	$\epsilon_z = 1$, $\alpha = 0.1$, $\beta = 1$	$\epsilon_z = 1$, $\alpha = 0.1$, $\beta = 1$

Table 3.1: Cross-validated parameters for the UF.



(a) Existing models and the UF.

(b) Existing models (line curves) and their replication by the UF (dashed curves).

Figure 3.4: Test loss (%) on the MSRC-12 data set averaged on 5 folds, per noise level. Each model is shown with best cross-validated parameters.

to the noise level corrupting the data set, and Figure 3.4b shows the average loss on the testing set for LSSVM, M3E and MSSVM, and their replication with the UF. This figure shows two things. First, by looking at the performance of their replication by the UF (dotted lines), we can compare LSSVM, MSSVM and M3E models without taking into account their specific training algorithm. Second, this figure also shows how the UF replicates existing models. When the UF replicates LSSVM and MSSVM results are similar, this was expected since the algorithm we derived for training the UF is similar to the specific algorithm of these models. When the UF replicates M3E, results are also similar even if the optimisation procedures are different (the M3E models use a trust-region based algorithm to solve the optimisation problem). From Figures 3.4a and 3.4b, we conclude that entropy-

based models (either M3E or the UF) give significantly better performance at all noise levels. We refer the reader to appendix 7.4.7 for p-values and comparison at statistical significance level of 0.05.

From this second experiment, we conclude that (i) when the size of the latent space is large (recall that we have on average 1200 frames per sequence), and (ii) there is uncertainty in the latent variable (especially as we add noise), then performing output prediction by minimising the latent space uncertainty is the most accurate criterion.

While we do not quantitatively evaluate the ability of the model to accurately predict the latent variable, we illustrate the behavior of the UF over the latent space. Figure 3.5 shows the scoring of the UF on an example "bowing" gesture of

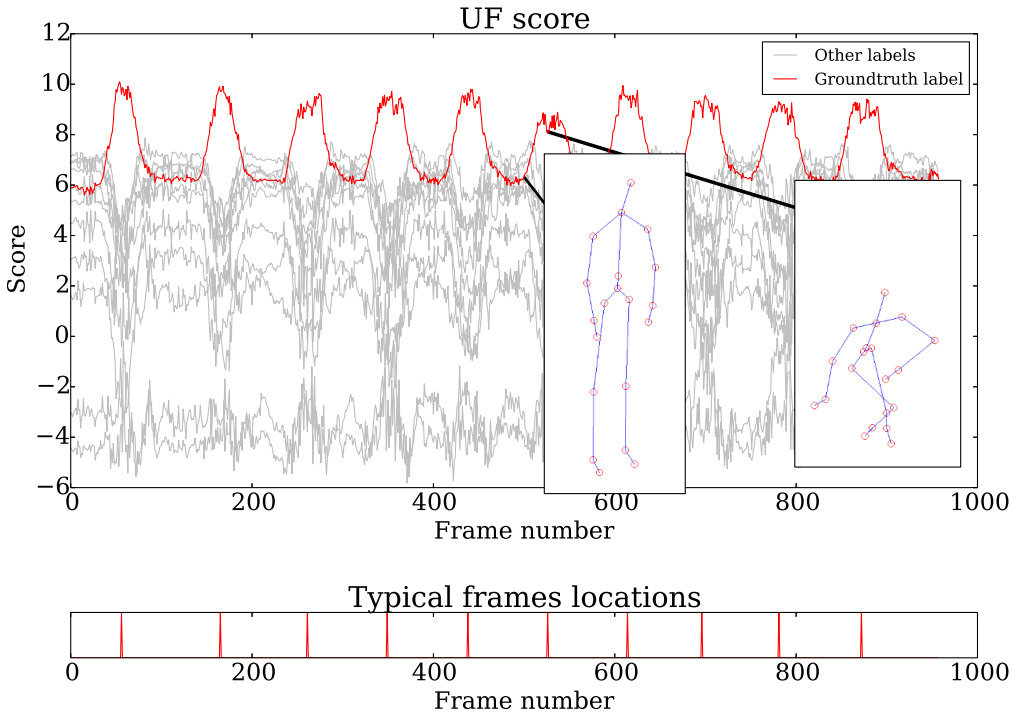


Figure 3.5: Scoring of the UF on an example "bowing" gesture of the non noisy MSRC-12 data set. Upper part of the plot shows the scoring for the ground-truth label (in red), and all other labels in grey. Lower part of the plot shows the location of the action frames as per MSRC-12 ground-truth (we do not use this frame-level annotation).

the non-noisy MSRC-12 data set. A 3D representation of the frame feature vector is added in the upper part of the plot. For the ground-truth label, the UF gives high score at typical frames of the video sequence when the person is effectively bowing, and a low score in between. In other words, our model predicts the correct gesture in the video by identifying the frames that give most information about the gesture performed. The scores of all others labels (shown in grey) are smaller than the smallest score of the ground-truth label frames.

3.6 Discussion

We developed a Unified Framework (UF) for the LSSVM model and its generalisations by defining a simple prediction criterion, and designed an optimisation algorithm for learning the parameters of the UF. This allowed us to perform a fair comparison between the prediction criteria of the aforementioned methods by discounting the differences in performance that could arise from using their specific optimisation algorithm. Our experimental results show that the use of the minimisation of the latent space uncertainty is an accurate prediction criterion when the size of the latent space is large and when there is uncertainty in the latent variable. This result aligns with our intuition. As noted in Miller et al. (2012), the assumption is that the latent variable is informative for predicting the output. Thereby, minimising the uncertainty on the latent variable value ensures that the output predicted (i) has a high probability and (ii) present little confusion in the value of the latent variable.

The UF also offers an additional advantage, namely, the use of the extra parameter β. This has yet to be explored thoroughly, and could lead to improved performance. Another interesting direction for future research would be to determine the advantage of minimising uncertainty in the latent variable in applications with large

and complex output spaces. The formulation of the LSSVM enables optimisation with energy minimisation techniques such as graph cuts (Boykov et al., 2001). In our experiments we choose application problems where all output configurations are estimated in a computationally feasible manner, in order to remove from our comparison the possible effects of the use of approximation methods. For this reason, we focused on discrete and low-dimensional output spaces, with categorical distributions on the latent and output variables. Moreover, we perform experiments using classic computer vision features. Since our publication (Bouchacourt et al., 2015), weakly supervised structured output prediction has benefitted from the development of deep neural networks in order to extract features, see for example Durand et al. (2016); Oquab et al. (2015); Pathak et al. (2015); Tokmakov et al. (2016).

Chapter 4

DISCO Nets: DISsimilarity COefficient Networks

The latent SVM framework, and its generalisations, rely on two implicit assumptions. First, while there may be high uncertainty in the latent variable, it is sufficient to provide only a single pointwise estimate of the output variable. Second, the joint feature vector representation has sufficient discriminative ability to obtain an accurate prediction of the output. In this chapter, we consider a significantly more challenging setting for learning where (i) the output variable has high uncertainty, thereby making pointwise models inappropriate; (ii) the output space is highly complex and structured, thereby requiring data-driven representation learning instead of hand-crafted features. Despite the challenges of this setting, we would still like our learning objective to be calibrated to a user-defined loss. To this end, we introduce a novel type of deep probabilistic model, which we call DISsimilarity COefficient Networks (DISCO Nets).

DISCO Nets allow us to efficiently sample from a posterior distribution parametrised by a neural network. During training, DISCO Nets are learned by

minimising a dissimilarity coefficient between the true distribution of the output and the distribution of the model. This coefficient is defined according to a loss of choice. DISCO Nets do not specify the type of the model's distribution, but enforce structure by using non-decomposable loss functions which do not decompose over the dimensions of the output. The main contributions of our method are (i) We design a new type of learning objective to perform probabilistic structured output prediction, which the user can tailor to the loss associated with the task at hand; (ii) Using the theory of scoring rules, we show that our learning objective is minimised if and only if the model recovers the true distribution; (iii) In the specific case of a discrete output, our learning procedure involves a non-differentiable step. We derive an approximation theorem and propose a corresponding optimisation algorithm; (iv) We empirically show the relevance of the model to capture distributions over complex, structured outputs.

4.1 Introduction

We are interested in the class of problems that require the prediction of a structured output $y \in \mathcal{Y}$ given an input $x \in \mathcal{X}$. Complex applications often have large uncertainty in the output y. For example, consider the task of hand pose estimation from depth images, where one wants to accurately predict the pose y of a hand in a depth image x. In this case, we illustrate in our introductory section 1.1.2 how occlusions and missing depth values in the image result in uncertainty in the pose to predict. It is, therefore, natural to use probabilistic models that are capable of representing this uncertainty. Moreover, some applications require multiple, yet plausible outputs that correspond to a single input. Consider for example that we want to perform hand tracking in video sequences. Probabilistic models provide us with multiple probable poses of the hand in a frame at time t. We can use these

candidate poses to track the hand in the frame at time $t + 1$.

In addition to the uncertainty in the output to predict, we must take into account model misspecification, that is, the fact that the model is of restricted capacity and cannot capture the true data distribution perfectly. We illustrate in section 1.1.3 how the choice of learning objective influences final performance. We argue that the learning objective should be tailored to the evaluation loss in order to obtain the best performance with respect to this loss. This is in contrast to the commonly employed learning objectives of probabilistic models, which are agnostic of the evaluation loss.

In order to tackle the aforementioned deficiency, we introduce DISCO Nets, a new class of probabilistic models for structured output prediction. DISCO Nets represent P, the true posterior distribution of the data, with a distribution Q parametrised by a neural network. We design a learning objective based on a dissimilarity coefficient between P and Q. The dissimilarity coefficient employed was first introduced by Rao (1982) and is defined for any non-negative symmetric loss function. Thus, our setting allows the user to tailor DISCO Nets by employing his or her loss of choice. Finally, in constrast to existing probabilistic models, DISCO Nets do not require any specific architecture, making them an efficient and easy-to-use class of models.

4.2 Deep probabilistic models for structured output prediction

In probabilistic structured output prediction, the distribution of interest is a conditional distribution of an output given an input. In this context, the Generative Adversarial Networks (GAN) model, discussed in section 2.3.2, has been adapted to conditional distributions with conditional GAN (cGAN) (Gauthier, 2015; Mirza and Osindero, 2014; Reed et al., 2016). However, GAN-based models, and therefore

cGAN, employ adversarial training that tends to oscillate (Goodfellow et al., 2014; Radford et al., 2015). Since our publication (Bouchacourt et al., 2016), the training of GAN-based models and their conditional variants has received a lot of attention in the literature (Lopez-Paz et al., 2016). As a result, researchers have successfully developed the range of applications of cGAN, see for example Denton et al. (2016); Isola et al. (2017). However, the training of GAN-based models is still broadly acknowledged to be a challenging optimisation task (Goodfellow, 2016). The Variational Autoencoder (VAE), presented in section 2.3.3, has also been generalised to the modelling of conditional distributions (Sohn et al., 2015; Yan et al., 2016). Similar to the original VAE, in contrast to implicit models, the conditional VAE model explicitly specificies the types of the encoding and decoding distributions. Concurrently to our publication (Bouchacourt et al., 2016), Ren et al. (2016) extend deep probabilistic models based on Maximum Mean Discrepancy (MMD) (Dziugaite et al., 2015; Li et al., 2015) to the discriminative case using kernel mean embedding of conditional distributions. We mention in section 2.1.3 that MMD can be related to the Kernel score. However, our work differs from theirs as we employ scoring rules to compare conditional distributions for a given input, and our formulation encourages the use of a scoring rule based on the task loss.

In hand pose estimation, imagine the user wants to obtain accurate positions of the thumb and the index finger but does not need accurate locations of the other fingers. The task loss might be a weighted L2-norm of the difference between two poses, with high weights on the thumb and the index. Existing probabilistic models cannot be tailored to task-specific losses and we propose the DISsimilarity COefficient Networks (DISCO Nets) to alleviate this deficiency.

4.3 The DISCO Nets model

The DISCO Nets model is an implicit deep probabilistic model and provides the user with samples from the distribution captured by the model.

4.3.1 Architecture

DISCO Nets use a neural network to parametrise the model's distribution Q. The network takes as input a pair $(x, z) \in \mathcal{X} \times \mathcal{Z}$, where x is an input data sampled from P_x and z is a random noise vector sampled from a known distribution P_z (for example a uniform distribution). Given a pair (x, z), DISCO Nets produce a value for the output variable y. Figure 4.1 illustrates this process in the example of hand pose estimation. The input depth image x is fed to the convolutional layers. The output of the last convolutional layer is flattened and concatenated with the noise sample z. The resulting vector is fed to several dense layers, and the last dense layer outputs a pose y. From a single depth image x, by using different noise samples, DISCO Nets produce different candidate poses for the same depth image.

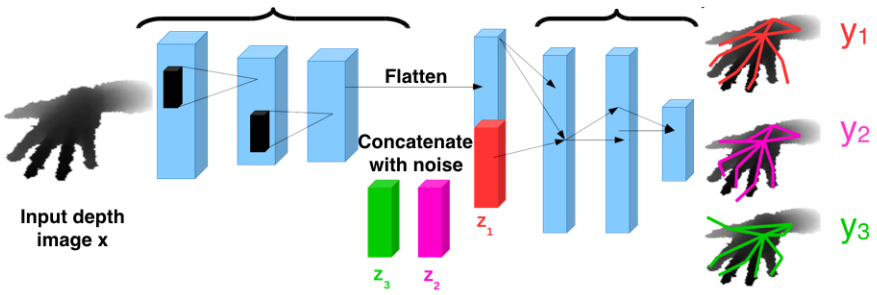


Figure 4.1: For a single depth image x, using 3 different noise samples (z_1, z_2, z_3), DISCO Nets output 3 different candidate poses (y_1, y_2, y_3) (shown superimposed on the depth image). The depth image is from the NYU Hand Pose data set (Tompson et al., 2014), preprocessed as in Oberweger et al. (2015b).

Importantly, DISCO Nets are flexible in the choice of the architecture. For example, the noise could be concatenated at any stage of the network, including

at the start. We denote by $Q(\boldsymbol{y}|x;\theta)$ the distribution captured by DISCO Nets, where θ are the parameters of the neural network. For a given input x, DISCO Nets provide the user with samples y drawn from $Q(\boldsymbol{y}|x;\theta)$ without requiring the expensive computation of the (often intractable) partition function.

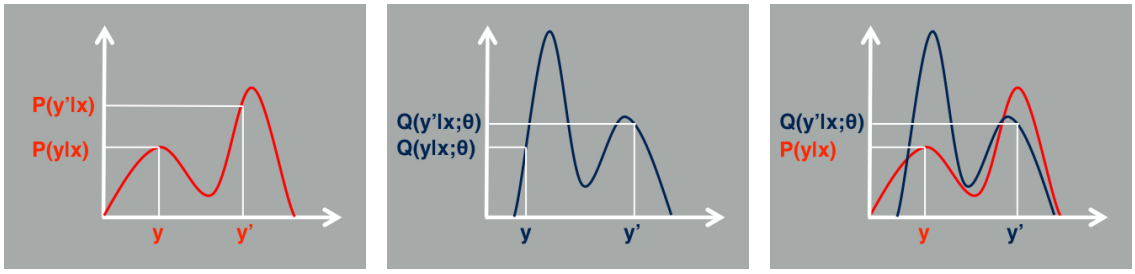
4.3.2 Prediction of a single output

Some applications, for example hand tracking in video sequences, benefit from using multiple hand poses for the same image as we mentioned in section 4.1. Other tasks require only a single output. Suppose that the user needs, given a 2D image, a single hand pose to generate a 3D reconstruction of the hand. Non-probabilistic models would provide the user with a single pose prediction to perform 3D rendering. However, they would not provide multiple outputs. By contrast, DISCO Nets produce multiple possible outputs for a given input. If the task at hand require a single output, how to elect, among samples drawn from DISCO Nets, the best sample to perform a task?

In order to obtain a single prediction y for a given input x, DISCO Nets use the principle of Maximum Expected Utility (MEU)(Premachandran et al., 2014). The prediction y_{MEU} maximises the expected utility, or rather minimises the expected task-specific loss, Δ. This expected loss is estimated using only samples from DISCO Nets, in other words, drawn from $Q(\boldsymbol{y}|x;\theta)$. Formally, the prediction is made as follows,

$$y_{\text{MEU}} = \underset{k=1,\ldots,K}{\arg\max}\, \text{EU}(y_k) = \underset{k=1,\ldots,K}{\arg\min} \sum_{k'=1}^{K} \Delta(y_k, y_{k'}),$$

where $(y_1, \ldots, y_K)$ are candidate outputs drawn from DISCO Nets for the input x.



(a) $\mathrm{DIV}_\Delta(P,P)^x$*: draw randomly* y, y' *from* P*, and compute the loss* $\Delta(y, y')$*.*

(b) $\mathrm{DIV}_\Delta(Q,Q)^x$*: draw randomly* y, y' *from* Q*, and compute the loss* $\Delta(y, y')$*.*

(c) $\mathrm{DIV}_\Delta(P,Q)^x$*: draw randomly* y *from* P*,* y' *from* Q*, and compute the loss* $\Delta(y, y')$*.*

Figure 4.2: Diversity coefficient of P*, of* Q*, and between* P *and* Q*.*

4.3.3 Loss-calibrated probabilistic objective

For a given input x, we want DISCO Nets to accurately model the true distribution $P(\boldsymbol{y}|x)$ via $Q(\boldsymbol{y}|x;\theta)$. In other words, $Q(\boldsymbol{y}|x;\theta)$ should be as similar as possible to $P(\boldsymbol{y}|x)$. However, we want to take into account the task at hand. We assume that with this task is associated a specific evaluation loss, $\Delta(y, y')$ with $(y, y') \in \mathcal{Y} \times \mathcal{Y}$, which measures the discrepancy between two outputs. We evaluate the similarity between $P(\boldsymbol{y}|x)$ and $Q(\boldsymbol{y}|x;\theta)$ with respect to the evaluation loss. To do so, we employ a diversity coefficient that is defined for the loss Δ. Consider a given input x, and the distribution $P(\boldsymbol{y}|x)$ over the output variable. The diversity of P according to the loss function Δ is the expected loss between two samples drawn randomly from P, that is,

$$\mathrm{DIV}_\Delta(P,P)^x = \mathbb{E}_{y\sim P(\boldsymbol{y}|x)}\left[\mathbb{E}_{y'\sim P(\boldsymbol{y}|x)}[\Delta(y, y')]\right].$$

We denote $\mathrm{DIV}_\Delta(P,P)^x$ to emphasise the fact that this coefficient is computed for a given input value x. The diversity of P is shown in Figure 4.2a. Similarly, the diversity of the DISCO Nets' distribution $Q(\boldsymbol{y}|x;\theta)$ is the expected loss between

two samples drawn randomly from $Q(\boldsymbol{y}|x;\theta)$, as follows,

$$\mathrm{DIV}_{\Delta}(Q,Q)^x = \mathbb{E}_{y\sim Q(\boldsymbol{y}|x;\theta)}\left[\mathbb{E}_{y'\sim Q(\boldsymbol{y}|x;\theta)}[\Delta(y,y')]\right].$$

Finally, as shown in Figure 4.2c, we can compute the diversity between the two distributions $P(\boldsymbol{y}|x)$ and $Q(\boldsymbol{y}|x;\theta)$, as follows,

$$\mathrm{DIV}_{\Delta}(P,Q)^x = \mathbb{E}_{y\sim P(\boldsymbol{y}|x)}\left[\mathbb{E}_{y'\sim Q(\boldsymbol{y}|x;\theta)}[\Delta(y,y')]\right].$$

The term $\mathrm{DIV}_{\Delta}(P,Q)^x$ is the expected loss between two samples drawn randomly from the two distributions respectively. Intuitively, one wants to minimise $\mathrm{DIV}_{\Delta}(P,Q)^x$ so that $Q(\boldsymbol{y}|x;\theta)$ is as similar as possible to $P(\boldsymbol{y}|x)$. However, for a given x, $\mathrm{DIV}_{\Delta}(P,P)^x \neq 0$, in other words, $P(\boldsymbol{y}|x)$ is diverse. Therefore, $Q(\boldsymbol{y}|x;\theta)$ should be diverse as well, or equivalently, $\mathrm{DIV}_{\Delta}(Q,Q)^x \neq 0$. We encourage $Q(\boldsymbol{y}|x;\theta)$ to provide diverse outputs, for a given x, by minimising the following dissimilarity coefficient:

$$\mathrm{DIV}_{\Delta}(P,Q)^x - \gamma\mathrm{DIV}_{\Delta}(Q,Q)^x - (1-\gamma)\mathrm{DIV}_{\Delta}(P,P)^x,$$

with $\gamma \in [0,1]$. The dissimilarity coefficient is the difference between the diversity of P and Q, and an affine combination of the diversity of each distribution. The scalar parameter γ balances between the two goals, that are, producing model samples that are close to the data samples, and enforcing diversity in the model's distribution. The diversity and dissimilarity coefficients were introduced by Rao (1982) with $\gamma = 1/2$ for any non-negative symmetric loss function, and used in latent variable models (Kumar et al., 2012). To build our probabilistic objective,

we integrate the dissimilarity coefficient over all possible input values x, which gives,

$$\begin{aligned}\mathrm{DISCO}_\Delta(P, Q, P_x) = \; & \mathbb{E}_{x\sim P_x(\boldsymbol{x})}[\mathrm{DIV}_\Delta(P, Q)^x \\ & - \gamma \mathrm{DIV}_\Delta(Q, Q)^x - (1-\gamma)\mathrm{DIV}_\Delta(P, P)^x].\end{aligned}$$

The resulting expression $\mathrm{DISCO}_\Delta(P, Q, P_x)$ is the DISCO Nets' probabilistic objective. We minimise this objective to learn the model's parameters θ. As we see, it is tailored to the evaluation loss Δ associated to the task at hand. When $\gamma = 1/2$ and for certain types of loss, using strictly proper scoring rule theory, we show in section 4.4 that we have the following guarantee:

$$\mathrm{DISCO}_\Delta(P, Q, P_x) = 0 \iff Q(\boldsymbol{y}|x;\theta) = P(\boldsymbol{y}|x) \text{ almost everywhere on } \mathcal{X}.$$

Let us define the following terms,

$$\begin{aligned}\mathrm{DIV}_\Delta(P, Q, P_x) &= \mathbb{E}_{x\sim P_x(\boldsymbol{x})}[\mathrm{DIV}_\Delta(P, Q)^x], \\ \mathrm{DIV}_\Delta(P, P, P_x) &= \mathbb{E}_{x\sim P_x(\boldsymbol{x})}[\mathrm{DIV}_\Delta(P, P)^x], \\ \mathrm{DIV}_\Delta(Q, Q, P_x) &= \mathbb{E}_{x\sim P_x(\boldsymbol{x})}[\mathrm{DIV}_\Delta(Q, Q)^x],\end{aligned}$$

and $\mathrm{DISCO}_\Delta(P, Q, P_x)$ is expressed as follows[1],

$$\mathrm{DISCO}_\Delta(P, Q, P_x) = \mathrm{DIV}_\Delta(P, Q, P_x) - \gamma \mathrm{DIV}_\Delta(Q, Q, P_x) - (1-\gamma)\mathrm{DIV}_\Delta(P, P, P_x).$$

During training, we do not need to consider the term $\mathrm{DIV}_\Delta(P, P, P_x)$ as it is a constant that only depends on the data, and we equivalently minimise:

$$\mathrm{F}_\Delta(P, Q, P_x) = \mathrm{DIV}_\Delta(P, Q, P_x) - \gamma \mathrm{DIV}_\Delta(Q, Q, P_x). \tag{4.1}$$

[1] We assume that all expectations are finite.

4.3.4 Optimisation procedure for continuous outputs

In this section, we consider the random variables $\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}$ to be continuous. Specifically, $x \in \mathcal{X} = \mathbb{R}^{d_x}, y \in \mathcal{Y} = \mathbb{R}^{d_y}, z \in \mathcal{Z} = \mathbb{R}^{d_z}$. The specific case of a discrete output $\boldsymbol{y}$ raises new challenges that we address in section 4.6.

In order to train DISCO Nets, we need to compute the training objective of equation (4.1). We do not have full knowledge of the true probability distributions $P(\boldsymbol{y}|x)$ and $P(\boldsymbol{x})$. In the context of statistical estimation, we consider a training data set composed of n example input-output pairs $\mathcal{D} = \{(x_i, y_i), i = 1, \ldots, n\}$ and estimate the diversity terms $\mathrm{DIV}_\Delta(P, Q, P_x)$ and $\mathrm{DIV}_\Delta(Q, Q, P_x)$. First, we take an empirical distribution of the data, that is, taking ground-truth pairs (x_i, y_i). Second, we estimate each distribution $Q(\boldsymbol{y}|x_i; \theta)$ by sampling K outputs from our model, given the input x_i. This gives us values of unbiased estimators of each diversity term, defined as follows,

$$\widehat{\mathrm{DIV}}_\Delta(P, Q, P_x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{K}\sum_{k=1}^{K}\Delta(y_i, G(x_i, z_{i,k}; \theta)),$$

$$\widehat{\mathrm{DIV}}_\Delta(Q, Q, P_x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{K(K-1)}\sum_{k=1}^{K}\sum_{\substack{k'=1,\\ k'\neq k}}^{K}\Delta(G(x_i, z_{i,k}; \theta), G(x_i, z_{i,k'}; \theta)),$$

where $y_{i,k} = G(x_i, z_{i,k}; \theta)$ is a candidate output sampled from DISCO Nets, given a pair $(x_i, z_{i,k})$. It is important to note that the second term of $\widehat{\mathrm{DIV}}_\Delta(Q, Q, P_x)$ is summing over k and $k' \neq k$ to have an unbiased estimator (Serfling, 1980), therefore we compute the loss between pairs $G(x_i, z_{i,k}; \theta)$ and $G(x_i, z_{i,k'}; \theta)$ where $k' \neq k$. As a result, we have an unbiased estimator of our training objective $\mathrm{F}_\Delta(P, Q, P_x)$, that is,

$$\widehat{\mathrm{F}}_\Delta(P, Q, P_x) = \widehat{\mathrm{DIV}}_\Delta(P, Q, P_x) - \gamma\widehat{\mathrm{DIV}}_\Delta(Q, Q, P_x). \tag{4.2}$$

We learn the parameters θ with mini-batch gradient descent. Specifically, we use

the Law of the Unconscious Statistician (LOTUS), discussed in section 2.3.1, to obtain an estimate of the gradient of our objective function. Algorithm 2 describes the training of DISCO Nets. In step 2 of Algorithm 2, we draw K random noise vectors $(z_{i,1}, \ldots, z_{i,k})$ per input example x_i, and generate K candidate outputs $G(x_i, z_{i,k}; \theta)$ corresponding to the input x_i in step 3. This allows us to build unbiased estimator of the gradient and compute a gradient estimate in step 5. In practice the number of epochs T is set to high value and we monitor the training objective on a subset of the training data set to check for convergence.

Algorithm 2: DISCO Nets training algorithm for continuous outputs.

for *t=1,...,T epochs* **do**

1 Sample mini-batch of b training example pairs $\{(x_1, y_1), \ldots, (x_b, y_b)\}$.

for *i=1,...,b* **do**

2 Sample K random noise vectors $(z_{i,1}, \ldots, z_{i,k})$ for input x_i.

3 Generate K candidate outputs $G(x_i, z_{i,k}; \theta^t), k = 1, \ldots, K$ for input x_i.

end

4 Compute the gradient estimates,

$$\widehat{\nabla_\theta}\mathrm{DIV}_\Delta(P, Q, P_x)|_{\theta^t} = \frac{1}{b}\sum_{i=1}^{b}\frac{1}{K}\sum_{k=1}^{K}\nabla_\theta\Delta(y_i, G(x_i, z_{i,k}; \theta))|_{\theta^t},$$

$$\widehat{\nabla_\theta}\mathrm{DIV}_\Delta(Q, Q, P_x)|_{\theta^t} = \frac{1}{b}\sum_{i=1}^{b}\frac{1}{K(K-1)}\sum_{k=1}^{K}\sum_{\substack{k'=1,\\ k'\neq k}}^{K}\nabla_\theta\Delta(G(x_i, z_{i,k}; \theta), G(x_i, z_{i,k'}; \theta))|_{\theta^t}.$$

5 Update parameters $\theta^{t+1} \leftarrow \theta^t$ by descending the gradient estimate evaluated at θ^t: $\widehat{\nabla_\theta}\mathrm{F}_\Delta(P, Q, P_x)|_{\theta^t}$.

end

4.4 Relation to the theory of scoring rules

In section 2.1, we introduce scoring rules to measure "how good" is a forecasting distribution Q to predict a process represented by a true distribution P. For-

mally, a scoring rule is a function $S : \mathcal{P} \times \mathcal{Y} \rightarrow \mathbb{R} \cup \{-\infty, \infty\}$ such that for $Q \in \mathcal{P}$, $S(Q, y)$ evaluates the quality of the predictive distribution Q to forecast the realisation $y \sim P$. In this chapter, we use scoring rules to evaluate conditional distributions of the output variable $\boldsymbol{y}$, for a given input x. Therefore, we use the notation $S(Q, y)^x$. The expected score associated to $S(Q, y)^x$ is

$$S(Q, P)^x = \int_{\mathcal{Y}} S(Q, y)^x \mathrm{d}P(y|x).$$

The score divergence corresponding to the scoring rule S is

$$d(Q, P)^x = S(P, P)^x - S(Q, P)^x. \tag{4.3}$$

If S is proper, d is a non-negative function, with value 0 if (and only if, in the case of strictly proper) $Q = P$. We relate the DISCO Nets' objective function to scoring rule theory. Specifically, we show that the DISCO Nets' objective can be written as the expectation with respect to the input variable $\boldsymbol{x}$ of the divergence $d(Q, P)^x$. First, let us note that when $\gamma = 1/2$, the DISCO Nets' objective is

$$\mathrm{DISCO}_{\Delta}(P, Q, P_x) = \mathbb{E}_{x \sim P_x(\boldsymbol{x})}[\mathrm{DIV}_{\Delta}(P, Q)^x - \frac{1}{2}\mathrm{DIV}_{\Delta}(Q, Q)^x - \frac{1}{2}\mathrm{DIV}_{\Delta}(P, P)^x]. \tag{4.4}$$

4.4.1 DISCO Nets as the Energy score

Consider the loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}, \Delta(y, y') = \|y - y'\|_2^{\beta}$, where $\|.\|_2$ denotes the Euclidean norm and $\beta \in (0, 2)$ is a scalar parameter. This loss $\Delta(y, y')$ measures the distance between y and y'. Replacing the expression of Δ in the diversity terms

of equation (4.4), we get the following terms for an input x:

$$\mathrm{DIV}_\Delta(P,Q)^x = \mathbb{E}_{y\sim P(\boldsymbol{y}|x)}\left[\mathbb{E}_{y'\sim Q(\boldsymbol{y}|x;\theta)}\left[\|y-y'\|_2^\beta\right]\right], \tag{4.5}$$

$$\mathrm{DIV}_\Delta(Q,Q)^x = \mathbb{E}_{y'\sim Q(\boldsymbol{y}|x;\theta)}\left[\mathbb{E}_{y''\sim Q(\boldsymbol{y}|x;\theta)}\left[\|y'-y''\|_2^\beta\right]\right], \tag{4.6}$$

$$\mathrm{DIV}_\Delta(P,P)^x = \mathbb{E}_{y'\sim P(\boldsymbol{y}|x)}\left[\mathbb{E}_{y''\sim P(\boldsymbol{y}|x)}\left[\|y'-y''\|_2^\beta\right]\right]. \tag{4.7}$$

If we plug-in equations (4.5, 4.6, 4.7) into equation (4.4), we get the following,

$$\begin{aligned}\mathrm{DISCO}_\Delta(P,Q,P_x) = \; & \mathbb{E}_{x\sim P_x(\boldsymbol{x})}\Big[\mathbb{E}_{y\sim P(\boldsymbol{y}|x)}\left[\mathbb{E}_{y'\sim Q(\boldsymbol{y}|x;\theta)}\left[\|y-y'\|_2^\beta\right]\right] \\ & - \frac{1}{2}\mathbb{E}_{y'\sim Q(\boldsymbol{y}|x;\theta)}\left[\mathbb{E}_{y''\sim Q(\boldsymbol{y}|x;\theta)}\left[\|y'-y''\|_2^\beta\right]\right] \\ & - \frac{1}{2}\mathbb{E}_{y'\sim P(\boldsymbol{y}|x)}\left[\mathbb{E}_{y''\sim P(\boldsymbol{y}|x)}\left[\|y'-y''\|_2^\beta\right]\right]\Big].\end{aligned} \tag{4.8}$$

We now show the connection with the Energy score. The Energy score of a distribution $T(\boldsymbol{y}|x)$, evaluated for a realisation $y \sim P(\boldsymbol{y}|x)$, is defined as follows (see Definition 2.4),

$$S(T,y)^x = \frac{1}{2}\mathbb{E}_{y',y''\sim T(\boldsymbol{y}|x)}\left[\|y'-y''\|_2^\beta\right] - \mathbb{E}_{y'\sim T(\boldsymbol{y}|x)}\left[\|y'-y\|_2^\beta\right].$$

We evaluate $P(\boldsymbol{y}|x)$ and $Q(\boldsymbol{y}|x;\theta)$ under this scoring rule, by computing their expected scores, that are,

$$\begin{aligned}S(Q,P)^x = \mathbb{E}_{y\sim P(\boldsymbol{y}|x)}[S(Q,y)^x] = & \frac{1}{2}\mathbb{E}_{y'\sim Q(\boldsymbol{y}|x;\theta)}\left[\mathbb{E}_{y''\sim Q(\boldsymbol{y}|x;\theta)}\left[\|y'-y''\|_2^\beta\right]\right] \\ & - \mathbb{E}_{y\sim P(\boldsymbol{y}|x)}\left[\mathbb{E}_{y'\sim Q(\boldsymbol{y}|x;\theta)}\left[\|y'-y\|_2^\beta\right]\right],\end{aligned} \tag{4.9}$$

$$S(P,P)^x = \mathbb{E}_{y\sim P(\boldsymbol{y}|x)}[S(P,y)^x] = -\frac{1}{2}\mathbb{E}_{y'\sim P(\boldsymbol{y}|x)}\left[\mathbb{E}_{y''\sim P(\boldsymbol{y}|x)}\left[\|y'-y''\|_2^\beta\right]\right]. \tag{4.10}$$

If we plug in equations (4.9, 4.10) into equation (4.3), we get:

$$\begin{aligned} d(Q,P)^x &= S(P,P)^x - S(Q,P)^x \\ &= -\frac{1}{2}\mathbb{E}_{y'\sim P(\boldsymbol{y}|x)}\left[\mathbb{E}_{y''\sim P(\boldsymbol{y}|x)}\left[\|y'-y''\|_2^\beta\right]\right] \\ &\quad - \frac{1}{2}\mathbb{E}_{y'\sim Q(\boldsymbol{y}|x;\theta)}\left[\mathbb{E}_{y''\sim Q(\boldsymbol{y}|x;\theta)}\left[\|y'-y''\|_2^\beta\right]\right] \\ &\quad + \mathbb{E}_{y\sim P(\boldsymbol{y}|x)}\left[\mathbb{E}_{y'\sim Q(\boldsymbol{y}|x;\theta)}\left[\|y'-y\|_2^\beta\right]\right]. \end{aligned} \tag{4.11}$$

Comparing equation (4.8) and equation (4.11), we see that when $\gamma = 1/2$ we can express the DISCO Nets' objective as the expectation of $d(Q,P)^x$, the divergence associated to the Energy score for a given x, as follows,

$$\mathrm{DISCO}_\Delta(P,Q,P_x) = \mathbb{E}_{x\sim P_x(\boldsymbol{x})}[d(Q,P)^x]\,.$$

The Energy score is a strictly proper scoring rule for $\beta \in (0,2)$[2]. Therefore, the associated divergence is such that,

$$\begin{aligned} &d(Q,P)^x > 0 \\ &d(Q,P)^x = 0 \iff Q(\boldsymbol{y}|x;\theta) = P(\boldsymbol{y}|x). \end{aligned}$$

If the integral of a non-negative, defined and measurable function is 0 then the function is 0 almost everywhere (a.e.) (Leadbetter et al., 2014, Theorem 4.4.7). Since $\mathrm{DISCO}_\Delta(P,Q,P_x)$ is the integral of $d(Q,P)^x$ over $\mathcal{X}$, scoring rule theory

[2]The score $S(P,P)^x$ must be finite, which is not a restrictive condition. We consider the range of values of the output given by the data or the model's architecture to be bounded. Similarly we consider bounded kernels in the case of the Negative kernel score.

provides us with the following guarantee[3]:

$$\text{DISCO}_\Delta(P, Q, P_x) = 0 \iff Q(\boldsymbol{y}|x;\theta) = P(\boldsymbol{y}|x) \text{ a. e. on } \mathcal{X}. \tag{4.12}$$

We use the Energy score in our experiments on hand pose estimation in section 4.5.

4.4.2 DISCO Nets as the Negative kernel score

Consider a characteristic kernel $K : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, measuring the similarity between two outputs y and y'. We employ the loss function $\Delta(y, y') = K_{max} - K(y, y')$ that measures the distance between y and y'. The constant $K_{max} = \max_{y,y' \in \mathcal{Y}\times\mathcal{Y}} K(y, y')$ is the maximum value achievable by the kernel on the space $\mathcal{Y} \times \mathcal{Y}$ and only ensures that the loss is non-negative. With this Δ, the diversity terms are:

$$\text{DIV}_\Delta(P, Q)^x = K_{max} - \mathbb{E}_{y\sim P(\boldsymbol{y}|x)}\left[\mathbb{E}_{y'\sim Q(\boldsymbol{y}|x;\theta)}[K(y, y')]\right], \tag{4.13}$$

$$\text{DIV}_\Delta(Q, Q)^x = K_{max} - \mathbb{E}_{y'\sim Q(\boldsymbol{y}|x;\theta)}\left[\mathbb{E}_{y''\sim Q(\boldsymbol{y}|x;\theta)}[K(y', y'')]\right], \tag{4.14}$$

$$\text{DIV}_\Delta(P, P)^x = K_{max} - \mathbb{E}_{y'\sim P(\boldsymbol{y}|x)}\left[\mathbb{E}_{y''\sim P(\boldsymbol{y}|x)}[K(y', y'')]\right]. \tag{4.15}$$

We now show the connection with the Negative kernel score. The Negative kernel score of a distribution $T(\boldsymbol{y}|x)$, evaluated for a realisation $y \sim P(\boldsymbol{y}|x)$, is defined as follows (see Definition 2.5 and Theorem 2.1),

$$S(T, y)^x = \mathbb{E}_{y'\sim T(\boldsymbol{y}|x)}[K(y', y)] - \frac{1}{2}\mathbb{E}_{y',y''\sim T(\boldsymbol{y}|x)}[K(y', y'')].$$

[3]We already implicitly assumed $d(Q, P)^x$ measurable when we defined our objective function.

Again, we evaluate $P(\boldsymbol{y}|x)$ and $Q(\boldsymbol{y}|x;\theta)$ under this scoring rule, by computing their expected scores, that are,

$$S(Q,P)^x = \mathbb{E}_{y\sim P(\boldsymbol{y}|x)}\left[\mathbb{E}_{y'\sim Q(\boldsymbol{y}|x;\theta)}[K(y',y)]\right] - \frac{1}{2}\mathbb{E}_{y'\sim Q(\boldsymbol{y}|x;\theta)}\left[\mathbb{E}_{y''\sim Q(\boldsymbol{y}|x;\theta)}[K(y',y'')]\right], \tag{4.16}$$

$$S(P,P)^x = \frac{1}{2}\mathbb{E}_{y'\sim P(\boldsymbol{y}|x)}\left[\mathbb{E}_{y''\sim P(\boldsymbol{y}|x)}[K(y',y'')]\right]. \tag{4.17}$$

By plugging-in equations (4.13, 4.14, 4.15) into equation (4.4) and equations (4.16, 4.17) into equation (4.3), again we can express the DISCO Nets' objective as the expectation of the divergence associated. Note that the constant $K_{\max}$ cancels in equation (4.4). The Negative kernel score is a strictly proper scoring rule whenever the kernel K is characteristic. In this case, the objective function $\mathrm{DISCO}_{\Delta}(P,Q,P_x)$ with $\gamma = 1/2$ is such that:

$$\mathrm{DISCO}_{\Delta}(P,Q,P_x) = 0 \iff Q(\boldsymbol{y}|x;\theta) = P(\boldsymbol{y}|x) \text{ a. e. on } \mathcal{X}. \tag{4.18}$$

In the specific case of a finite output space $\mathcal{Y}$, the strict positive definiteness of the kernel K is sufficient to ensure strict propriety of the Negative kernel score (see section 2.1.3, and Theorem 2.1). We use the Negative kernel score in our experiments on psychology attributes ratings prediction in section 4.7.

4.4.3 Scoring rules for training and evaluation

In our experiments, we will always employ loss functions such that our training objective is expressed as a strictly proper scoring rule. This encourages the model to accurately capture the distribution of interest. We will compare DISCO Nets to other existing implicit models. Implicit models (including DISCO Nets) do not return probability values. Therefore, we cannot compute likelihood values of test

data without using Kernel density estimation (KDE), which is known to fail in high-dimensional output spaces (Fukumizu et al., 2013). However, we explain in section 2.1.2 that scoring rules can be used to perform model evaluation and comparison. Thereby, we will use as probabilistic evaluation metric the DISCO Nets' objective, corresponding to a strictly proper scoring rule divergence.

4.5 Hand pose estimation experiment

Given a depth image x, which often contains occlusions and missing values, we wish to capture the distribution of $\boldsymbol{y}$, the pose of the hand. We perform experiment using the NYU Hand Pose data set (Tompson et al., 2014).

4.5.1 Setting

Loss Δ employed. For this experiment, we rely on the Energy score, that is, we use the loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}, \Delta(y, y') = \|y - y'\|_2^\beta$ with $\beta = 1$, which is the Euclidean norm. With this loss, the DISCO Nets' training objective is

$$\begin{aligned} \mathrm{F}_\Delta(P, Q, P_x) = \; & \mathbb{E}_{x \sim P_x(\boldsymbol{x})} \Big[\mathbb{E}_{y \sim P(\boldsymbol{y}|x)} \big[\mathbb{E}_{y' \sim Q(\boldsymbol{y}|x;\theta)} \big[\|y - y'\|_2^1 \big] \big] \\ & - \gamma \mathbb{E}_{y' \sim Q(\boldsymbol{y}|x;\theta)} \big[\mathbb{E}_{y'' \sim Q(\boldsymbol{y}|x;\theta)} \big[\|y' - y''\|_2^1 \big] \big] \Big]. \end{aligned}$$

The above training objective corresponds to $\mathrm{DISCO}_\Delta(P, Q, P_x)$ without the constant term $\mathrm{DIV}_\Delta(P, P, P_x)$. When $\gamma = 1/2$, our training objective can be viewed as a strictly proper scoring rule and guarantees equation (4.12).

Methods. Our goal with this experiment is two-fold. First, we want to assess the advantages of DISCO Nets with respect to non-probabilistic deep networks. To do so, we report classic non-probabilistic metrics for hand pose estimation employed in Oberweger et al. (2015a,b) and Taylor et al. (2012). These metrics use

the Euclidean distance between a single prediction and the ground-truth annotation. They require a single pointwise prediction, which we elect with the MEU method among K candidates. Specifically, we report the Mean Joint Euclidean Error (MeJEE), the Max Joint Euclidean Error (MaJEE) and the Fraction of Frames within distance (FF). FF(d) is the fraction of test examples that have all predicted joints of the pointwise pose below a given maximum Euclidean distance d in mm from the ground-truth. Note that for the metric FF, higher is better. The detailed expression of these metrics and their computation is in appendix 7.5.2.

Second, we want to compare DISCO Nets with existing deep probabilistic models, specifically the conditional GAN (cGAN) model. To do so, we need a way to evaluate the quality of the distributions captured by DISCO Nets and cGAN. Both models are implicit models, therefore they do not provide tractable likelihoods. As explained in section 4.4.3, we use as probabilistic evaluation metric the DISCO Nets' objective with $\gamma = 1/2$, that is, $\mathrm{DISCO}_\Delta(P, Q, P_x)$ with the Energy score. In this experiment, we only have one ground-truth annotation y_i per input sample x_i and cannot compute an estimate of $\mathrm{DIV}_\Delta(P, P, P_x)$, as we would need at least 2 annotations y_i, y_i' per input x_i. This is not a problem since our goal is to compare models, in other words, to assess if $\mathrm{DISCO}_\Delta(P, Q_1, P_x) < \mathrm{DISCO}_\Delta(P, Q_2, P_x)$ where Q_1, Q_2 are the distributions respectively captured by DISCO Nets and cGAN. The term $\mathrm{DIV}_\Delta(P, P, P_x)$ is a constant depending only on the data and disappears. We can equivalently assess if $\mathrm{F}_\Delta(P, Q_1, P_x) < \mathrm{F}_\Delta(P, Q_2, P_x)$. Specifically, we compute $\widehat{\mathrm{F}}_\Delta(P, Q, P_x)$ (equation (4.2), with $\gamma = 1/2$) on the test set, using K samples per input, and report the corresponding value as F_Δ.

NYU Hand Pose data set. The NYU Hand Pose data set (Tompson et al., 2014) contains $8,252$ testing and $72,757$ training frames of captured RGBD data with 1 ground-truth hand pose information. The training set is composed of images

of one person whereas the testing set is composed of images of two persons. For each frame, the RGBD data from 3 Kinects is provided: a frontal view and 2 side views. In our experiments we use only the depth data from the frontal view. While the ground-truth contains $J = 36$ annotated joints, we follow the evaluation protocol of Oberweger et al. (2015a,b) and use the same subset of $J = 14$ joints. We use $10,000$ examples from the $72,757$ training frames to construct a validation data set and train only on $62,757$ examples. We describe the preprocessing of the data set in appendix 7.5.2.

Models and architectures. The model BASE stands for taking (i) $\gamma = 0$ in the training objective $\mathrm{F}_\Delta(P, Q, P_x)$ and (ii) no noise is concatenated. This corresponds to a classic deep network which, for a given input x, generates a single output $y = G(x;\theta)$[4]. In order to use the MEU method for the model BASE, we construct K candidates by adding Gaussian random noise of mean 0 and diagonal covariance $\Sigma = \sigma^2 1$, with $\sigma^2 \in \{1\mathrm{mm}^2, 5\mathrm{mm}^2, 10\mathrm{mm}^2\}$ to the single output. We refer to this model as BASE_σ.

The models referred to as DISCO_γ employ the training objective $\mathrm{F}_\Delta(P, Q, P_x)$ with different values of γ. When $\gamma = 0$, noise is injected and the network capacity is the same as $\mathrm{DISCO}_{\gamma\neq 0}$. This is to assess that sampling from the random noise is not just preventing overfitting. When $\gamma = 1/2$, $\mathrm{DISCO}_{\gamma=1/2}$'s objective has the unique minimum guarantee from strictly proper scoring rule theory. The novelty of DISCO Nets resides in their objective function and they do not require the use of a specific network architecture. This allows us to employ a network architecture inspired by Oberweger et al. (2015a), detailed in appendix 7.5.2.

[4]Note that we write $G(x;\theta)$ and not $G(x, z;\theta)$ since no noise is concatenated.

Model	F_Δ (mm)	MeJEE (mm)	MaJEE (mm)	FF (80mm)
$BASE_{\sigma=1}$	103.8±0.627	25.2±0.152	52.7±0.290	86.040
$BASE_{\sigma=5}$	99.3±0.620	25.5±0.151	52.9±0.289	85.773
$BASE_{\sigma=10}$	96.3±0.612	25.7±0.149	53.2±0.288	85.664
$DISCO_{\gamma=0}$	92.9±0.533	21.6±0.128	46.0±0.251	92.971
$DISCO_{\gamma=0.25}$	89.9±0.510	21.2±0.122	46.4±0.252	93.262
$DISCO_{\gamma=1/2}$	**83.8 ±0.503**	**20.9±0.124**	**45.1±0.246**	**94.438**

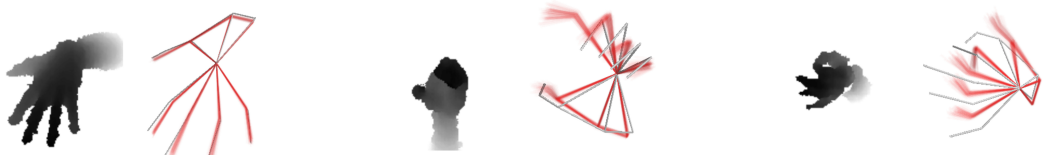
Table 4.1: Metrics values on the testing set ± standard error of the mean computed using $K = 100$ sampled outputs per test image. Best performances in bold.

4.5.2 Results

Quantitative evaluation. Table 4.1 reports performances on the testing data set, with parameters cross-validated on the validation set. All versions of DISCO Nets outperform the BASE model. Among the different values of γ, we see that $\gamma = 1/2$ better captures the true distribution while retaining accurate performance on the standard pointwise metrics. This shows that using $\gamma = 1/2$, which guarantees that DISCO Nets' objective is minimised only at the true distribution, is the most relevant training objective.

Qualitative evaluation. In Figure 4.3 we show candidate poses generated by $DISCO_{\gamma=1/2}$ for 3 test images. The left image shows the input depth image, and the right image shows the ground-truth pose (in grey) with 100 candidate outputs (superimposed in transparent red). The model predicts the joint locations and we interpolate the joints with edges. If an edge is thinner and more opaque, it means the different predictions overlap and that the uncertainty in the location of the edge's joints is low. We can see that $DISCO_{\gamma=1/2}$ captures relevant uncertainty in the structure of the hand[5].

[5]We note an offset in the predicted poses, it might be due to the fact that the training set (hence the validation set) contains images of a single person, while the testing set contains samples from two persons. We cannot verify this hypothesis since, to the best of our knowledge, the test samples are not annotated with the person identity.



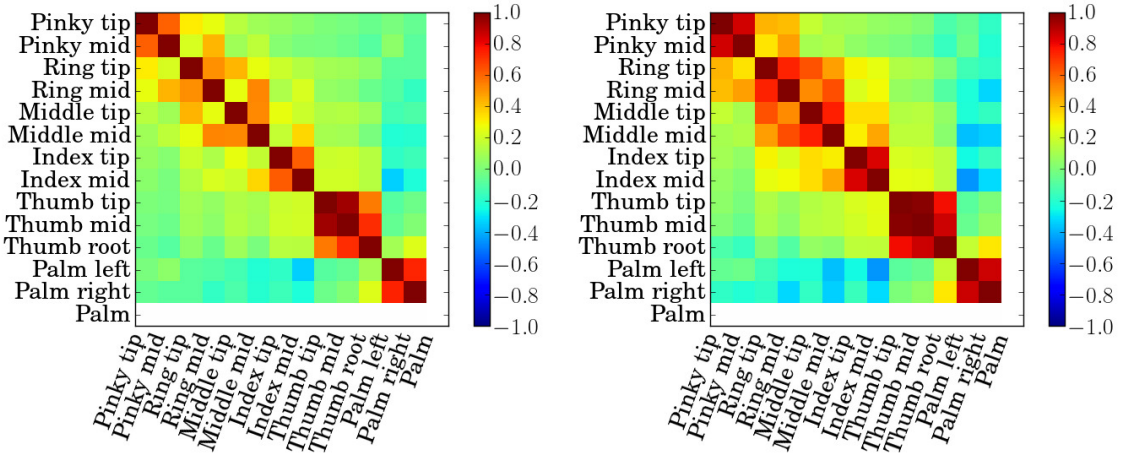
(a) When there are no occlusions, DISCO Nets model low uncertainty in all joints.

(b) When the hand is half-fisted, DISCO Nets model the uncertainty in the location of the fingertips.

(c) Here the fingertips of all fingers but the forefinger are occluded and DISCO Nets model high uncertainty in them.

Figure 4.3: DISCO$_{\gamma=1/2}$ *predictions for* 3 *testing samples. The left image shows the input depth image, and the right image shows the ground-truth pose in grey with* 100 *candidate outputs superimposed in transparent red. The depth images are from the NYU Hand Pose data set (Tompson et al., 2014), preprocessed as in Oberweger et al. (2015b).*

Figure 4.4 shows the matrices of Pearson product-moment correlation coefficients between joints. We compute these matrices by sampling 100 outputs per input, and computing the Pearson product-moment correlation coefficient matrix of each test input. Then, we compute the mean of the per-input matrices, and show the resulting matrix. We note that DISCO Nets with DISCO$_{\gamma=1/2}$ capture more intuitive correlations between the joints of a finger and between the fingers.



(a) $\gamma = 0$ *(b)* $\gamma = 1/2$

Figure 4.4: Pearson coefficients matrices of the joints. Palm has no value as certain input empirical variances are null.

Model	F_Δ (mm)	MeJEE (mm)	MaJEE (mm)	FF (80mm)
cGAN	442.7±0.513	109.8±0.128	201.4±0.320	0.000
$cGAN_{\text{init, fixed}}$	128.9±0.480	31.8±0.117	64.3±0.230	78.454
$DISCO_{\gamma=1/2}$	**83.8 ±0.503**	**20.9±0.124**	**45.1±0.246**	**94.438**

Table 4.2: Metrics values on the testing set ± standard error of the mean for cGAN, and $DISCO_{\gamma=1/2}$ performance for comparison. Best performances in bold.

Comparison with existing probabilistic models. In order to compare cGAN to DISCO Nets, several issues must be overcome. First, we must design a network architecture for the Discriminator. This is a first disadvantage of cGAN compared to DISCO Nets which require no adversary. Second, as mentioned in Goodfellow et al. (2014) and Radford et al. (2015), GAN (and thus cGAN) require very careful design of the networks' architecture and training procedure. In order to do a fair comparison, we followed the work in Mirza and Osindero (2014) and practical advice for GAN presented in Larsen and Sønderby (2015). We try (i) cGAN, initialising all layers of D and G randomly, and (ii) $cGAN_{\text{init, fixed}}$ initialising the convolutional layers of G and D with the trained best-performing $DISCO_{\gamma=1/2}$, and keeping these layers fixed. That is, the convolutional parts of G and D are fixed feature extractors for the depth image. This is a setting similar to the one employed for tag-annotation of images in Mirza and Osindero (2014). We report the results in Table 4.2.

Table 4.2 shows that the cGAN model obtains satisfactory results only when the convolutional layers of G and D are initialised with our trained model and kept fixed, that is $cGAN_{\text{init, fixed}}$. These results are still outperformed by $DISCO_{\gamma=1/2}$. As we mention in section 4.2, since our publication (Bouchacourt et al., 2016) research has developed the training of GAN-based models. While there may be a better architecture for cGAN, our experiments demonstrate the difficulty of training cGAN over DISCO Nets.

Comparison to state-of-the-art Finally, we compare DISCO Nets to state-of-the-art non-probabilistic models, specifically designed for hand pose estimation.

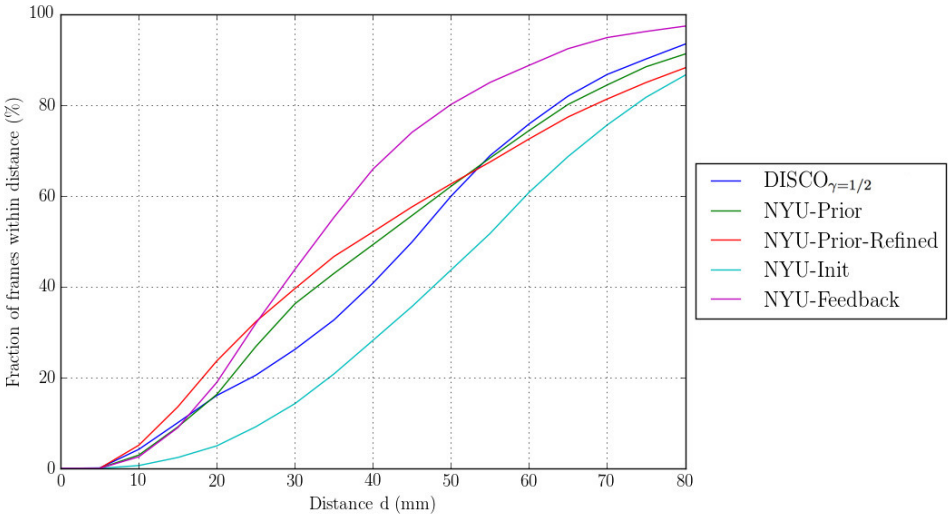


Figure 4.5: Fractions of frames within distance d in (by 5 mm).

In Oberweger et al. (2015b) a constrained prior hand model, referred to as NYU-Prior, is refined on each hand joint position to increase accuracy, referred to as NYU-Prior-Refined. In Oberweger et al. (2015b), the input depth image is fed to a first network NYU-Init, which outputs a pose used to synthesise an image with a second network. The synthesised image is used with the input depth image to derive a pose update. We refer to the whole model as NYU-Feedback. On the contrary, DISCO Nets use a single network with a network architecture similar to the one of NYU-Prior (without constraining on a pose prior). We train the best-performing DISCO$_{\gamma=1/2}$ on the entire training set, and compare performances with state-of-the-art models in Table 4.3 and Figure 4.5 on the testing set.

Model	MeJEE (mm)	MaJEE (mm)	FF (80mm)
NYU-Prior	20.7±0.150	44.8±0.289	91.190
NYU-Prior-Refined	19.7±0.157	44.7±0.327	88.148
NYU-Init	27.4±0.152	55.4±0.265	86.537
NYU-Feedback	**16.0±0.096**	**36.1±0.208**	**97.334**
DISCO$_{\gamma=1/2}$	20.7±0.121	45.1±0.246	93.250

Table 4.3: DISCO Nets and state-of-the-art performances ± standard error of the mean. The architecture most similar to DISCO Nets is NYU-Prior. Best performances in bold.

By accurately modeling the distribution of the pose given the depth im-

age, DISCO Nets obtain comparable performances to NYU-Prior and NYU-Prior-Refined on classic hand pose estimation metrics. Without any extra effort, DISCO Nets could be embedded in the more complicated NYU-Prior-Refined and NYU-Feedback methods, possibly boosting the performance of the state-of-the-art.

4.6 DISCO Nets for discrete outputs

Many important problems in structured output prediction are discrete by nature (Nowozin and Lampert, 2011). Take the example of predicting how one would rate a person's face on attributes such as *kind* and *attractive*. For a specific rater, we expect each attribute's rating to exhibit strong inter-dependencies. In addition, different raters assign different ratings, especially on subjective attributes such as kindness. Figure 4.6 shows a face image from the US10k Face Data set (Bainbridge et al., 2013), rated on 20 attributes by 3 Amazon Mechanical Turk (AMT) workers (each color corresponds to an AMT worker). We see the discrepancies among ratings, for example the ratings on the attribute *introverted* are {1,4,3} and on *trustworthy* are {4,9,8}. This results in a complex joint distribution of the attributes' ratings for a given image.

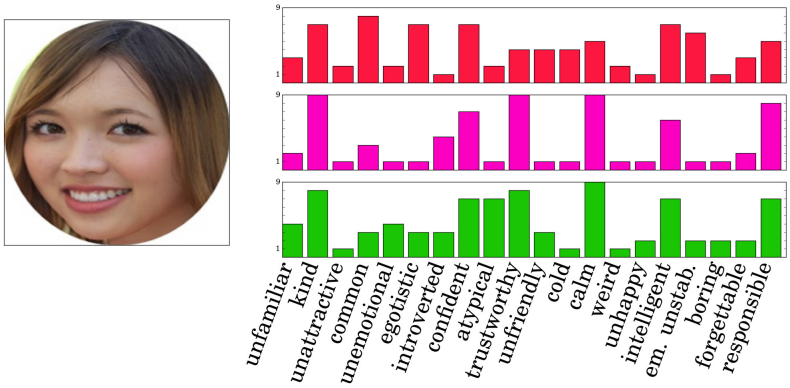


Figure 4.6: The face image is rated on 20 attributes (em. unstab. stands for emotionally unstable) by 3 different AMT workers on a range from 1 to 9. Each color corresponds to one AMT worker. The face image and ratings values are part of the US10k Face Data set (Bainbridge et al., 2013).

4.6.1 The difficulty of discrete spaces

We explain in section 2.3.1 how deep probabilistic models, including DISCO Nets, use the Law of the Unconscious Statistician (LOTUS) to compute the gradient of their objective function. This is no longer applicable when the distribution to capture is discrete. Therefore, the VAE model handles discrete output variables, but not discrete latent spaces. By contrast, the original GAN model handles discrete latent spaces but not discrete outputs. An approach for training GAN on discrete sequences is proposed in Yu et al. (2017) with the REINFORCE method (Williams, 1992). However the REINFORCE method leads to a high-variance estimator that converges slowly and gives poor performances (Paisley et al., 2012). London and Schwing (2016) introduce a smoothed generator to apply GANs to discrete outputs. The "Gumbel-softmax trick" allows the reparametrisation of discrete distributions using the Gumbel distribution and the arg max operator. This trick is used in Jang et al. (2017); Kusner and Hernández-Lobato (2016); Maddison et al. (2017), where the non-differentiable arg max operator is approximated by the softmax function and a temperature parameter. Rolfe (2017) smooth the discrete latent space of the VAE model, and Zhao et al. (2017a) overcome the difficulty by encoding the discrete data into a continuous space.

However, these methods fail to take into account the task-specific loss. For example in semantic segmentation, the evaluation loss is the intersection-over-union (IoU) or Jaccard Index (Jaccard, 1901). However, most semantic segmentation models employ differentiable losses such as cross-entropy and research is advancing towards directly optimising the IoU (Rahman and Wang, 2016). Encouraged by the positive results of DISCO Nets on the task of hand pose estimation, we extend DISCO Nets to the modelling of discrete structured output distributions. Similar to GAN, the training procedure of DISCO Nets involves a non-differentiable step in

this case. Since we need samples from the model to compute the loss in our objective function, approximating the arg max operator with the softmax function at training is not applicable. Inspired by the Direct Loss Minimisation method (McAllester et al., 2010; Song et al., 2016), we derive an optimisation algorithm to extend DISCO Nets to discrete output variables.

4.6.2 Prediction

We consider a structured discrete output $y \in \mathcal{Y}$ corresponding to an input $x \in \mathcal{X}$. The sampling process is similar to the continuous case, with the addition of a discretisation step. Specifically, given input x, a random noise z, and model parameters θ, DISCO Nets output a *grades set* $G_{\mathcal{Y}}(x, z; \theta)$, that is, a set of values $G_{\tilde{y}}(x, z; \theta)$ for each possible $\tilde{y} \in \mathcal{Y}$. The *grade* of a particular $\tilde{y}$ is $G_{\tilde{y}}(x, z; \theta)$[6]. From this output we perform:

$$\hat{y} = \arg\max_{\tilde{y} \in \mathcal{Y}} G_{\tilde{y}}(x, z; \theta). \tag{4.19}$$

We call this process the *maximal grade prediction process*, and $\hat{y}$ the *maximal graded output*. The sampling process is shown in Figure 4.7. We assume $\mathcal{Y}$ to be finite with no ties in maximisation, and $x \in \mathcal{X}$ and $z \in \mathcal{Z}$ to remain continuous.

4.6.3 A non-differentiable training procedure

Given a pair of input and random noise samples (x, z) and the model parameters θ, the maximal grade prediction process is a deterministic function. Recall our training

[6]We purposely employ the term *grade* and not *score* to avoid any confusion with scoring rule theory.

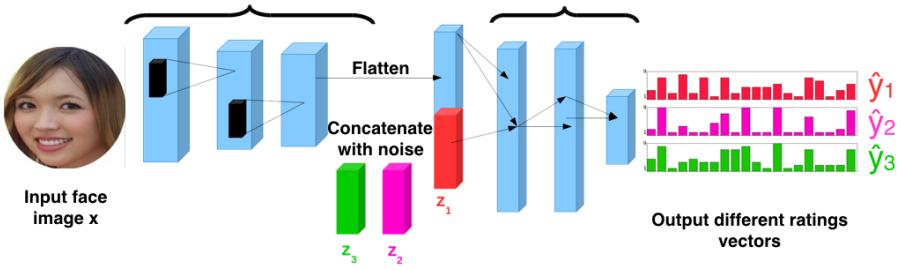


Figure 4.7: DISCO Nets sampling process for discrete outputs. By using 3 random samples z_1, z_2, z_3, DISCO Nets output 3 grades sets $G_{\mathcal{Y}}(x, z_1; \theta)$, $G_{\mathcal{Y}}(x, z_2; \theta)$, $G_{\mathcal{Y}}(x, z_3; \theta)$, and we obtain 3 maximal graded outputs $\hat{y}_1, \hat{y}_2, \hat{y}_3$. The face image is from the US10k Face Data set (Bainbridge et al., 2013).

objective function is as follows,

$$\mathrm{F}_{\Delta}(P, Q, P_x) = \mathrm{DIV}_{\Delta}(P, Q, P_x) - \gamma \mathrm{DIV}_{\Delta}(Q, Q, P_x).$$

$\mathrm{F}_{\Delta}(P, Q, P_x)$ corresponds to $\mathrm{DISCO}_{\Delta}(P, Q, P_x)$ without the term $\mathrm{DIV}_{\Delta}(P, P, P_x)$, which is a constant depending on the data and does not appear in the gradient. Using the Law of the Unconscious Statistician (LOTUS) (see section 2.3.1), we write the expectations with respect to $Q(\boldsymbol{y}|x;\theta)$ as expectations over the random variable $\boldsymbol{z}$. This gives us the following,

$$\begin{aligned}
\mathrm{DIV}_{\Delta}(P, Q, P_x) &= \mathbb{E}_{x \sim P_x(\boldsymbol{x})}\left[\mathbb{E}_{y \sim P(\boldsymbol{y}|x)}\left[\mathbb{E}_{y' \sim Q(\boldsymbol{y}|x;\theta)}[\Delta(y, y')]\right]\right] \\
&= \mathbb{E}_{x \sim P_x(\boldsymbol{x})}\left[\mathbb{E}_{y \sim P(\boldsymbol{y}|x)}\left[\mathbb{E}_{z \sim P_z(\boldsymbol{z})}[\Delta(y, \hat{y})]\right]\right], \\
\mathrm{DIV}_{\Delta}(Q, Q, P_x) &= \mathbb{E}_{x \sim P_x(\boldsymbol{x})}\left[\mathbb{E}_{y \sim Q(\boldsymbol{y}|x;\theta)}\left[\mathbb{E}_{y' \sim Q(\boldsymbol{y}|x;\theta)}[\Delta(y, y')]\right]\right] \\
&= \mathbb{E}_{x \sim P_x(\boldsymbol{x})}\left[\mathbb{E}_{z \sim P_z(\boldsymbol{z})}\left[\mathbb{E}_{z' \sim P_z(\boldsymbol{z})}[\Delta(\hat{y}, \hat{y}')]\right]\right],
\end{aligned}$$

where $\hat{y}, \hat{y}'$ are maximal graded outputs,

$$\hat{y} = \arg\max_{\tilde{y} \in \mathcal{Y}} G_{\tilde{y}}(x, z; \theta),$$

$$\hat{y}' = \arg\max_{\tilde{y} \in \mathcal{Y}} G_{\tilde{y}}(x, z'; \theta).$$

The gradients $\nabla_\theta \mathrm{DIV}_\Delta(P, Q, P_x)$ and $\nabla_\theta \mathrm{DIV}_\Delta(Q, Q, P_x)$ can exist, but the use of the arg max operator renders the sampling process non-differentiable with respect to θ. That is, we cannot compute an estimate of the gradient using the LOTUS. The term $\mathrm{DIV}_\Delta(P, Q, P_x)$ corresponds to the classic structured output prediction objective, that is, the expected loss between the prediction and the ground-truth. Therefore it admits a convex upper bound, or can be optimised using the Direct Loss Minimisation method (DLM) (McAllester et al., 2010; Song et al., 2016). There is no existing method to optimise $\mathrm{DIV}_\Delta(Q, Q, P_x)$, which corresponds to the expected value of the loss between two samples both drawn from the model's distribution $Q(\boldsymbol{y}|x; \theta)$. We solve this issue by deriving an approximation of the gradient of the second term of the DISCO Nets' objective with a temperature parameter, similar to McAllester et al. (2010); Song et al. (2016). The gradient is provided in Theorem 4.1. When the temperature parameter $\epsilon > 0$ goes to the limit value 0, denoted $\epsilon \searrow 0$, we retrieve the true gradient.

Theorem 4.1 (DISCO Nets Loss gradient theorem)**.** *For a finite space $\mathcal{Y}$, under some regularity conditions (detailed in the proof), the gradient $\nabla_\theta \mathrm{F}_\Delta(P, Q, P_x)$ is:*

$$\begin{aligned}\nabla_\theta \mathrm{F}_\Delta(P, Q, P_x) &= \pm \lim_{\epsilon \searrow 0} \frac{1}{\epsilon} \nabla_\theta \mathrm{F}^\epsilon_\Delta(P, Q, P_x) \\ &= \pm \lim_{\epsilon \searrow 0} \frac{1}{\epsilon} \Big(\nabla_\theta \mathrm{DIV}^\epsilon_\Delta(P, Q, P_x) - \frac{1}{2} \nabla_\theta \mathrm{DIV}^\epsilon_\Delta(Q, Q, P_x) \Big),\end{aligned}$$

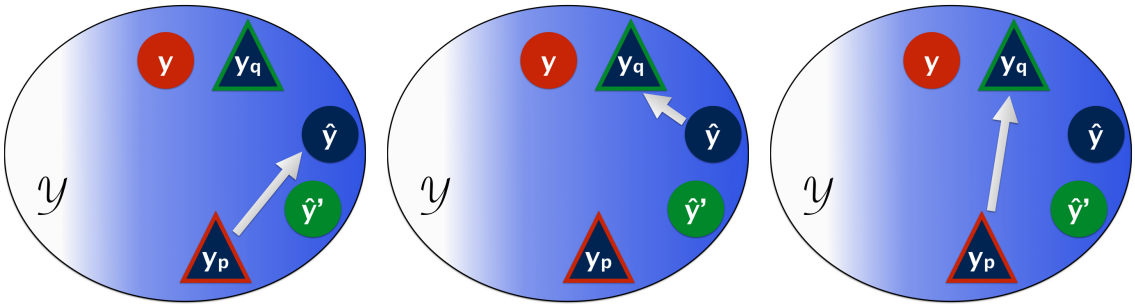
where

$$\begin{aligned}&\nabla_\theta \mathrm{DIV}^\epsilon_\Delta(P, Q, P_x) \\ &\quad = \mathbb{E}_{x \sim P_x(\boldsymbol{x})} \big[\mathbb{E}_{y \sim P(\boldsymbol{y}|x)} \big[\mathbb{E}_{z \sim P_z(\boldsymbol{z})} \big[\nabla_\theta G_{y_p}(x, z; \theta) - \nabla_\theta G_{\hat{y}}(x, z; \theta) \big] \big] \big], \\ &\hat{y} = \arg\max_{\tilde{y} \in \mathcal{Y}} G_{\tilde{y}}(x, z; \theta), \\ &y_p = \arg\max_{\tilde{y} \in \mathcal{Y}} G_{\tilde{y}}(x, z; \theta) \pm \epsilon \Delta(\tilde{y}, y), \\ &\nabla_\theta \mathrm{DIV}^\epsilon_\Delta(Q, Q, P_x) \\ &\quad = 2 \ \mathbb{E}_{x \sim P_x(\boldsymbol{x})} \big[\mathbb{E}_{z \sim P_z(\boldsymbol{z})} \big[\mathbb{E}_{z' \sim P_z(\boldsymbol{z})} \big[\nabla_\theta G_{y_q}(x, z; \theta) - \nabla_\theta G_{\hat{y}}(x, z; \theta) \big] \big] \big], \\ &\hat{y}' = \arg\max_{\tilde{y} \in \mathcal{Y}} G_{\tilde{y}}(x, z'; \theta), \\ &y_q = \arg\max_{\tilde{y} \in \mathcal{Y}} G_{\tilde{y}}(x, z; \theta) \pm \epsilon \Delta(\tilde{y}, \hat{y}').\end{aligned}$$

Proof. In appendix 7.5.1. □

The $\pm$ sign means that the theorem is true (i) when one replaces $\pm$ by a $+$ everywhere (positive update direction) (ii) $\pm$ by a $-$ everywhere (negative update direction), but the signs cannot be mixed. Let us give an intuitive explanation of Theorem 4.1. We consider the positive update direction, that is, we replace all $\pm$ with a $+$ and this gives $\nabla_\theta \mathrm{F}_\Delta(P, Q, P_x) = \lim_{\epsilon \searrow 0} \frac{1}{\epsilon} \nabla_\theta \mathrm{F}^\epsilon_\Delta(P, Q, P_x)$. We fix $\epsilon > 0$. Using gradient descent, the model parameters are updated in the direction:

$$-\nabla_\theta \mathrm{F}^\epsilon_\Delta(P, Q, P_x) = \frac{1}{2} \nabla_\theta \mathrm{DIV}^\epsilon_\Delta(Q, Q, P_x) - \nabla_\theta \mathrm{DIV}^\epsilon_\Delta(P, Q, P_x). \tag{4.20}$$



(a) $\nabla_\theta \mathrm{DIV}^\epsilon_\Delta(P, Q, P_x)$ *pushes towards* $\hat{y}$ *(maximal graded output), away from* y_p *(maximal loss-augmented graded output w.r.t. the ground-truth.)*

(b) $\nabla_\theta \mathrm{DIV}^\epsilon_\Delta(Q, Q, P_x)$ *pushes away from* $\hat{y}$ *and towards* y_q *(maximal loss-augmented graded output w.r.t. the other maximal graded output* $\hat{y}'$*), encouraging diversity between* $\hat{y}$ *and* $\hat{y}'$*.*

(c) The case $\gamma = 1/2$ *pushes directly away from* y_p*, a high-graded output with high loss w.r.t. the ground-truth towards* y_q*, a high-graded output with high loss (hence, diverse) w.r.t. the other maximal graded output* $\hat{y}'$*.*

Figure 4.8: Illustration of the Loss gradient theorem for DISCO Nets. Blue and green circles represent maximal graded outputs, and the ground-truth in red circle. Triangles represent loss-augmented maximal graded outputs, with the border corresponding to the output with respect to (w.r.t.) which the loss is computed (ground-truth or other maximal graded output). The shading represents values of the grades set $G_\mathcal{Y}(x, z; \theta)$ *over the space* $\mathcal{Y}$ *(darker is higher).*

Consider an input x, a ground-truth sampled output $y \sim P(\boldsymbol{y}|x)$. From two noise samples z, z', DISCO Nets output two grades sets $G_\mathcal{Y}(x, z; \theta)$ and $G_\mathcal{Y}(x, z'; \theta)$. These grades sets respectively have the corresponding maximal graded outputs $\hat{y}$ and $\hat{y}'$. We look at the two terms in equation (4.20), and how the move they induce on the grades set $G_\mathcal{Y}(x, z; \theta)$.

- $-\nabla_\theta \mathrm{DIV}^\epsilon_\Delta(P, Q, P_x)$: the move in the output space corresponding to this term is shown in Figure 4.8a. The term $-\nabla_\theta \mathrm{DIV}^\epsilon_\Delta(P, Q, P_x)$ makes the parameters move along the direction:

$$-\nabla_\theta G_{y_p}(x, z; \theta) + \nabla_\theta G_{\hat{y}}(x, z; \theta).$$

This updates the model parameters in the direction of the maximal graded output $\hat{y}$ of the grades set $G_{\mathcal{Y}}(x, z; \theta)$. At the same time, it encourages the model to move in the opposite direction of $\nabla_\theta G_{y_p}(x, z; \theta)$, that is, away from y_p. Recall that y_p has the maximal loss-augmented grade $G_{\tilde{y}}(x, z; \theta) + \epsilon\Delta(\tilde{y}, y)$ with respect to the true data distribution sample y. Moving away from y_p encourages the model to move away from a prediction with high grade but also high loss with respect to the ground-truth.

- $\nabla_\theta \text{DIV}^\epsilon_\Delta(Q, Q, P_x)$: the move in the output space corresponding to this term is shown in Figure 4.8b. The term $\nabla_\theta \text{DIV}^\epsilon_\Delta(Q, Q, P_x)$ makes the parameters move along the direction:

$$\nabla_\theta G_{y_q}(x, z; \theta) - \nabla_\theta G_{\hat{y}}(x, z; \theta).$$

This updates the model parameters in the direction of y_q. Here, y_q is the output of maximal loss-augmented grade $G_{\tilde{y}}(x, z; \theta) + \epsilon\Delta(\tilde{y}, \hat{y}')$ with respect to the other maximal graded output $\hat{y}'$. This encourages diversity between the two maximal graded outputs, $\hat{y}$ and $\hat{y}'$, for a given x. At the same time, it makes the model move away from $\hat{y}$.

- Figure 4.8c corresponds to the specific case $\gamma = 1/2$. When $\gamma = 1/2$, the update of the gradient away from $\hat{y}$ in (b) cancels out with the update towards $\hat{y}$ in (a). In this case, the direction of the gradient is directly away from y_p, a high-graded output with high loss with respect to the ground-truth, and towards y_q, a high-graded output with high loss with respect to the other maximal graded output $\hat{y}'$, encouraging diverse and accurate samples. This cancelling effect is expected. For $\gamma = 1/2$ the DISCO Net's objective is a strictly proper scoring rule, therefore it is maximised at the true data distribution, where the gradient is 0. Consider that the model is at this maximum,

that is, the model's samples are drawn from the same distribution as ground-truth samples. Then, y_p (maximal loss-augmented graded output with respect to the ground-truth) and y_q (maximal loss-augmented graded output with respect the model sample) are the same (precisely, the same on expectation), and the gradient update is indeed 0.

Theorem 4.1 requires regularity conditions detailed in the appendix 7.5.1. In practise, if these conditions are not met or if the gradients $\nabla_\theta \mathrm{DIV}_\Delta(P, Q, P_x)$ and $\nabla_\theta \mathrm{DIV}_\Delta(Q, Q, P_x)$ are undefined, we still use our approximation to learn the model's parameters.

4.6.4 Optimisation procedure for discrete outputs

We consider a training data set composed of n examples input-output pairs $\mathcal{D} = \{(x_i, y_i), i = 1, \ldots, n\}$. We learn the parameters θ with mini-batch gradient descent. We construct an unbiased estimator of the gradient approximation by computing unbiased estimators of each diversity gradient approximation.

$$\widehat{\nabla}_\theta \mathrm{F}^\epsilon_\Delta(P, Q, P_x) = \widehat{\nabla}_\theta \mathrm{DIV}^\epsilon_\Delta(P, Q, P_x) - \gamma \widehat{\nabla}_\theta \mathrm{DIV}^\epsilon_\Delta(Q, Q, P_x). \tag{4.21}$$

Similar to the continuous setting, the estimators are computed using samples drawn from the data set and from DISCO Nets. The learning algorithm is shown in Algorithm 3 (shown for clarity for a single input-output pair $(x_i, y_i) \in \mathcal{D}$ and a single update of the parameters).

4.6.5 Strictly definite positive kernels on discrete data

In the discrete setting, we employ DISCO Nets expressed as the Negative kernel score, that is, using a loss $\Delta(y, y') = K_{max} - K(y, y')$ where K is a strictly positive definite (SPD) kernel. We list now existing SPD kernels on discrete data used in

Algorithm 3: DISCO Nets training algorithm for discrete outputs, for a single input-output pair $(x_i, y_i) \in \mathcal{D}$.

Sample K random noise vectors $(z_{i,1}, \ldots, z_{i,k})$ for input x_i.

Generate K grades sets $G_{\mathcal{Y}}(x_i, z_{i,k}; \theta^t), k = 1, \ldots, K$ for input x_i.

for *k=1,...,K* **do**

  Find maximal graded output,
  $\hat{y}_{i,k} = \arg\max_{\tilde{y} \in \mathcal{Y}} G_{\tilde{y}}(x, z_{i,k}; \theta^t)$.

  Find loss-augmented maximal graded output w.r.t. ground-truth y_i,
  $y_{i,p,k} = \arg\max_{\tilde{y} \in \mathcal{Y}} G_{\tilde{y}}(x, z_{i,k}; \theta^t) \pm \epsilon\Delta(\tilde{y}, y_i)$.

end

for *k=1,...,K* **do**

  for *k'=1,...,K,* $k' \neq k$ **do**

    Find loss-augmented maximal graded output w.r.t. other maximal graded output $\hat{y}_{i,k'}$,
    $y_{i,q,k,k'} = \arg\max_{\tilde{y} \in \mathcal{Y}} G_{\tilde{y}}(x, z_{i,k}; \theta^t) \pm \epsilon\Delta(\tilde{y}, \hat{y}_{i,k'})$.

  end

end

Compute approximated gradient estimates:

$$\widehat{\nabla_\theta} \mathrm{DIV}^{\epsilon}_{\Delta}(P, Q, P_x)|_{\theta^t} = \frac{1}{K} \sum_{k=1}^{K} \left[\nabla_\theta G_{y_{p,i,k}}(x_i, z_{i,k}; \theta)|_{\theta^t} - \nabla_\theta G_{\hat{y}_{i,k}}(x_i, z_{i,k}; \theta)|_{\theta^t} \right],$$

$$\widehat{\nabla_\theta} \mathrm{DIV}^{\epsilon}_{\Delta}(Q, Q, P_x)|_{\theta^t} = 2 \frac{1}{K(K-1)} \sum_{k=1}^{K} \sum_{\substack{k'=1, \\ k' \neq k}}^{K} \left[\nabla_\theta G_{y_{q,i,k,k'}}(x_i, z_{i,k}; \theta)|_{\theta^t} - \nabla_\theta G_{\hat{y}_{i,k}}(x_i, z_{i,k}; \theta)|_{\theta^t} \right].$$

Update parameters $\theta^{t+1} \leftarrow \theta^t$ by descending the approximated gradient estimate $\widehat{\nabla_\theta} \mathrm{F}^{\epsilon}_{\Delta}(P, Q, P_x)|_{\theta^t}$.

real-world applications. We focus on non-decomposable kernels that do not decompose over the dimensions of the ouput, in order to have non-decomposable losses. First, this ensures that the model captures the structure of the output. Second, real-world applications often employ non-decomposable loss functions, such as the Jaccard index (or Intersection over Union) used in semantic segmentation of images.

Jaccard index. For binary-valued outputs $y \in \mathcal{Y} = \{0,1\}^A$, that is, $y = (y^1, \ldots, y^A), y^1, \ldots, y^A \in \{0,1\}$, the Jaccard index (Jaccard, 1901) between two outputs $(y, y') \in \mathcal{Y} \times \mathcal{Y}$ is defined as follows,

$$\mathrm{K}_{\mathrm{Jaccard}}(y, y') = \frac{\sum_{a=1}^{A} 1_{\{y^a=1 \wedge y'^a=1\}}}{\sum_{a=1}^{A} 1_{\{y^a=1 \vee y'^a=1\}}}.$$

Bouchard et al. (2013); Gower (1971) show that the Jaccard index is a SPD kernel.

MinMax kernel. If we consider range-valued outputs $y \in \mathcal{Y} = \{0, ..., R\}^A$, that is, $y = (y^1, \ldots, y^A), y^1, \ldots, y^A \in \{0, \ldots, R\}$, the MinMax kernel is defined between two outputs $(y, y') \in \mathcal{Y} \times \mathcal{Y}$ as follows,

$$\mathrm{K}_{\mathrm{MinMax}}(y, y') = \frac{\sum_{a=1}^{A} \min(y^a, y'^a)}{\sum_{a=1}^{A} \max(y^a, y'^a)}.$$

The MinMax kernel has been introduced for molecules structures and is a SPD kernel (Ralaivola et al., 2005). When $R = 1$, it is equivalent to the Jaccard index.

Exact loss-augmented inference. We focus on non-decomposable kernels such as the MinMax kernel. In this case, solving the loss-augmented problems in steps 4 and 5 of Algorithm 3 easily becomes intractable. Methods have been developed to find approximate solutions, for example Tarlow and Zemel (2012). However, in our experiments we use exact inference in order to perform a fair comparison. Using dynamic programming methods, we find exact solutions to the loss-augmented problems in pseudo-polynomial time. We briefly detail how in appendix 7.5.3.

4.7 Psychology attributes ratings prediction experiment

We tackle the challenge of predicting, given an input face image x, the distribution of y, the ratings over a collection of psychology attributes. That is, the output $y \in \{0, \ldots, R\}^A$ is composed of A integer values between 0 and R. This is a multi-label (A attributes) multi-class (R possible ratings) problem.

4.7.1 Setting

Loss Δ employed. For this experiment, we rely on the Negative kernel score, that is, we use the loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}, \Delta(y, y') = 1 - K_{\text{MinMax}}(y, y')$ where K_{MinMax} is the MinMax kernel, which maximal value is 1 in the case $y = y'$. With this loss, the DISCO Nets' training objective is

$$\mathrm{F}_{\Delta}(P, Q, P_x) = \mathbb{E}_{x \sim P_x(\boldsymbol{x})} \Big[\mathbb{E}_{y \sim P(\boldsymbol{y}|x)} \big[\mathbb{E}_{y' \sim Q(\boldsymbol{y}|x;\theta)} [1 - K(y, y')] \big] - \gamma \mathbb{E}_{y' \sim Q(\boldsymbol{y}|x;\theta)} \big[\mathbb{E}_{y'' \sim Q(\boldsymbol{y}|x;\theta)} [1 - K(y', y'')] \big] \Big].$$

which corresponds to $\mathrm{DISCO}_{\Delta}(P, Q, P_x)$ without the constant term $\mathrm{DIV}_{\Delta}(P, P, P_x)$. When $\gamma = 1/2$, strictly proper scoring rule theory guarantees equation (4.18).

The US10k Face Data set. The US10k Face Data Set (Bainbridge et al., 2013) contains 10,168 natural face photographs, among which 2,222 of the images have been annotated with memorability scores, computer vision and psychology attributes, and landmark point annotations. We tackle the prediction of the psychology attributes ratings of the face images. We describe the preprocessing of the data set in appendix 7.5.3. The attributes are rated on a scale from 1 to 9, which we rescale from 0 to 8. After preprocessing, each of the 2,222 images has 12

boring	calm	cold	confident	egotistic
emotionally unstable	intelligent	introverted	kind	responsible
trustworthy	unattractive	unhappy	weird	atypical
forgettable	unfamiliar	common	unemotional	unfriendly

Table 4.4: Psychology attributes from Bainbridge et al. (2013) that we consider.

annotations, that is, 12 vectors $y_1, \ldots, y_{12}$ of ratings over the 20 attributes. The attributes we consider are listed in Table 4.4. We split the data set into $1,022$ images for training, 600 for validation and 600 for testing.

Methods. As in the continuous case, we use as probabilistic evaluation metric the DISCO Nets' objective with $\gamma = 1/2$, that is, $\text{DISCO}_\Delta(P, Q, P_x)$ with the Negative kernel score. In this experiment, we have 12 ground-truth annotations $y_{i,1}, \ldots, y_{i,12}$ per input sample x_i and we can compute an estimate of $\text{DIV}_\Delta(P, P, P_x)$. Therefore, we report an estimate of $\text{DISCO}_\Delta(P, Q, P_x)$ on the test data set denoted DISCO_Δ. For each model, we draw K samples $\hat{y}_{i,k}$ per input x_i from the model's distribution and compute:

$$\begin{aligned}\text{DISCO}_\Delta = &\frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} \Big[\frac{1}{12} \sum_{l=1}^{12} \frac{1}{K} \sum_{k=1}^{K} \Delta(y_{i,l}, \hat{y}_{i,k}) \\ &- \frac{1}{2} \frac{1}{K(K-1)} \sum_{k=1}^{K} \sum_{\substack{k'=1 \\ k' \neq k}}^{K} \Delta(\hat{y}_{i,k}, \hat{y}_{i,k'}) - \frac{1}{2} \frac{1}{12 * 11} \sum_{l'=1}^{12} \sum_{\substack{l'=1 \\ l' \neq l}}^{12} \Delta(y_{i,l}, y_{i,l'}) \Big].\end{aligned}$$

As a pointwise evaluation metric, we report the value of the loss Δ. Specifically, for each test input x_i, we compute $\Delta(y_{i,k}, \hat{y}_{i,\text{MEU}})$ where $y_{i,l}, l = 1, \ldots, 12$ is a ground-truth rating for image x_i, and $\hat{y}_{i,\text{MEU}}$ is the output elected with the MEU method among rating samples drawn from the model. We report the mean value on the test set, as follows,

$$\Delta_{\text{MEU}} = \frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} \frac{1}{12} \sum_{l=1}^{12} \Delta(y_{i,l}, \hat{y}_{i,\text{MEU}}).$$

Architecture. Given an input and noise pair (x, z), we use a deep network that represents the grades set $G_{\mathcal{Y}}(x, z; \theta)$ as a matrix $M^{\theta}_{x,z} \in \mathbb{R}^{A \times R}$. In this experiment, $A = 20$ (number of attributes) and $R = 9$ (ratings go from 0 to 8). Recall that the input x is a face image and the output y is a 20-dimensional vector of ratings over the attributes. Each element of $M^{\theta}_{x,z}$ is a scalar value $M^{\theta}_{x,z}(a, r)$ that grades "$y^a = r$", that is, the dimension $a \in \{1, \ldots, A\}$ of y taking the value $r \in \{0, \ldots, R\}$. The total grade $G_y(x, z; \theta)$ of an output y is the sum of its per-dimension values:

$$G_y(x, z; \theta) = \sum_{a=1}^{A} G_{\tilde{y}}(x, z; \theta)^a = \sum_{a=1}^{A} \sum_{r=0}^{R} M^{\theta}_{x,z}(a, r) 1_{\{y^a = r\}}.$$

We use a standard CNN architecture described in appendix 7.5.3. In order to keep the values of the score matrix within a range, we use a softmax activation function after the last layer.

DISCO Nets model DISCO$_{\gamma=1/2}$. Despite the use of the softmax function, DISCO Nets do not consider the grades values as probabilities. The maximal grade prediction process is as follows,

$$\hat{y} = \arg\max_{\tilde{y} \in \mathcal{Y}} G_{\tilde{y}}(x, z; \theta) = \arg\max_{\tilde{y} \in \mathcal{Y}} \sum_{a=1}^{A} G_{\tilde{y}}(x, z; \theta)^a = \arg\max_{\tilde{y} \in \mathcal{Y}} \sum_{a=1}^{A} \sum_{r=0}^{R} M^{\theta}_{x,z}(a, r) 1_{\{\tilde{y}^a = r\}}$$

$$\iff \forall a \in \{1, \ldots, A\}, \hat{y}^a = \arg\max_{r \in \{0, \ldots, R\}} M^{\theta}_{x,z}(a, r).$$

We train DISCO Nets with $\gamma = 1/2$, which corresponds to a strictly proper scoring rule and guarantees equation (4.18).

Non-probabilistic model DISCO$_{\gamma=0}$. As in the experiment on hand pose estimation, we compare with the DISCO Nets model when $\gamma = 0$, that is, noise is injected and the model capacity is the same as DISCO$_{\gamma=1/2}$. We train DISCO$_{\gamma=0}$

by minimising the following Hinge-type upper bound on $\mathrm{DIV}_\Delta(P, Q, P_x)$[7]:

$$\mathrm{DIV}_\Delta(P, Q, P_x) \leq \mathbb{E}_{x\sim P_x(\boldsymbol{x})}\Big[\mathbb{E}_{y\sim P(\boldsymbol{y}|x)}[\mathbb{E}_{z\sim P_z(\boldsymbol{z})}[\max_{y'\in\mathcal{Y}}\{\Delta(y, y') + G_{y'}(x, z; \theta) - G_y(x, z; \theta)\}]]\Big].$$

Baseline probabilistic model CNN-CE. In order to compare DISCO Nets to a probabilistic model, we train the same CNN architecture using the cross-entropy (CE) loss. We refer to this model as CNN-CE. It considers the output matrix $M^\theta_{x,z} \in \mathbb{R}^{A\times R}$ of the last layer (after the softmax activation function) as the probabilities of the distribution $Q(\boldsymbol{y}|x, z; \theta)$. We define the marginalised pdf $q(\boldsymbol{y}|x; \theta) = \mathbb{E}_{z\sim P_z(\boldsymbol{z})}[q(\boldsymbol{y}|x, z; \theta)]$ to factorise over each dimension a of the output $\boldsymbol{y}$, as $q(\boldsymbol{y}|x; \theta) = \prod_{a=1}^{A} q(\boldsymbol{y}^a|x; \theta)$. The training of CNN-CE is done by minimising the negative log-likelihood of the data under this model, as follows,

$$\begin{aligned}
\text{CNN-CE}(P, Q, P_x) &= -\mathbb{E}_{x\sim P_x(\boldsymbol{x})}\big[\mathbb{E}_{y\sim P(\boldsymbol{y}|x)}[\log q(y|x; \theta)]\big] \\
&= -\mathbb{E}_{x\sim P_x(\boldsymbol{x})}\left[\mathbb{E}_{y\sim P(\boldsymbol{y}|x)}\left[\sum_{a=1}^{A} \log q(y^a|x; \theta)\right]\right] \\
&= -\mathbb{E}_{x\sim P_x(\boldsymbol{x})}\left[\mathbb{E}_{y\sim P(\boldsymbol{y}|x)}\left[\sum_{a=1}^{A} \log \mathbb{E}_{z\sim P_z(\boldsymbol{z})}[q(y^a|x, z; \theta)]\right]\right].
\end{aligned} \tag{4.22}$$

To sample from CNN-CE, we marginalise the variable $\boldsymbol{z}$ and we sample from the

[7]We also tried training the $\mathrm{DISCO}_{\gamma=0}$ model using the Direct Loss Minimisation method (McAllester et al., 2010; Song et al., 2016) but better performance for this model is obtained using the Hinge-type upper bound.

resulting factorised distribution:

$$\hat{y} \sim Q(\boldsymbol{y}|x;\theta) \iff \forall a \in \{1,\dots,A\}, \hat{y}^a \sim Q(\boldsymbol{y}^a|x;\theta),$$
$$\text{where } q(y^a = r|x;\theta) = \mathbb{E}_{z\sim P_z(\boldsymbol{z})}[q(y^a = r|x,z;\theta)] = \mathbb{E}_{z\sim P_z(\boldsymbol{z})}\left[M^{\theta}_{x,z}(a,r)\right]. \quad (4.23)$$

We denote the CNN-CE's sampling process as the *marginalised grade prediction process*. In comparison DISCO Nets performs sampling through the use of the arg max thresholding. The following remark explains that the CNN-CE's training objective in equation (4.22) is also guaranteed to be minimised at the true distribution. However, as the DISCO Nets' objective is tailored to the loss at hand, we expect better performance from DISCO Nets on this loss, since we assumed that we are under model misspecification.

Remark 4.1 (Maximum log-likelihood as a scoring rule)**.** *The objective function equation (4.22) also corresponds to the minimisation of the expectation of the divergence associated to a strictly proper scoring rule. In this case, the corresponding scoring rule is the Logarithmic score (see section 2.1.2). The Logarithmic score is strictly proper and is defined, for a distribution $T(\boldsymbol{y}|x)$, evaluated at $y \sim P(\boldsymbol{y}|x)$, as $S(T,y)^x = \log t(y|x)$. Under this scoring rule, the expected scores of $P(\boldsymbol{y})$ and $Q(\boldsymbol{y}|x;\theta)$, and the expectation of the associated divergence, are:*

$$S(Q,P)^x = \mathbb{E}_{y\sim P(\boldsymbol{y}|x)}[\log q(y|x;\theta)],$$
$$S(P,P)^x = \mathbb{E}_{y\sim P(\boldsymbol{y}|x)}[\log p(y|x)],$$
$$\mathbb{E}_{x\sim P_x(\boldsymbol{x})}[d(Q,P)^x] = \mathbb{E}_{x\sim P_x(\boldsymbol{x})}\left[\mathbb{E}_{y\sim P(\boldsymbol{y}|x)}[\log p(y|x) - \log q(y|x;\theta)]\right]$$

Thereby, the Logarithmic score guarantees that

$$\textit{CNN-CE}(P,Q,P_x) = 0 \iff Q(\boldsymbol{y}|x;\theta) = P(\boldsymbol{y}|x) \text{ a. e. on } \mathcal{X}. \quad (4.24)$$

4.7.2 Results

Quantitative evaluation. Table 4.5 reports the mean value of DISCO_Δ and Δ_{MEU} of the DISCO Nets and CNN-CE models. For each test face image, we draw 100 sets of ratings of the 20 attributes. We see that $\text{DISCO}_{\gamma=1/2}$ gives a slightly lower value of the divergence measure DISCO_Δ than CNN-CE, and outperforms CNN-CE on Δ_{MEU}. This supports our claim that using the same loss at training and testing gives better performance. As expected, the non-probabilistic counterpart $\text{DISCO}_{\gamma=0}$, and fails on DISCO_Δ. For a given input, the learning objective of $\text{DISCO}_{\gamma=0}$ encourages the prediction of a single output that minimises the loss with respect to the multiple ground-truth annotations. It does not capture the distribution of the ratings, for a given input. However, we note that there is room for improvement of $\text{DISCO}_{\gamma=1/2}$ compared to $\text{DISCO}_{\gamma=0}$ on the pointwise evaluation metric.

We do an experiment where at test time we sample from DISCO Nets the same way we sample from CNN-CE, that is, with the marginalised grade prediction process of equation (4.23). This gives a value of DISCO_Δ of 0.0368, to compare with the value of 0.0093 from Table 4.5 using the maximal graded prediction process. This asserts that DISCO Nets do not consider the values after the softmax activation function as probabilities of a factorised distribution.

However, the difference in performance are not statistically significant. Using Hoeffding inequality, given that the testing set consists of only 600 samples, one

Model	DISCO_Δ	Δ_{MEU}
$\text{DISCO}_{\gamma=1/2}$	**0.0093±0.000**	0.399±0.001
CNN-CE	0.0106±0.000	0.41±0.001
$\text{DISCO}_{\gamma=0}$	0.1196±0.001	**0.358±0.002**

Table 4.5: Evaluation metrics ± standard error of the mean on the testing set, computed using $K = 100$ sample outputs per test image. Best values are in bold.

can show that the true value of DISCO_{Δ} is in the interval $\text{DISCO}_{\Delta} \pm 0.066$ with probability 0.99. We leave for future work applications on a larger dataset, and pursue a qualitative analysis of the results.

Qualitative evaluation. In order to analyse the distributions captured by the models, we compute for each image (i) the attributes ratings mean and (ii) the attributes ratings covariance matrix of each model's distribution. As in the case of the correlation matrices in the hand pose estimation experiment, we estimate these quantities for each test input image and compute their mean on the testing data set. Each test image has 12 annotations, allowing us to estimate these statistics on the testing data set. For each model, we sample 12 ratings per input image, to match the number of annotations of the testing data set. Figure 4.9 shows the testing data set mean of the absolute error on the per-input attributes ratings mean. The performances of the CNN-CE and $\text{DISCO}_{\gamma=1/2}$ are on par. $\text{DISCO}_{\gamma=0}$ gives better results by returning a single output that is close to the mean of the ratings for that input.

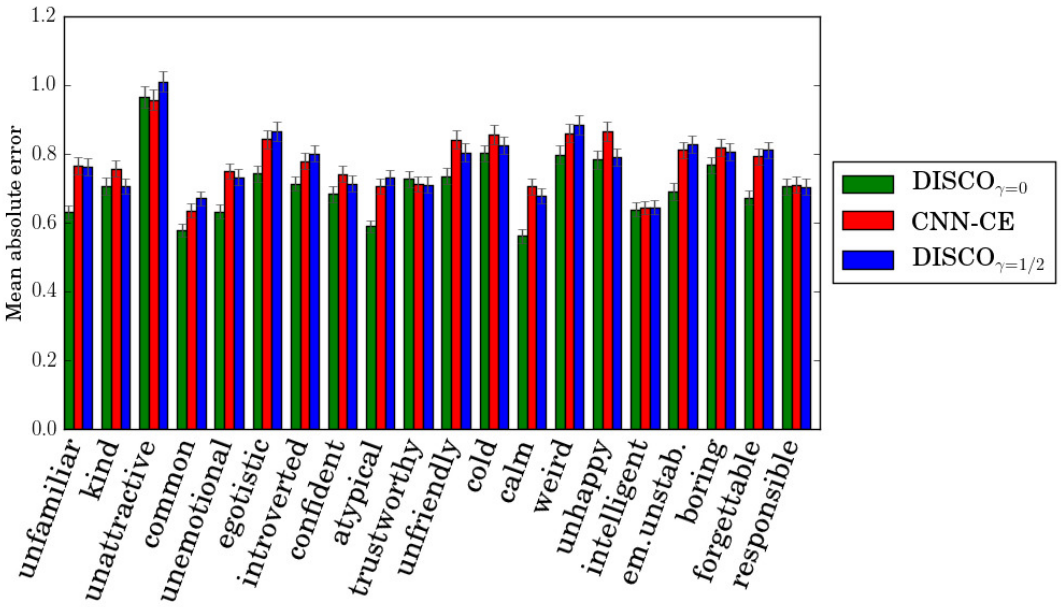
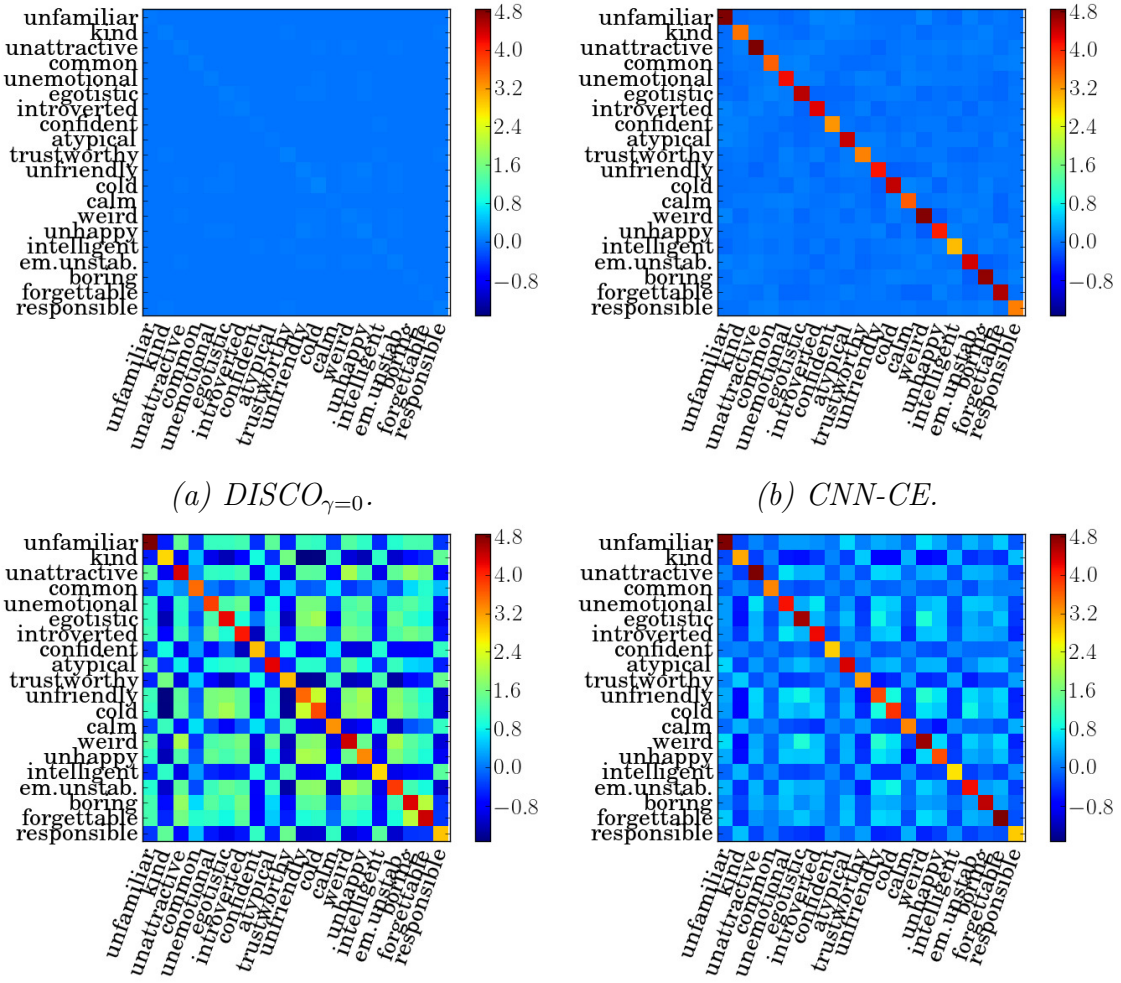


Figure 4.9: Mean and standard error of the mean of the absolute error on the per-input attribute ratings mean (em. unstab. stands for emotionally unstable).

Figure 4.10 shows the testing data set mean of the per-input covariance matrices. As we mentioned, $\text{DISCO}_{\gamma=0}$ predicts a single output per input, resulting in almost all 0 per-input variance on the attributes ratings in Figure 4.10a. Importantly, comparing Figures 4.10b, 4.10c and 4.10d we see that $\text{DISCO}_{\gamma=1/2}$ accurately captures covariances among attributes ratings that the CNN-CE model fails to capture[8]. However, this does not strongly impact the value of the metric DISCO_{Δ}.



(a) $DISCO_{\gamma=0}$. *(b) CNN-CE.*

(c) Test data set. *(d)* $DISCO_{\gamma=1/2}$.

Figure 4.10: Mean of the per-input attributes ratings covariance matrix (em. unstab. stands for emotionally unstable). Colorbars are all set to the test data set's colorbar.

[8] As the data is ordinal, we performed analysis using the Spearman rank coefficient and the conclusions are similar.

In appendix 7.5.4 we analyse the ability of the Negative kernel score to distinguish between the quality of two models' distribution, with the aim of understanding the limitations of our dissimilarity coefficient objective. We conclude that the Negative kernel score is more sensitive to an error in the estimation of the marginal means of a distribution than to the covariances.

4.8 Discussion

We proposed DISCO Nets, a new class of models to perform probabilistic structured output prediction that allows the user to tailor the model according to his/her needs. Using theory related to strictly proper scoring rules, we showed that for certain types of loss our objective function is guaranteed to be minimised if and only if the model's distribution perfectly represents the true data distribution. In the case of a continuous output, experimental results on the task of hand pose estimation consistently support our theoretical hypothesis as DISCO Nets outperform equivalent non-probabilistic models and existing probabilistic models. As their novelty resides in their objective function, DISCO Nets do not require any specific architecture and can be easily applied to new problems. In the case of a finite output space, we derived an approximation algorithm to learn the parameters of DISCO Nets. We performed experiments on a multi-class multi-label prediction task. By using a loss-calibrated training objective, based on a non-decomposable kernel, DISCO Nets accurately capture structure in the learned distribution.

In the discrete setting, our evaluation does not lead to a significant gain in quantitive performance, and we relate this to the discrimination ability of the Negative kernel score. Li et al. (2017) learn a parametrised kernel with adversarial training to increase the expressiveness of kernel Maximum Mean Discrepancy based models (Dziugaite et al., 2015; Li et al., 2015). Similarly, we contemplate parametrising

and learning the loss function employed in our objective to maximise the discrimination ability of the associated scoring rule. Moreover, in our experiment with discrete outputs, we used exact inference to solve the loss-augmented inference problems in order to perform a fair comparison. It would be interesting to tackle applications with large data sets and high-dimensional outputs, for example semantic segmentation, using existing approximation methods.

Chapter 5

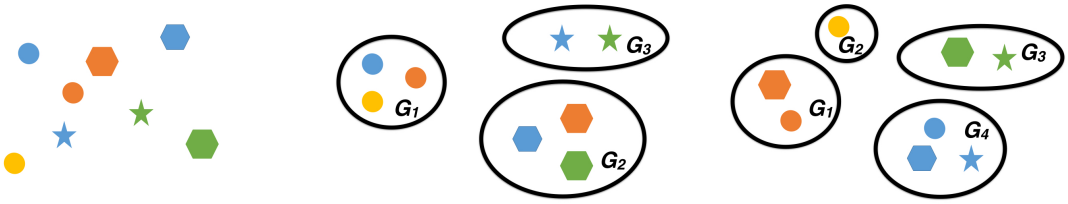
Multi-Level Variational Autoencoder for grouped data

In the previous two chapters, we considered that the task to perform was explicitly known, and that we had access to its associated loss function. In this chapter, we tackle the more general setting where the user needs are implicit. Specifically, we assume that the only input from the user is a particular grouping of the data, where within a group the observations (or samples) share a common factor of variation. In this context, our goal is to learn a representation that decomposes the data according to the semantics embedded in this specific grouping, and can be easily manipulated. This setting raises new challenges. First, the (possibly multiple) tasks at hand are not explicitly defined. Second, we want to independently control the factors of variation in the learned representation. Third, existing deep probabilistic models often assume that the observed variables are independent and identically distributed (iid), which is no longer true in the case of grouped data. Finally, the model should generalise to unseen realisations of the different factors of variation, while retaining efficient test-time inference over the latent representation.

We present the Multi-Level Variational Autoencoder (ML-VAE), a new deep probabilistic model for learning a disentangled representation of a set of grouped observations. The ML-VAE separates the latent representation into semantically meaningful parts by working both at the group level and the observation level. The main contributions of our method are (i) We propose the ML-VAE model to learn a disentangled and controllable latent representation of non-iid grouped data, while retaining the advantages of amortised inference; (ii) The proposed model handles an arbitrary number of groups at training, which needs not be the same at test-time, and generalises to unseen groups; (iii) The ML-VAE disentangles test data even if there is no grouping information, and leverages grouping when available; (iv) We perform qualitative and quantitative evaluations that show the relevance of our method.

5.1 Introduction

Representation learning refers to the task of learning a representation of the data that is useful for performing a variety of tasks (Bengio et al., 2013). Our goal is to build a model that disentangles the data into separate salient factors of variation and easily applies to a variety of tasks and different types of observation. Towards this goal there are multiple difficulties. First, the representative power of the learned representation depends on the information one wishes to extract from the data. Second, the multiple factors of variation impact the observations in a complex and correlated manner. Finally, we have access to very little, if any, supervision over these different factors. If there is no specific meaning to embed in the desired representation, the *infomax principle* (Linsker, 1988) states that an optimal representation is one of bounded entropy that retains as much information about the data as possible. By contrast, in our case there exists a semantically mean-



(a) Toy data set of objects of different shapes and colors.

(b) The toy data set, grouped by shape.

(c) The toy data set, grouped by color.

Figure 5.1: Illustration of group supervision: shape and color are two factors of variation.

ingful disentanglement of interesting latent factors. How can we anchor semantics in high-dimensional representations?

We propose *group level supervision*: observations are organised in groups, where within a group the observations share a common but unknown value for one of the factors of variation. For example, consider a data set of objects with two factors of variation: shape and color, as shown in Figure 5.1a. A possible grouping organises the objects by shape, as shown in Figure 5.1b. Another possible grouping of the same data set organises the objects by color as in Figure 5.1c. Grouped observations allow us to anchor the semantics of the data (shape and color) into the learned representation. Grouped observations are a form of weak supervision that is inexpensive to collect, and we do not assume we know the factor of variation that defines the grouping. We explain in section 2.3.3 that the Variational Autoencoder model approximates the posterior distribution of a latent variable (also referred to as latent representation, or latent code) given a set of observations. The VAE model uses amortised inference, that is, the observations parametrise the posterior of the latent variables, and all observations share a single set of parameters. This allows efficient test-time inference. However, the VAE model assumes that the observed variables are independent and identically distributed (iid). In the case of grouped observations, this assumption is no longer true. Let us consider again the toy ex-

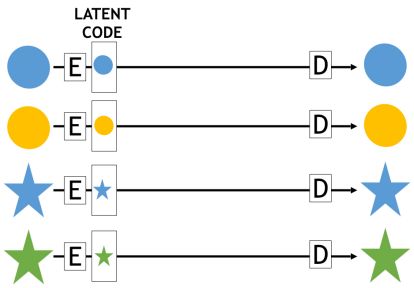
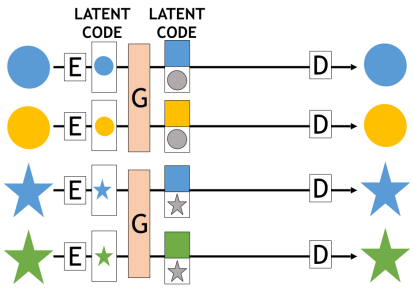


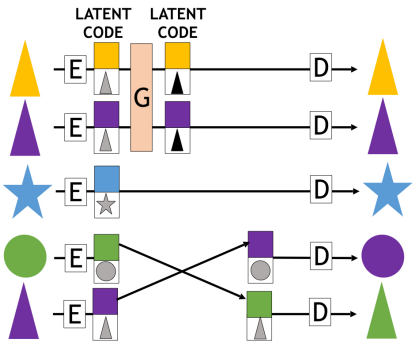
Figure 5.2: Illustration of the VAE on the toy data set, grouped by shape as in Figure 5.1b. Original VAE assumes iid observations and does not leverage grouping information. That is, the VAE model discards the knowledge that the circles belong to the same group, and the stars to a different group. E is the Encoder, D is the Decoder.

ample of the objects data set in Figure 5.1, and assume that the data is grouped by shape as in Figure 5.1b. The VAE model makes the iid assumption and processes each observation independently. This is shown in Figure 5.2. Therefore, the VAE model does not leverage group level information. How can we build a probabilistic model that easily incorporates this grouping information and learns the corresponding relevant representation, while retaining the advantages of amortised inference?

We present the Multi-Level Variational Autoencoder (ML-VAE), a new deep probabilistic model that learns a disentangled representation of a set of grouped observations. The ML-VAE separates the latent representation into semantically meaningful parts by working both at the group level and the observation level, and retains the benefits of amortised inference. Without loss of generality we assume that there are two latent factors of variation, *style* and *content*. The content is common for a group, while the style can differ within the group. We emphasise that our approach is general in that there can be more than two factors. Moreover, multiple groupings of the same data set, along different factors of variation, are possible. To process grouped observations, the ML-VAE uses a grouping operation that separates the latent code into two parts, style and content, and observations in the same group have the same content. This is illustrated in Figure 5.3a. For



(a) The grouping operation at training.



(b) ML-VAE at test-time.

Figure 5.3: Illustration of our ML-VAE on the toy data set, grouped by shape. Upper part of the latent code is color, lower part is shape. Black shapes show the ML-VAE accumulating evidence on the shape from the two grey shapes. E is the Encoder, D is the Decoder, G is the grouping operation.

illustrative purposes, the upper part of the latent code represents the style (color) and the lower part the content (shape). Recall that we consider the objects grouped by shape. In Figure 5.3a, after the grouping operation the two circles share the same shape in the lower part of the latent code (corresponding to content). The variations within the group (style), in this case color, get naturally encoded in the upper part. Importantly, the ML-VAE does not need to know that the objects are grouped by shape nor what shape and color represent; the only supervision at training is the organisation of the data into groups. The grouping operation makes the encoder learn a semantically meaningful disentanglement. Once trained the ML-VAE encoder is able to disentangle observations even without grouping information, for example the single blue star in Figure 5.3b. If samples are grouped the grouping operation increases the certainty on the content: in Figure 5.3b black triangles show that the model has accumulated evidence of the content (triangle) from the two disentangled codes (grey triangles). The ML-VAE generalises to unseen realisations of the factors of variation, for example a purple triangle, and we can manipulate the latent code to perform operations such as swapping the style to generate new observations, as shown in Figure 5.3b.

5.2 Representation learning with deep probabilistic models

Unsupervised and semi-supervised settings In the unsupervised setting, the Generative Adversarial Networks (GAN) and Variational Autoencoder (VAE) models, discussed in section 2.3, have been extended to the learning of an interpretable representation (Abbasnejad et al., 2016; Chen et al., 2016; Higgins et al., 2017; Wang and Gupta, 2016). As they are unsupervised, these models do not anchor a specific meaning into the disentanglement.

In the semi-supervised setting, the VAE model has been extended to the learning of a disentangled representation by introducing a semi-supervised variable, either discrete (Kingma et al., 2014) or continuous (Siddharth et al., 2017). Also in the semi-supervised context, Makhzani et al. (2015) and Mathieu et al. (2016) propose adversarially trained autoencoders to learn disentangled representations. However, semi-supervised models require the semi-supervised variable to be observed on a limited number of input points. The VAE model has also been applied to the learning of representations that are invariant to a certain source of variation (Alemi et al., 2017; Chen et al., 2017; Edwards and Storkey, 2016; Louizos et al., 2016). As in the semi-supervised case, these models require supervision on the source of variation to be invariant to. Consider the data set of objects, grouped by shape as in Figure 5.1b, and assume that the training set contains only 2 shapes: circle and star. Semi-supervised models using a discrete variable would have to fix its dimension, denoted K, for example taking $K = 2$ the number of training shapes. This does not allow to have an unbounded number of shapes and unseen shapes such as a triangle at test-time. Semi-supervised models with a continuous latent variable would choose an arbitrary fixed way to construct training labels from grouped data, for example per-shape statistics. At test-time, the unseen triangle shape would be

encoded as a mixture of the training shapes: circle and star.

By contrast, we address the setting in which training samples are grouped. A grouping is different from a label because test samples generally do not belong to any of the groups seen during training. Our approach is not semi-supervised, because each training example needs to belong to a group (may it be a singleton), but grouping is a relatively inexpensive form of supervision.

Interpretable representation of grouped data While not directly applied to interpretable representation learning, Murali et al. (2017) perform computer program synthesis from grouped user-supplied example programs, and Allamanis et al. (2017) learn semantic representations of mathematical and logical expressions grouped in equivalence classes. To perform 3D rendering of objects, Kulkarni et al. (2015) enforce a disentangled representation by using training batches where only one factor of variation varies. However, this requires to be able to fix each factor of variation. Multiple works perform image-to-image translation between two unpaired images sets using adversarial training (Bousmalis et al., 2017; Fu et al., 2017; Kim et al., 2017; Liu et al., 2017; Shrivastava et al., 2017; Taigman et al., 2017; Yi et al., 2017; Zhu et al., 2017). Two images sets can be seen as two groups of images, grouped by image type. Donahue et al. (2017) disentangles the latent space of GAN using images grouped by identity, and Denton and Birodkar (2017) and Tulyakov et al. (2017) learn disentangled representations of videos with adversarial training. A video can be seen as a group of images with common content (identity) and various styles (background). In contrast to these methods, we do not require adversarial networks. Moreover, it is unclear how to extend the cited models to other types of data, more than two groups, and several groupings (along multiple factors of variation) of the same data set.

We relate group supervision to the case of triplets annotations (Karaletsos et al.,

2016; Tian et al., 2017; Veit et al., 2017). A triplet is an ordering on three oberved data *a,b,c* of the form "*a* is more similar to *b* than *c*", which we see as a form of soft-grouping. Karaletsos et al. (2016) learn a latent representation jointly from observations and triplets. The distribution of the triplets is conditioned on the latent representation, which makes the model select parts of the latent code that are relevant for triplets prediction.

The *neural statistician* (Edwards and Storkey, 2017) computes representations of *datasets*, where samples in the same dataset share a common *context* latent variable. Their concept of dataset can be seen as a group, and the context latent variable would be the content. As in our model, the neural statistician uses an analytical distribution (for example a Normal distribution) for the common latent variable (context). Statistics of a dataset are fed to a network that outputs the parameters of this distribution. In our model, we explicitly build the content posterior distribution from the codes of the observations in the group, which are also written analytically, as detailed in section 5.3.2. While we chose an analytical distribution for the samples encodings, the neural statistician arbitrary chooses the statistics to compute. It would be interesting to perform an experimental comparison between the two approaches. Moreover, we want to learn a disentangled and controllable latent representation. Thereby, we model observations within a group to have a shared group content variable and an independent style variable, with style and content independent given the observation. By contrast, the neural statistician uses variables that capture the variations in a dataset, but these are conditioned on its context. We could also use the neural statistician to aggregate grouping information in our model, while keeping our independent style variable.

Our goal is to learn a disentangled and controllable latent representation of grouped data, while retaining the advantages of amortised inference. The model should manage an arbitrary number of groups, possibly multiple groupings along

different factors of variation, and unseen groups at test-time. In order to tackle the aforementioned deficiencies of existing methods, we propose the Multi-Level VAE (ML-VAE).

5.3 The Multi-Level VAE model

In the probabilistic framework, we assume that the observed variable $\boldsymbol{x}$ is generated by a latent variable $\boldsymbol{z}$ via the distribution $p(\boldsymbol{x}|\boldsymbol{z};\theta)$. The goal is to perform inference of the latent variable $\boldsymbol{z}$ given an observed value of $\boldsymbol{x}$, that is, to compute the posterior distribution $p(\boldsymbol{z}|\boldsymbol{x};\theta)$ which is often intractable. We consider a data set of n observations $\mathcal{D} = \{x_1, \ldots, x_n\}$.

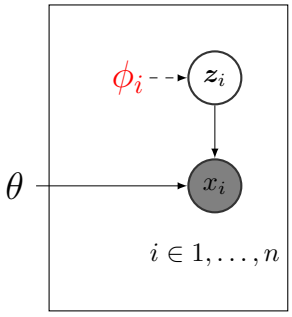
5.3.1 Amortised inference with the VAE model

Contrary to Stochastic Variational Inference (SVI) (Hoffman et al., 2013), we explain in section 2.3.3 that the VAE model allows efficient test-time inference by using amortised inference. The observations parametrise the posterior distribution of the latent code, and all observations share a single set of parameters ϕ. Figure 5.4 shows the SVI and VAE graphical models, where we highlight in red that SVI does not perform amortised inference.

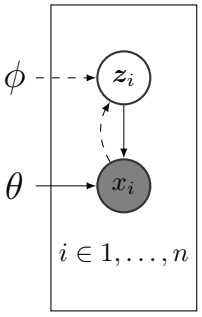
However, the VAE model assumes independent, identically distributed (iid) observed variables and maximises the average marginal log-likelihood (or evidence) lower bound decomposed on the observations $x_1, \ldots, x_n$, that is,

$$\mathcal{L}(\mathcal{D};\phi,\theta) := \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{z_i\sim q(\boldsymbol{z}_i|x_i;\phi)}[\log p(x_i|z_i;\theta)] - \mathrm{KL}(q(\boldsymbol{z}_i|x_i;\phi)||p(\boldsymbol{z}_i)).$$

where KL is the Kullback-Leibler divergence. Therefore, the VAE model does not leverage grouping information. The question is how to perform amortised inference in the context of non-iid, grouped observations?



(a) SVI graphical model for iid observations.



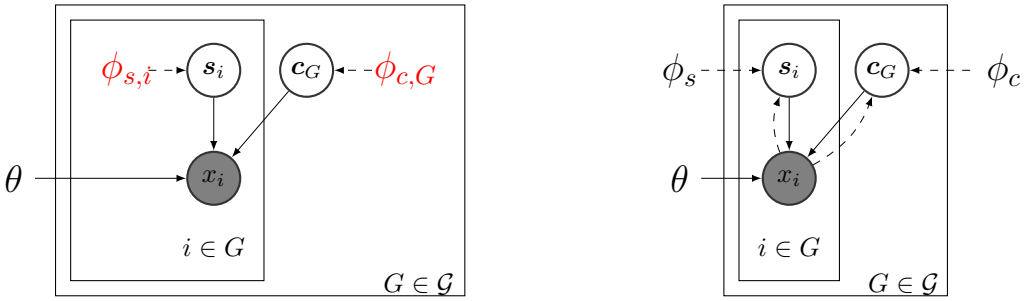
(b) VAE graphical model for iid observations.

Figure 5.4: VAE and SVI graphical models. Solid lines denote the generative model, dashed lines denote the variational approximation. Shaded nodes indicate that the observed variables $\boldsymbol{x}_i$ have been set to their observed value.

5.3.2 The Multi-Level VAE for grouped data

In the grouped data setting, the observations are organised in a set $\mathcal{G}$ of distinct groups, with a factor of variation that is shared among all observations within a group. The grouping forms a partition of $1, \ldots, n$, that is, each group $G \in \mathcal{G}$ is a subset of $1, \ldots, n$ of arbitary size, disjoint of all other groups. Without loss of generality, we separate the latent representation in two latent variables $\boldsymbol{z} = (\boldsymbol{c}, \boldsymbol{s})$ with *style* $\boldsymbol{s}$ and *content* $\boldsymbol{c}$. The content is the factor of variation along which the groups are formed. In this context, referred to as the grouped observations setting, the latent representation has a single content latent variable $\boldsymbol{c}_G$ per group. SVI can easily be adapted by enforcing that all observations within a group share a single content latent variable while the style remains untied, see Figure 5.5a. However, SVI does not use amortised inference and requires expensive test-time inference. Experimentally, it also requires more passes on the training data as we show in appendix 7.6.4.

We denote by $\boldsymbol{X}_G = (\boldsymbol{x}_i, \forall i \in G)$ the collection of observed variables of a group G. We do not assume iid observations, but make the iid assumption at the grouped observations level. The average marginal log-likelihood of the data set



(a) SVI graphical model for non-iid, grouped observations.

(b) Our ML-VAE graphical model for non-iid, grouped observations.

Figure 5.5: SVI and our ML-VAE graphical models. Solid lines denote the generative model, dashed lines denote the variational approximation. Shaded nodes indicate that the observed variables $\boldsymbol{x}_i$ have been set to their observed value.

decomposes over groups of observations, as follows,

$$\frac{1}{|\mathcal{G}|}\log p(\mathcal{D};\theta) = \frac{1}{|\mathcal{G}|}\sum_{G\in\mathcal{G}}\log p(X_G;\theta).$$

We model each $\boldsymbol{x}_i \in \boldsymbol{X}_G$ to have its independent latent representation for the style $\boldsymbol{s}_i$, and $\boldsymbol{S}_G = (\boldsymbol{s}_i, i \in G)$ is the collection of style latent variables for the group G. By constrast, we model a single content latent representation $\boldsymbol{c}_G$ shared among all $\boldsymbol{x}_i \in \boldsymbol{X}_G$. We make the following assumptions.

1. We approximate the true posterior $p(\boldsymbol{c}_G, \boldsymbol{S}_G|X_G;\theta)$ with a variational posterior $q(\boldsymbol{c}_G, \boldsymbol{S}_G|X_G;\phi)$ that decomposes as the product of $q(\boldsymbol{c}_G|X_G;\phi_c)$ and $q(\boldsymbol{S}_G|X_G;\phi_s)$, where ϕ_c and ϕ_s are the variational parameters for content and style respectively;

2. The style latent variables for a group are assumed independent, that is, the prior on the group style latent variables $p(\boldsymbol{S}_G)$ decomposes as the product of identical priors $p(\boldsymbol{s}_i), \forall i \in G$. Similarly, the posterior $q(\boldsymbol{S}_G|X_G;\phi_s)$ decomposes as the product of $q(\boldsymbol{s}_i|x_i;\phi_s), \forall i \in G$;

3. The observed variables in a group are independent given the style and content latent variables, and $p(\boldsymbol{X}_G|c_G, S_G;\theta)$ also decomposes as the product

of $p(\boldsymbol{x}_i|c_G, s_i; \theta), \forall i \in G$.

This results in the graphical model shown Figure 5.5b. For each group of observations $X_G = (x_i, \forall i \in G)$, we decompose its evidence as follows,

$$\log p(X_G; \theta) = \mathcal{L}(X_G; \theta, \phi_c, \phi_s) + \mathrm{KL}(q(\boldsymbol{c}_G, \boldsymbol{S}_G|X_G; \phi_c, \phi_s)||p(\boldsymbol{c}_G, \boldsymbol{S}_G|X_G; \theta)),$$

where we denote the Group Evidence Lower Bound (Group ELBO) $\mathcal{L}(X_G; \theta, \phi_c, \phi_s)$. Since $\mathrm{KL}(q(\boldsymbol{c}_G, \boldsymbol{S}_G|X_G; \phi_c, \phi_s)||p(\boldsymbol{c}_G, \boldsymbol{S}_G|X_G; \theta))$ is always positive, $\mathcal{L}(X_G; \theta, \phi_c, \phi_s)$ is a lower bound on the evidence, that is,

$$\log p(X_G; \theta) \geq \mathcal{L}(X_G; \theta, \phi_c, \phi_s).$$

Given our assumptions, the Group ELBO can be rewritten as follows,

$$\begin{aligned}
\mathcal{L}(X_G; \theta, \phi_c, \phi_s) &= \mathbb{E}_{(c_G, S_G) \sim q(\boldsymbol{c}_G, \boldsymbol{S}_G|X_G; \phi_c, \phi_s)}[\log p(X_G|c_G, S_G; \theta)] \\
&\quad - \mathrm{KL}(q(\boldsymbol{c}_G, \boldsymbol{S}_G|X_G; \phi_c, \phi_s)||p(\boldsymbol{c}_G, \boldsymbol{S}_G)) \\
&= \sum_{i \in G} \mathbb{E}_{c_G \sim q(\boldsymbol{c}_G|X_G; \phi_c)} \left[\mathbb{E}_{s_i \sim q(\boldsymbol{s}_i|x_i; \phi_s)}[\log p(x_i|c_G, s_i; \theta)]\right] \\
&\quad - \sum_{i \in G} \mathrm{KL}(q(\boldsymbol{s}_i|x_i; \phi_s)||p(\boldsymbol{s}_i)) - \mathrm{KL}(q(\boldsymbol{c}_G|X_G; \phi_c)||p(\boldsymbol{c}_G)).
\end{aligned}$$

We maximise the average Group ELBO of the data set $\mathcal{D}$, defined as,

$$\mathcal{L}(\mathcal{D}, \phi_c, \phi_s, \theta) := \frac{1}{|\mathcal{G}|} \sum_{G \in \mathcal{G}} \mathcal{L}(X_G; \theta, \phi_c, \phi_s). \tag{5.1}$$

It is a lower bound on the average marginal log-likelihood of the dataset since each Group ELBO $\mathcal{L}(X_G; \theta, \phi_c, \phi_s)$ is a lower bound on $\log p(X_G; \theta)$, as shown below.

$$\frac{1}{|\mathcal{G}|} \log p(\mathcal{D}; \theta) = \frac{1}{|\mathcal{G}|} \sum_{G \in \mathcal{G}} \log p(X_G; \theta) \geq \frac{1}{|\mathcal{G}|} \sum_{G \in \mathcal{G}} \mathcal{L}(X_G; \theta, \phi_c, \phi_s) = \mathcal{L}(\mathcal{D}; \theta, \phi_c, \phi_s).$$

Algorithm 4: ML-VAE training algorithm.

1 **for** *t=1,...,T epochs* **do**
2 Sample mini-batch of groups $\mathcal{G}_b$.
3 **for** $G \in \mathcal{G}_b$ **do**
4 **for** $i \in G$ **do**
5 Encode x_i into $q(\boldsymbol{c}_G|x_i;\phi_c^t)$,$q(\boldsymbol{s}_i|x_i;\phi_s^t)$.
6 **end**
7 Construct $q(\boldsymbol{c}_G|X_G;\phi_c^t)$ using $q(\boldsymbol{c}_G|x_i;\phi_c^t), \forall i \in G$.
8 **for** $i \in G$ **do**
9 Sample $c_{G,i} \sim q(\boldsymbol{c}_G|X_G;\phi_c^t)$, $s_i \sim q(\boldsymbol{s}_i|x_i;\phi_s^t)$.
10 Decode $c_{G,i}, s_i$ into $p(\boldsymbol{x}_i|c_{G,i}, s_i;\theta^t)$.
11 **end**
12 **end**
13 Update parameters $\theta^{t+1}, \phi_c^{t+1}, \phi_s^{t+1} \leftarrow \theta^t, \phi_c^t, \phi_s^t$ by ascending the gradient estimate $\widehat{\nabla}_{\theta,\phi_c,\phi_s}\mathcal{L}(B,\theta,\phi_c,\phi_s)|_{\theta^t,\phi_c^t,\phi_s^t}$.
14 **end**

In practise, we estimate the average Group ELBO of the data set using mini-batches of groups, as follows,

$$\hat{\mathcal{L}}(B,\theta,\phi_c,\phi_s) := \frac{1}{|\mathcal{G}_b|}\sum_{G\in\mathcal{G}_b}\mathcal{L}(X_G;\theta,\phi_c,\phi_s). \tag{5.2}$$

where $\mathcal{G}_b \subseteq \mathcal{G}$ is a mini-batch of groups. If we take each group $G \in \mathcal{G}_b$ in its entirety it is an unbiased estimator of the average Group ELBO of the data set. When the groups sizes are too large, because of limited memory capacity we subsample G, resulting in a bias discussed in appendix 7.6.3. Our training algorithm is shown in Algorithm 4. As in the previous chapter, we use the Law of the Unconscious Statistician (LOTUS), discussed in section 2.3.1, to obtain an estimate of the gradient of our objective function. Note that in step 9 of Algorithm 4 we sample one content $c_{G,i}$ per observation in the group, but c_G can be sampled once and used for all the samples in a group. We experimentally tried this method which resulted in similar performances. We attribute this to the fact that the variances of the content distribution tend to be very small.

5.3.3 Accumulating group evidence with the product of Normal densities method

For each group G, in step 7 of Algorithm 4, we build the group content posterior distribution by accumulating information from the result of encoding each sample in G. How can we accumulate the information in a relevant manner to compute the group content distribution?

Our idea is to explicitly build the group content posterior distribution $q(\boldsymbol{c}_G|X_G;\phi_c)$ from the encoding of the grouped observations $X_G = (x_i, \forall i \in G)$. While any distribution could be employed, we focus on using a product of Normal density functions, which can be seen as an instance of Structured Variational Autoencoder (SVAE) (Johnson et al., 2016). Other possibilities, such as a mixture of Normal density functions, are discussed in appendix 7.6.4. Specifically, we construct the probability density function of the posterior of the content variable $\boldsymbol{c}_G$ by multiplying $|G|$ Normal density functions, each of them evaluating the probability of the variable $\boldsymbol{c}_G$ taking the value c_G, given the observation $\boldsymbol{x}_i = x_i, \forall i \in G$.

$$q(\boldsymbol{c}_G = c_G|\boldsymbol{X}_G = X_G;\phi_c) \propto \prod_{i\in G} q(\boldsymbol{c}_G = c_G|\boldsymbol{x}_i = x_i;\phi_c),$$

where assume $q(\boldsymbol{c}_G|\boldsymbol{x}_i = x_i;\phi_c)$ to be a Normal distribution $\mathcal{N}(\mu_i, \Sigma_i)$. The normalisation constant is the resulting product marginalised over all possible values of $\boldsymbol{c}_G$. Murphy (2007) show that the product of two Gaussians is a Gaussian. Similarly, in appendix 7.6.1 we show that the density function $q(\boldsymbol{c}_G|X_G;\phi_c)$ is the density

function of a Normal distribution of mean μ_G and variance Σ_G, expressed as follows,

$$\Sigma_G^{-1} = \sum_{i \in G} \Sigma_i^{-1},$$
$$\mu_G^T \Sigma_G^{-1} = \sum_{i \in G} \mu_i^T \Sigma_i^{-1}.$$

It is interesting to note that the variance of the resulting Normal distribution, Σ_G, is inversely proportional to the sum of the group's observations inverse variances $\sum_{i \in G} \Sigma_i^{-1}$. Therefore, we expect that by increasing the number of observations in a group, the variance of the resulting distribution decreases. This is what we refer to as "accumulating evidence". We empirically investigate this effect in section 5.4. Since the resulting distribution is a Normal distribution, the term $\mathrm{KL}(q(\boldsymbol{c}_G|X_G;\phi_c)||p(\boldsymbol{c}_G))$ can be evaluated in closed-form. We also assume a Normal distribution for $q(\boldsymbol{s}_i|x_i;\phi_s), i \in G$.

5.4 Digit and face images disentanglement experiment

5.4.1 Setting

In our experiments we use the product of Normal densities method presented in section 5.3.3 to build the ML-VAE content code. Our goal with the experiments is two-fold. First, we want to evaluate the performance of ML-VAE to learn a semantically meaningful disentangled representation. Second, we want to explore the impact of "accumulating evidence" at test-time. To do so, at test-time when we encode test images we employ two possible strategies, as follows,

- Strategy 1 consists of disregarding the grouping information of the test samples, that is, each test image is a group.

- Strategy 2 consists of taking into account the grouping information of the test samples, that is, taking multiple test images per group to construct the content latent representation with the product of Normal densities method.

Similar to Mathieu et al. (2016), we propose qualitative and quantitative evaluations. In our quantitative evaluation, we compare with the original VAE to compare with a variational model that does not leverage grouping information at training. However we do not show qualitative results of the original VAE model as there is no objective choice on which part of its code is style or content. Details on our experimental setting, as well as additional results are in the appendix 7.6.4.

MNIST data set. We perform evaluation on MNIST (LeCun et al., 1998). We consider the data grouped by digit. We randomly separate the $60,000$ training examples into $50,000$ training samples and $10,000$ validation samples, and use the standard MNIST testing data set.

MS-Celeb-1M data set. Next, we perform evaluation on the face aligned version of the MS-Celeb-1M data set (Guo et al., 2016). The data set was constructed by retrieving approximately 100 images per celebrity from popular search engines, and noise has not been removed from the data set. For each query, we consider the top ten results (there were multiple queries per celebrity, therefore identities can have more than 10 images). This creates a data set of $98,880$ identities for a total of $811,792$ images. We group the data by identity. Importantly, we randomly separate the data set in disjoints sets of identities as the training, validation and testing data sets. This way we evaluate the ability of ML-VAE level to generalise to unseen groups (unseen identities) at test-time.

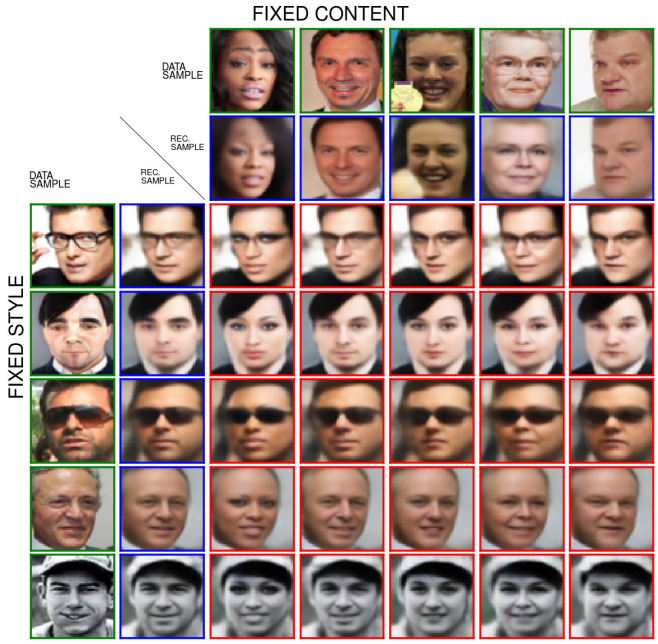
The training data set consists of $48,880$ identities (total $401,406$ images), the validation data set consists of $25,000$ identities (total $205,015$ images) and the

testing data set consists of $25,000$ identities (total $205,371$ images). We resize the images to 64×64 pixels to fit the encoder network architecture.

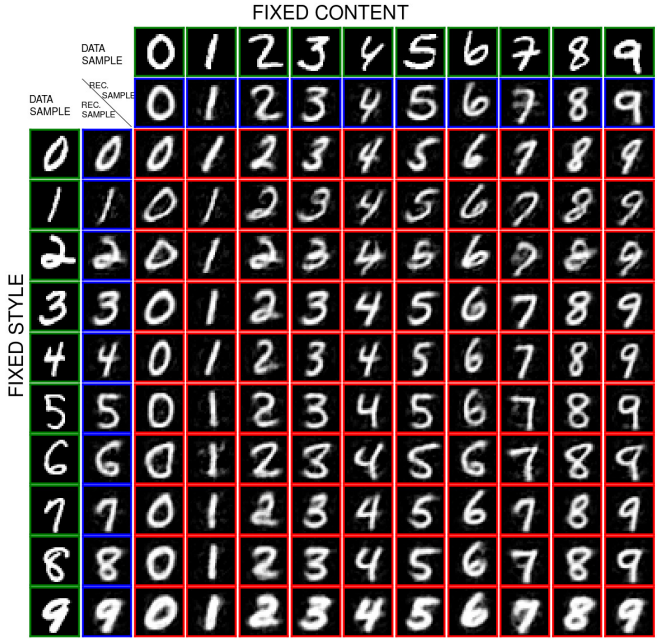
5.4.2 Results

Qualitative Evaluation. We qualitatively assess the relevance of the learned representation by performing operations on the latent space. First we perform *swapping*. We encode test images, draw a sample per image from its style and content latent representations, and swap the style between images. Figure 5.6 shows the swapping results with strategy 2, with accumulating evidence of up to 10 images that belong to the same group. The first row and the first column show the test data sample input to ML-VAE (green boxes), the second row and column are reconstructed samples (blue boxes). In the remaining rows and columns, each row is a fixed style and each column is a fixed content (red boxes). Looking at each column in Figure 5.6a, we see that the model encodes the factor of variation that grouped the data, that is the identity, into the facial traits. Indeed, the facial traits remain consistent along each column, while the style varies. The model encodes the remaining factors (for example background color, face orientation, accessories) into the style latent representation. This shows that the ML-VAE learns a disentangled and controllable representation of the data, which anchors the semantics of the grouping. The model learns this meaningful disentanglement without knowing that the data is grouped by identity but only knowing the organisation of the data into groups. Similarly, Figure 5.6b shows that the ML-VAE encodes the digit label into the content. Moreover, we see that the ML-VAE generalises to unseen groups, as for MS-Celeb-1M training and testing identities are disjoints.

For comparison, we present the results of swapping without accumulating in Figure 5.7. This shows that the ML-VAE learns a disentangled representation even without grouping information at test-time.

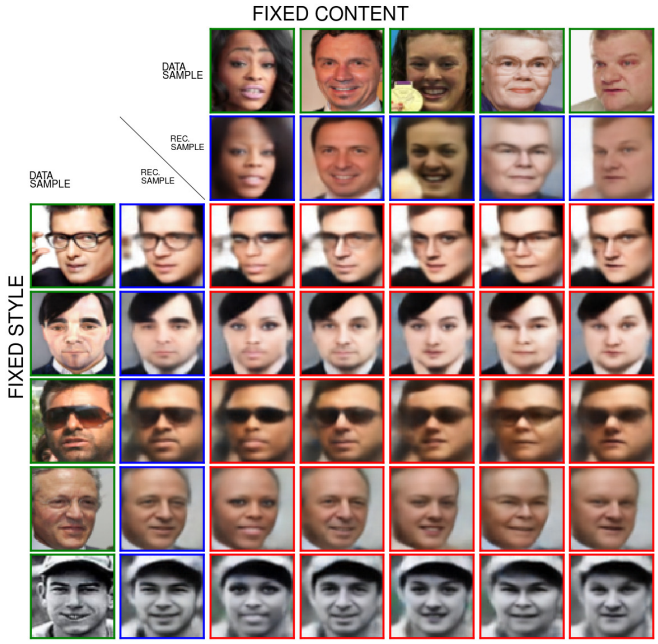


(a) Swapping on MS-Celeb-1M testing data set with strategy 2.

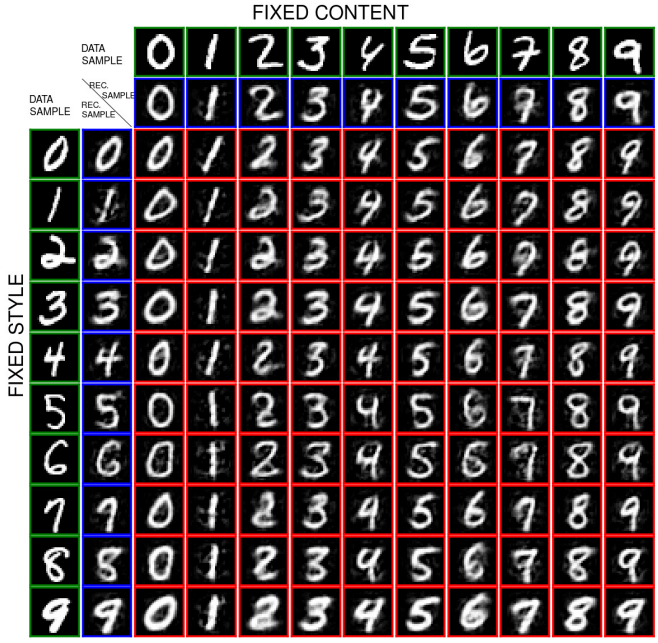


(b) Swapping on MNIST testing data set with strategy 2.

Figure 5.6: Swapping with strategy 2. The first row and the first column show the test data sample input (green boxes), the second row and column are reconstructed samples (blue boxes). In the remaining rows and columns, each row is a fixed style and each column is a fixed content (red boxes).



(a) Swapping on MS-Celeb-1M testing data set with strategy 1.



(b) Swapping on MNIST testing data set with strategy 1.

Figure 5.7: Swapping with strategy 1. *The first row and the first column show the test data sample input (green boxes), the second row and column are reconstructed samples (blue boxes). In the remaining rows and columns, each row is a fixed style and each column is a fixed content (red boxes).*

Second we perform *interpolation*. We encode a pair of test images, draw one sample from each image style and content latent representations, and linearly interpolate between the style and content samples. Figure 5.8 shows the results of the interpolation task. From top left to bottom right, rows correspond to a fixed style and interpolating on the content, columns correspond to a fixed content and interpolating on the style. We see that the identity, in the form of facial traits, remains consistent along the column, while we linearly interpolate the style. If we look along each line, the style remain consistent and the identity smoothly varies as we interpolate on the content.

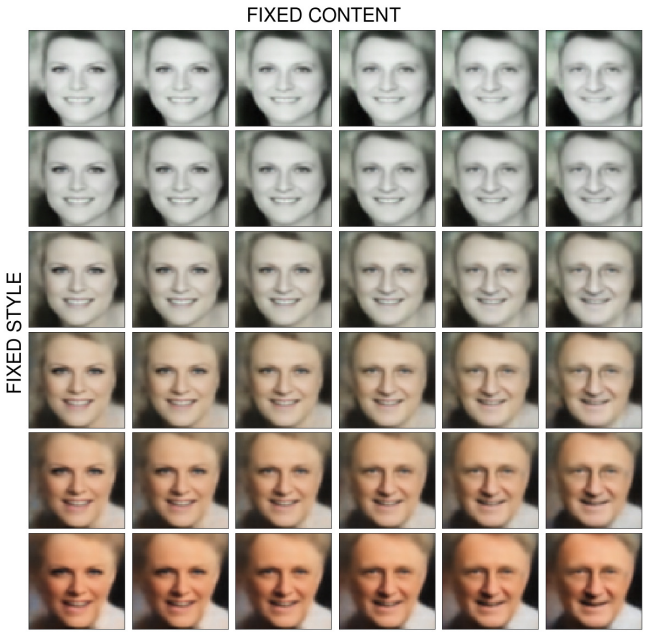


Figure 5.8: Interpolation, from top left to bottom right, rows correspond to a fixed style and interpolating on the content, columns correspond to a fixed content and interpolating on the style.

We present the results of interpolation with accumulating evidence of up to 10 images (strategy 2). Results without accumulated evidence (strategy 1) are also satisfactory and available in appendix 7.6.4. Third, we perform *generation*. For a given test identity, we build the content latent representation by accumulating images of this identity. We take the mean of the resulting content distribution and generate

(a) Generation. *(b) Accumulating evidence.*

Figure 5.9: In (a): generation. Style is sampled from the random prior and the content distribution is computed using the test data images for this identity (green boxes on the left). In (b): accumulating evidence. Left column are test data samples, middle column are reconstructed samples without accumulating evidence (w/o acc. ev.), right column are reconstructed samples with accumulating evidence (w/ acc. ev.), using the four digits images to build the content code.

images with different style drawn from the prior distribution. Figure 5.9a shows the results. We see that the facial traits remain consistent in the generated images, while sampling the style from the prior results in different head orientation, lighting conditions, presence/absence of moustache, etc. These results highlight that our model covers the data manifold, and emphasise on the disentanglement power of the model. Finally, in Figure 5.9b, we reconstruct digits of the same label with and without taking into account the grouping information (strategies 1 and 2). We see that in the first row and second column, the ML-VAE infers a 0 with strategy 1, but corrects inference in the third column by accumulating evidence with strategy 2.

Qualitative evaluation showed that the ML-VAE learns a representation that disentangles according the semantics of the grouping and can be manipulated. This

would allow a user to employ the model for a variety of tasks. For example in the case of the MS-Celeb-1M data set, we expect the content latent representation to be highly informative about the identity of the person, and could be used to perform identity classification. By contrast, the style should remain uninformative. We test this assumption with a quantitative evaluation.

Quantitative Evaluation. We use the style latent representation and content latent representation as features for a classification task. We denote by $\boldsymbol{y}$ the random variable representing the class, and by G_y a group of observations from the same class. The quality of the disentanglement is high if the content latent variable $\boldsymbol{c}_{G_y}$ is informative about the class, while the style latent variable $\boldsymbol{S}_{G_y}$ is not. In the case of MNIST the class is the digit label and for MS-Celeb-1M the class is the identity. We emphasise that in the case of MS-Celeb-1M test images are all unseen classes (unseen identities) at training. We learn to classify the test images with a neural network classifier once using $\boldsymbol{S}_{G_y}$ and once using $\boldsymbol{c}_{G_y}$ as input features. We also use the original VAE model full latent code as features. We also accumulate evidence with the product of Normal densities method for samples of the same class to construct the features from the original VAE latent code.

Let us take the example of the latent representation $\boldsymbol{c}_{G_y}$ used as features. We train the neural network classifier to learn a distribution $r(\boldsymbol{y}|\boldsymbol{c}_{G_y})$ by minimising the cross-entropy loss $-\mathbb{E}_{p(\boldsymbol{y},\boldsymbol{c}_{G_y})}\big[\log r(\boldsymbol{y}|\boldsymbol{c}_{G_y})\big]$, where $p(\boldsymbol{y}, \boldsymbol{c}_{G_y})$ is the joint distribution of the class and content latent representation, specific to each trained model (VAE or ML-VAE). We estimate this quantity by sampling from $p(\boldsymbol{y}, \boldsymbol{c}_{G_y})$, as follows,

1. Sample a class $y \sim p(\boldsymbol{y})$.
2. Sample grouped observations for this class $X_{G_y} \sim p(\boldsymbol{X}_{G_y})$.
3. Sample the latent representation to use as features, in this example the content, $c_{G_y} \sim q(\boldsymbol{c}_{G_y}|X_{G_y}; \phi_c)$.

We can upper bound $\mathbb{H}(\boldsymbol{y}|\boldsymbol{c}_{G_y})$, the conditional entropy of the class given the latent representation as follows,

$$\begin{aligned}
\mathbb{H}(\boldsymbol{y}|\boldsymbol{c}_{G_y}) &= -\mathbb{E}_{p(\boldsymbol{y},\boldsymbol{c}_{G_y})}\big[\log p(\boldsymbol{y}|\boldsymbol{c}_{G_y})\big] \\
&= -\mathbb{E}_{p(\boldsymbol{y},\boldsymbol{c}_{G_y})}\left[\log \frac{p(\boldsymbol{y}|\boldsymbol{c}_{G_y})}{r(\boldsymbol{y}|\boldsymbol{c}_{G_y})} r(\boldsymbol{y}|\boldsymbol{c}_{G_y})\right] \\
&= -\mathbb{E}_{p(\boldsymbol{y},\boldsymbol{c}_{G_y})}\big[\log r(\boldsymbol{y}|\boldsymbol{c}_{G_y})\big] - \mathbb{E}_{p(\boldsymbol{y},\boldsymbol{c}_{G_y})}\left[\log \frac{p(\boldsymbol{y}|\boldsymbol{c}_{G_y})}{r(\boldsymbol{y}|\boldsymbol{c}_{G_y})}\right] \\
&= -\mathbb{E}_{p(\boldsymbol{y},\boldsymbol{c}_{G_y})}\big[\log r(\boldsymbol{y}|\boldsymbol{c}_{G_y})\big] - \mathbb{E}_{p(\boldsymbol{y},\boldsymbol{c}_{G_y})}\left[\log \frac{p(\boldsymbol{y},\boldsymbol{c}_{G_y})}{r(\boldsymbol{y},\boldsymbol{c}_{G_y})}\right] \\
&= -\mathbb{E}_{p(\boldsymbol{y},\boldsymbol{c}_{G_y})}\big[\log r(\boldsymbol{y}|\boldsymbol{c}_{G_y})\big] - \mathrm{KL}(p(\boldsymbol{y},\boldsymbol{c}_{G_y})||r(\boldsymbol{y},\boldsymbol{c}_{G_y})) \\
&\leq -\mathbb{E}_{p(\boldsymbol{y},\boldsymbol{c}_{G_y})}\big[\log r(\boldsymbol{y}|\boldsymbol{c}_{G_y})\big]\,,
\end{aligned}$$

since the Kullback-Leibler divergence is always positive. By training the neural network classifier to minimise the cross-entropy loss, we minimise an upper bound on the conditional entropy of the class given the latent representation. We report the classifier test accuracy, and the value of $-\mathbb{E}_{p(\boldsymbol{y},\boldsymbol{c}_{G_y})}\big[\log r(\boldsymbol{y}|\boldsymbol{c}_{G_y})\big]$ on the classifier testing set as the conditional entropy in bits. Similarly, we report the performances with the ML-VAE style latent representation, and the original VAE model full latent representation.

We explore the benefits of accumulating evidence: (i) for training the classifier, we construct the variational approximation of the content by accumulating K images per class (ii) for testing the classifier, we use only $k \leq K$ images per class, where $k = 1$ corresponds to no grouping information. When k increases we expect the performance of the classifier trained on $\boldsymbol{c}_{G_y}$ to improve as the features become more informative. We expect the performance using features $\boldsymbol{S}_{G_y}$ to remain constant.

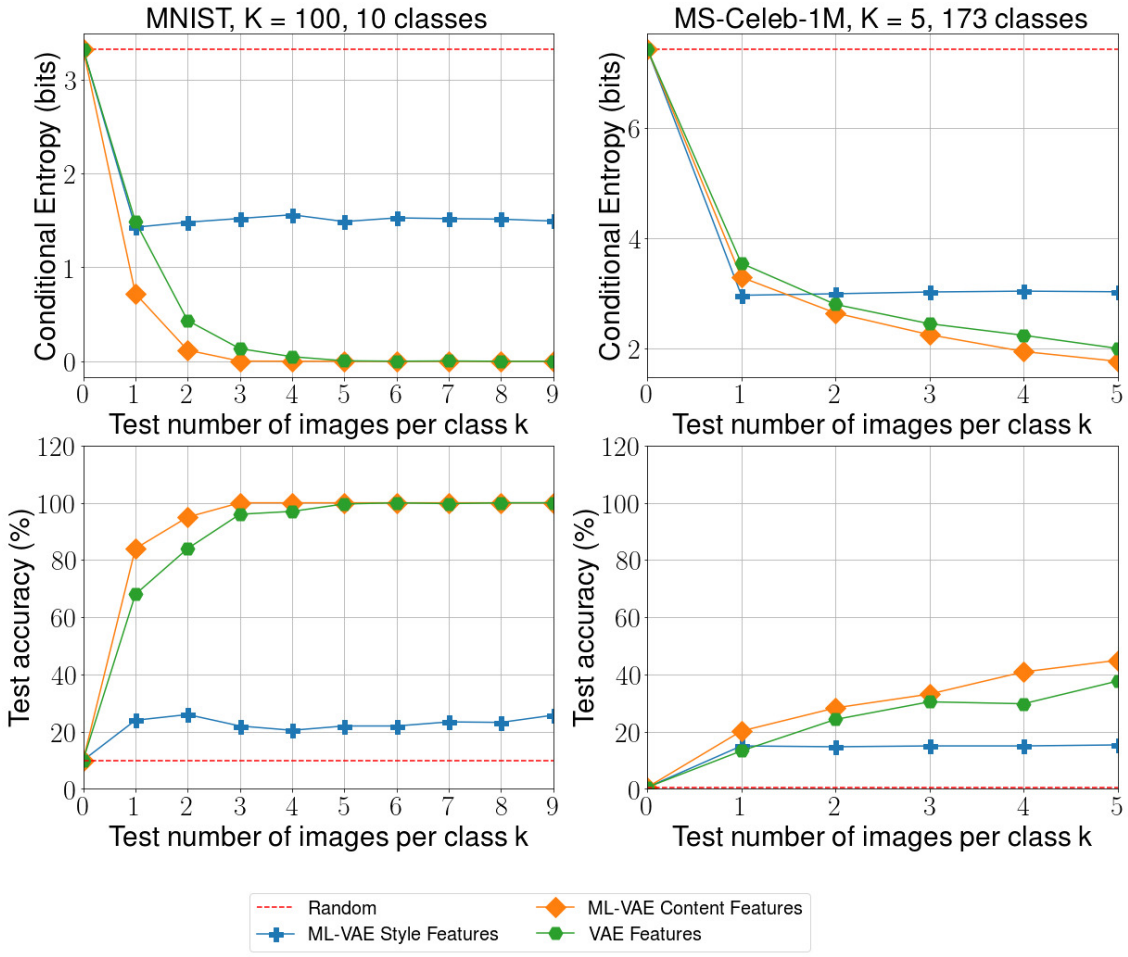


Figure 5.10: Quantitative evaluation of the ML-VAE. Accuracy (higher is better) and conditional entropy (lower is better). For clarity on MNIST we show up to $k = 10$. Values stay stationary for larger k (see appendix 7.6.4).

The results are shown in Figure 5.10. First, we see that for small values of k, the ML-VAE content latent representation is more informative about the class than VAE latent representation, especially on MNIST. Second, when k increases this shows the benefit of accumulating evidence. Recall that we also accumulate evidence for the original VAE model full latent code. The ML-VAE also provides relevant disentanglement as the style remains uninformative about the class.

5.5 Discussion

We proposed the Multi-Level Variational Autoencoder (ML-VAE) model for learning a latent representation that disentangles according to a particular grouping of the data, while retaining the advantages of amortised inference. The ML-VAE handles an arbitrary number of groups of observations, which needs not be the same at training and testing. We presented different methods for incorporating group level supervision, and showed that the product of Normal densities method allows the model to accumulate evidence. Quantitative and qualitative evaluation on digits and face images show that the ML-VAE (i) learns a semantically meaningful disentanglement, (ii) enables control on the latent representation, (iii) disentangles without grouping information at test-time, and leverages grouping information when available, and (iv) generalises to unseen groups.

An interesting direction for future research would be to apply the proposed model to discrete data. For example, we could easily collect groups of texts in a specific language. As the ML-VAE provides control over the learned representation, using the swapping operation we might be able to perform tasks such as text-to-text translation. We would train a ML-VAE on text data in different languages, grouped by language, and the content latent variable would be the language. At test-time, we would encode a two texts in two different languages, and by exchanging their content latent code we hope to generate two new texts that are translations of each text in the other text's language.

In our experiments we focused on the product of Normal densities method, and explained that other methods are possible. For example, to embed a specific structure in the learned representation, we could use more complex instances of the Structured VAE (Johnson et al., 2016).

Finally, Figure 5.7 showed that the ML-VAE is able to transfer the style while

preserving the content, even without grouping information at test-time. However, we note that in Figure 5.10, when $k = 1$ the style and content performances are similar on MS-Celeb-1M. We attribute this to the fact that there is information in the style that allows the classifier to correctly label two images of the same identity. That is, in the two images used to train and test the classifier, the person has the same hair color, or wears eyeglasses. While we do not pursue further exploration, this raises an interesting question. We defined identity in the form of the facial traits, but should eyeglasses be defined as identity, while sunglasses should not? Would two groupings collected from two different users along the same factor of variation be exactly similar? We will come back to this question in our final chapter.

Chapter 6

Discussion and future work

Machine learning provide powerful methods to process and predict real-world data. Such data is often represented as multi-dimensional structured objects, which calls for the use of flexible and expressive models. Moreover, in many cases there is strong uncertainty in the object to predict. We classify machine learning models into two types, (i) non-probabilistic models that return a single pointwise prediction and (ii) probabilistic models that capture a distribution. In order to obtain the best performance, non-probabilistic training objectives have been designed to take into account the loss associated to the task at hand. However, non-probabilistic models do not capture uncertainty in the structure to predict. By contrast, probabilistic models are a natural choice to capture structured probability distributions. However, existing probabilistic models employ training objectives that are often agnostic of the evaluation loss.

In this thesis, we focused on bridging the gap between probabilistic and non-probabilistic models by developing learning objectives for distributions that take into account the task at hand. First, we considered the case where the uncertainty in the prediction process came from partial annotation of the data. In the context of weakly supervised structured output prediction, we proposed a novel

entropy-based prediction criterion that unifies existing empirical risk minimisation methods. Using our proposed model, we experimentally demonstrated that, for large and ambiguous latent spaces, performing prediction via the minimisation of the latent space uncertainty leads to best performance. Second, we tackled the more challenging setting where there is ambiguity in the prediction itself. We proposed the DISsimilarity COefficient Networks (DISCO Nets), a new type of model for probabilistic structured output prediction, and show theoretical guarantees of our objective function. In contrast to existing models, our training objective can be tailored to a user-defined loss. Experiments on continuous data showed that (i) by modeling uncertainty in the output value, DISCO Nets outperform equivalent non-probabilistic predictive models and (ii) DISCO Nets accurately capture complex, structured probability distributions, outperforming existing probabilistic models. In the specific case of discrete data, our learning procedure is non-differentiable and we derive an approximation algorithm to learn the parameters of the model. Experimental evaluation in the discrete setting confirmed that DISCO Nets capture structure in the output to predict by using non-decomposable loss functions. Finally, we considered the case where there is no explicit loss function. The user needs are expressed via a particular grouping of the data, where within a group the observations share a common factor of variation. In this context, the goal is to learn a controllable representation that decomposes the data according to the semantics embedded in the grouping. As existing models are not suited to leverage grouping supervision, we presented a new deep probabilistic model, the Multi-Level Variational Autoencoder (ML-VAE). The ML-VAE learns a disentangled representation of a set of grouped observations by working both at the group level and the observation level, while retaining efficient test-time inference. Quantitative and qualitative evaluations showed that by taking advantage of the grouping, the ML-VAE model (i) learns a semantically meaningful disentanglement of grouped data (ii) enables

manipulation of the latent representation, and (iii) generalises to unseen groups.

6.1 Future work

Our work opens up several interesting directions for future research, which we briefly outline below.

Entropy-based prediction criterion. Experimental evaluation using our Unified Framework supports the use of an entropy-based prediction criterion. We note in section 2.1 that we can approach existing entropies using scoring rules theory. For example, Musio and Dawid (2016) show that the Rényi entropy can be related to a strictly proper (negatively oriented) scoring rule. The associated divergence is a variant of the classic Rényi divergence. We contemplate pursuing a similar analysis for the Aczél and Daròczy (AD) entropy. Our hope is to identify divergences that could be used as the objective function. Towards this goal, the first step would be to assess if there exist values for the parameters α, β that ensure the concavity of the AD entropy.

Scoring rules based deep probabilistic objectives. The DISCO Nets' objective, presented in Chapter 4, compares conditional distributions of the output for a specific input, and integrates the resulting divergence over all possible inputs. Ren et al. (2016) extend deep probabilistic models based on Maximum Mean Discrepancy (MMD) (Dziugaite et al., 2015; Li et al., 2015) to the conditional case. While MMD can be related to the Kernel score (Huszár, 2013), contrarily to our work they use kernel mean embeddings of conditional distributions. This allow them to compare larger sets of outputs by computing a kernel-based distance between all pairs of outputs. The distance between two outputs in the resulting objective is weighted by a factor based on the kernel embedding of the their respective in-

put. By contrast, our setting would allow us to potentially weight each divergence term $d(P, Q)^x$, while keeping the theoretical guarantee of our objective function. For instance, during training the model could focus on specific parts of the input space, by weighting each input according to the amplitude of its corresponding divergence (or its gradient).

Another direction for future research is to employ a mixture of scoring rules as the training objective. Clearly, a non-negative weighted sum of strictly proper scoring rules remains strictly proper. We identify two benefits of this method. First, this could allow a user to embed different needs in the training of the model. Second, we could consider using a combination of (i) a scoring rule that has appealing properties, such as high discrimination ability, but is not specifically of interest to the user and (ii) a scoring rule built on a user-defined loss, and adapt their respective weights during training. Recent works propose to use a combination of existing deep probabilistic training objectives, motivated by the different advantages of each model (Li et al., 2017; Zhao et al., 2017b). We believe the proposed method would be a formal and general approach.

The grouping supervision. In Chapter 5, we consider that the data are grouped according to one factor of variation, and use a two-part latent code. However, the proposed model could handle multiple grouping of the same data set. For instance, we could consider a data set of face images grouped once by identity, and once by facial expression. In this setting, the model would have a three-part latent representation corresponding to identity, facial expression, and the rest.

Furthermore, Chapter 5 tackles the case where the intended use of the model is embedded in a specific grouping of the data, given by the user. However there might be some uncertainty in the grouping itself. This can be due to errors in the data collection process, or the use of multiple groupings gathered from different

users. Consider for example that we group the US10k Face Data set (Bainbridge et al., 2013) used is Chapter 4 on a subjective attribute such as kind. This would give multiple partionings along the same factor of variation. We could consider the different partionings as *relations* and introduce a group random variable in the form of a binary matrix where the rows are the samples and the columns are the groups. Using a Chinese Restaurant Process prior (Pitman, 2002) on the group variable (to have an undefined, unbounded number of groups) we would infer its posterior from the observed relations, similar to the Infinite Relation Model (Kemp et al., 2006).

Chapter 7

Appendices

7.1 Appendix for Chapter 1

7.1.1 Introductory example experiment

We provide here details on the toy example presented in section 1.1.3. We use the following experimental setting. All covariances for the bidimensional distributions are diagonal, therefore all bidimensional Normal distributions are parametrised by 4 parameters $(\mu_1, \mu_2, \sigma_1, \sigma_2)$ where μ, σ is a mean-variance pair on each dimension. We consider a data distribution that is a mixture of 2 bidimensional Normal distributions, referred to as $\mathcal{GMM}$. The first Normal of the mixture, $\mathcal{G}_1$, is parametrised by $(1, 1.5, 2, 0.8)$ and the second Normal $\mathcal{G}_2$ is parametrised by $(0, -0.5, 0.7, 0.6)$. The mixture weights are 0.7 and 0.3, such that $\mathcal{GMM} = 0.7 \times \mathcal{G}_1 + 0.3 \times \mathcal{G}_2$. The training data set is composed of $n = 10000$ examples drawn randomly from $\mathcal{GMM}$, denoted $(x_1, ..., x_n)$. The testing data set is composed of 1000 examples drawn randomly from $\mathcal{GMM}$.

We consider two models to capture the data distribution $\mathcal{GMM}$. Each model is able to represent a bidimensional Normal distribution parametrised

by $(\mu_1, \mu_2, \sigma_1, \sigma_2)$. Model A employs a loss that emphasises on the first dimension of the data, specified for $x = (x_1, x_2), x' = (x'_1, x'_2) \in \mathbb{R}^2$ by $\Delta_A(x-x') = (10\times(x_1-x'_1)^2+0.1\times(x_2-x'_2)^2\Big)^{1/2}$. Model B does the opposite and employs the loss function $\Delta_B(x-x') = \Big(0.1\times(x_1-x'_1)^2+10\times(x_2-x'_2)^2\Big)^{1/2}$. Each model performs a grid search over the best parameters values for $(\mu_1, \mu_2, \sigma_1, \sigma_2)$. The sets in which to search for the parameters are the same in both dimensions and both models. The set to search the means ranges from -3 to 3 by 1, and the set to search the variances ranges from 0.1 to 2 by 0.5.

Specifically, to find the best parameters, we draw $K = 2$ samples from the model for each data sample x_i and compute a linear estimate of the following form, based on the DISCO Nets model we present in Chapter 4,

$$\frac{1}{n}\sum_{i=1}^{n}\Big[\frac{1}{K}\sum_{k}\Delta_M(x_i, G_M(z_{k,i})) - \frac{1}{2}\frac{1}{K(K-1)}\sum_{k}\sum_{k'\neq k}\Delta_M(G_M(z_{k',i}), G_M(z_{i,k}))\Big].$$

where M indexes the model, Δ_M is the model's specific loss, and $G_M(z_{k,i})$ is the k^{th} sample drawn from M for the training index i. During testing, for each model, for each input we draw $K = 10$ samples from the model per index, and choose a point-wise prediction with the Maximum Expected Utility (MEU) method, also described in Chapter 4, section 4.3.2. The MEU method employs the same loss as the evaluation loss. Each model is tested with its own loss and the loss of the other model.

7.2 Appendix for Chapter 2

7.3 Probability theory

We present in this section well-known results in probability theory that we use throughout the thesis.

7.3.1 Definitions and notations

We begin by introducing the concept of probability spaces. A probability space $(\Omega, \mathcal{F}, \mathbb{P})$ consists of three parts:

1. The sample space Ω which is the set of all possible outcomes.
2. The σ-algebra $\mathcal{F}$ which is a set of events, that is, a collection of subsets of Ω such that (i) Ω is in $\mathcal{F}$, (ii) $\mathcal{F}$ is closed under complement: if $A \in \mathcal{F}$, so is the complement of A, and (iii) $\mathcal{F}$ is closed under countable unions: if $A_i \in \mathcal{F}$ for $i = 1, 2, \ldots$ then $\left(\bigcup_{i=1}^{\infty} A_i \right) \in \mathcal{F}$. Using De Morgan's law in conjunction with (ii) and (iii), $\mathcal{F}$ is closed under countable intersections: if $A_i \in \mathcal{F}$ for $i = 1, 2, \ldots$ then $\left(\bigcap_{i=1}^{\infty} A_i \right) \in \mathcal{F}$.
3. The probability measure $\mathbb{P}$ is a function $\mathcal{F} \rightarrow [0, 1]$, such that $\mathbb{P}(\emptyset) = 0$, $\mathbb{P}(\Omega) = 1$, and $\mathbb{P}$ is countably additive: if $\{A_i\}_{i=1}^{\infty} \subseteq \mathcal{F}$ is a countable collection of pairwise disjoint sets, then $\mathbb{P}(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} \mathbb{P}(A_i)$.

Probability spaces are the basic notion to define random variables. A random variable is a variable quantity whose value depends on possible outcomes. Formally, let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and $(\mathcal{X}, \mathcal{B}(\mathcal{X}))$ a measurable space, where $\mathcal{B}(\mathcal{X})$ is the Borel algebra on $\mathcal{X}$. A random variable is a measurable function $\boldsymbol{x} : \Omega \rightarrow \mathcal{X}$. A random variable does not return a probability but a numerical quantity. The

probability that $\boldsymbol{x}$ takes a value x is the probability measure of the set of outcomes $\{\omega \in \Omega : \boldsymbol{x}(\omega) = x\}$ denoted $\mathbb{P}(\boldsymbol{x} = x)$. A realisation of a random variable is the value that is actually observed, $x = \boldsymbol{x}(\omega)$.

Let us illustrate these concepts with a concrete example. Consider the experiment of flipping a coin twice. The sample space Ω consists of four possible outcomes $\Omega = \{(\text{tails}, \text{tails}), (\text{heads}, \text{heads}), (\text{heads}, \text{tails}), (\text{tails}, \text{heads})\}$. The σ-algebra $\mathcal{F}$ is a collection of events, and an event is a collection of outcomes. For example the event "at least one heads occurs" is composed of $\{(\text{heads}, \text{heads}), (\text{heads}, \text{tails}), (\text{tails}, \text{heads})\}$. A random variable $\boldsymbol{x} : \Omega \to \mathcal{X}$ could be the number of heads that occured during the experiment, for example if the outcome of the experiment is $\omega = (\text{heads}, \text{tails}) \in \Omega$ then $\boldsymbol{x}((\text{heads}, \text{tails})) = 1$.

Given a probability space and a random variable $\boldsymbol{x}$, often we are interested in the probability of $\boldsymbol{x}$ taking values over a certain range, and this is defined by the cumulative distribution function.

Definition 7.1 (Cumulative distribution function)**.** *The cumulative distribution function (cdf) of a random variable $\boldsymbol{x}$, denoted $P(\boldsymbol{x})$, is a function $P : \mathcal{X} \to [0, 1]$ such that*

$$P(x) := \mathbb{P}(\boldsymbol{x} \leq x).$$

In this thesis, we will always consider real-valued random variables, that is, $\mathcal{X} \subseteq \mathbb{R}^d, d \geq 1$ ($d = 1$ for scalar-valued random variables, and $d \geq 2$ for vector-valued random variables). We refer to continuous random variables when the set $\mathcal{X}$ is uncountable and the cdf is absolutely continuous, and discrete random variables when the set $\mathcal{X}$ is countable, in which case the cdf is discontinuous.

If we are interested in the likelihood that a random variable takes a certain value, we turn to probability density functions if $\boldsymbol{x}$ is a continuous random variable.

Definition 7.2 (Probability density function)**.** *The probability density func-*

tion (pdf) of a continuous random variable $\boldsymbol{x}$*, denoted* $p(\boldsymbol{x})$*, is a function* $p : \mathcal{X} \to [0, +\infty)$ *such that*

$$p(x) := \frac{\mathrm{d}}{\mathrm{d}x} P(x).$$

Pdfs satisfy $p(x) \geq 0 \ \forall x \in \mathcal{X}$*, and* $\int_{\mathcal{X}} p(x)\mathrm{d}x = 1$.

If $\boldsymbol{x}$ is a discrete random variable, we turn to probability mass functions.

Definition 7.3 (Probability mass function)**.** *The probability mass function (pmf) of a discrete random variable* $\boldsymbol{x}$*,denoted* $p(\boldsymbol{x})$*, is a function* $p : \mathcal{X} \to [0, +\infty)$ *such that*

$$p(x) := \mathbb{P}(\boldsymbol{x} = x).$$

Pmfs satisfy $p(x) \geq 0 \ \forall x \in \mathcal{X}$*,* $\sum_{x \in \mathcal{X}} p(x) = 1$*, and can be extended to all real values with* $p(x) = 0 \ \forall x \notin \mathcal{X}$.

In the case of scalar-valued random variables, we have

$$P(x) = \int_{-\infty}^{x} p(t)\mathrm{d}t,$$
$$P(x) = \sum_{t \leq x} p(t).$$

for continuous and discrete random variables respectively.

In this thesis, we use the notation $P(x)$ to refer to a value of the cdf. In few cases, we will explicitly write $P(\boldsymbol{x} = x)$ instead of $P(x)$ to emphasise on the random variable $\boldsymbol{x}$ taking a specific value x. Similarly, we will use $p(x)$, and in few cases we will write $p(\boldsymbol{x} = x)$ instead of $p(x)$.

Remark 7.1 (Terminology)**.** *Given a probability space* $(\Omega, \mathcal{F}, \mathbb{P})$*, the formal definition of the distribution of a random variable* $\boldsymbol{x} : \Omega \to \mathcal{X}$ *refers to* $\mathbb{P}_{\boldsymbol{x}} = \mathbb{P} \circ \boldsymbol{x}^{-1}$*. The*

distribution $\mathbb{P}_{\boldsymbol{x}}$ is a probability measure on the measurable space $(\mathcal{X}, \mathcal{B}(\mathcal{X}))$ (Khoshnevisan, 2007; Leadbetter et al., 2014). Let us consider a scalar real-valued random variable $\boldsymbol{x} : \Omega \to \mathbb{R}$. We can associate the probability space $(\mathbb{R}, \mathcal{B}(\mathbb{R}), \mathbb{P}_{\boldsymbol{x}})$. In this case, the cdf is defined as $P(x) = \mathbb{P}_{\boldsymbol{x}}((-\infty, x))$ and uniquely defines the measure $\mathbb{P}_{\boldsymbol{x}}$. The proof relies on the Carathèodory extension theorem, see Khoshnevisan (2007); Leadbetter et al. (2014), and extends to the case of vector-valued random variables. While the cdf uniquely defines the distribution, the pdf does not. The pdf can be changed on a finite set of points without changing the integral. However, the non-uniquess of the pdf rarely poses an issue. Therefore, we will employ the term distribution of a random variable $\boldsymbol{x}$ to refer to either the cdf, pdf (when defined) or pmf (when defined) of $\boldsymbol{x}$. The meaning should be clear from context.

Remark 7.2 (Notations when comparing probability distributions)**.** *Formally, a random variable is defined for a given probability space, therefore the distribution is unique and corresponds to $\mathbb{P}_{\boldsymbol{x}}$. In this thesis, we will be interested in comparing probability distributions of a specific random variable $\boldsymbol{x}$. In this case, we will not consider the random variable to be associated to a specific probability space $(\Omega, \mathcal{F}, \mathbb{P})$. We will consider $\boldsymbol{x}$ as a measurable function from the measurable space $(\Omega, \mathcal{F})$ to the measurable space $(\mathcal{X}, \mathcal{B}(\mathcal{X}))$ where $\mathcal{X} = \mathbb{R}^d, d \geq 1$, and a class of probability measures $\mathcal{P}'$ on $(\Omega, \mathcal{F})$. Two probability measures $\mathbb{P}, \mathbb{Q} \in \mathcal{P}'$ corresponds to two distributions $\mathbb{P}_{\boldsymbol{x}}, \mathbb{Q}_{\boldsymbol{x}} \in \mathcal{P}$, where $\mathcal{P}$ is a class of probability measures on $(\mathcal{X}, \mathcal{B}(\mathcal{X}))$. Furthermore, if the sample space is $\mathcal{X}$, we implicitly consider the class of probability measures $\mathcal{P} = \{\mathbb{P}_{\boldsymbol{x}} | \mathbb{P}_{\boldsymbol{x}} = \mathbb{P} \circ \boldsymbol{x}^{-1}, \mathbb{P} \in \mathcal{P}'\}$. In this context since the cdf P is uniquely associated to the measure $\mathbb{P}_{\boldsymbol{x}} \in \mathcal{P}$, we will associate the measure $\mathbb{P}_{\boldsymbol{x}}$ and its cdf P, and denote $P \in \mathcal{P}$.*

When we need to make an optimal choice given an experiment characterised by a random variable, we are interested in the average value of repetitions of the experiment. This quantity is the expected value of the random variable.

Definition 7.4 (Expected value)**.** *The expected value of a continuous random variable* $\boldsymbol{x}$ *is*

$$\mathbb{E}[\boldsymbol{x}] = \int_{\mathcal{X}} xp(x)\mathrm{d}x.$$

The expected value of a discrete random variable $\boldsymbol{x}$ *is*

$$\mathbb{E}[\boldsymbol{x}] = \sum_{x \in \mathcal{X}} xp(x).$$

One can also be interested in the spread of the possible values around the expected value. This quantity is measured by the variance of the random variable.

Definition 7.5 (Variance)**.** *The variance of a random variable* $\boldsymbol{x}$ *is*

$$\mathbb{V}[\boldsymbol{x}] = \mathbb{E}\left[(\boldsymbol{x} - \mathbb{E}[\boldsymbol{x}])^2\right].$$

In the remainder of the thesis, we use the notation $x \sim P(\boldsymbol{x})$ or $x \sim p(\boldsymbol{x})$ to denote that x is a sampled value for the random variable $\boldsymbol{x}$ with cdf P and pdf (or pmf) p. We will write equivalently $\mathbb{E}_{x\sim P(\boldsymbol{x})}[\ ]$, $\mathbb{E}_{x\sim P}[\ ]$, $\mathbb{E}_{x\sim p(\boldsymbol{x})}[\ ]$ or $\mathbb{E}_{p(\boldsymbol{x})}[\ ]$ to refer to the expectation with respect to $\boldsymbol{x}$ with cdf P and pdf (or pmf) p. This is not an inconsistency but it is for the purpose of clarity without overloading the reader, using the most relevant notation given the context.

When dealing simultaneously with more than one random variable, we consider the joint cumulative distribution function. For clarity, the next definitions are in the context of two jointly distributed variables $\boldsymbol{x}$ and $\boldsymbol{y}$ with joint distribution $p(\boldsymbol{x}, \boldsymbol{y})$, but the same definitions and results apply to more than two variables. The marginal distribution of $\boldsymbol{x}$ given the joint distribution $p(\boldsymbol{x}, \boldsymbol{y})$ is $p(\boldsymbol{x})$ and the total probability rule gives,

$$p(\boldsymbol{x}) = \sum_{y \in \mathcal{Y}} p(\boldsymbol{x}, y),$$

$$p(\boldsymbol{x}) = \int_{\mathcal{Y}} p(\boldsymbol{x}, y)\mathrm{d}y,$$

for discrete and continuous variables, respectively, and similarly for $p(\boldsymbol{y})$.

The conditional probability distribution of $\boldsymbol{x}$ given $\boldsymbol{y}$ is the probability distribution of $\boldsymbol{x}$ when $\boldsymbol{y}$ is known to be a particular value. The conditional pdf (or pmf) of $\boldsymbol{x}$ given the occurrence of the value y of $\boldsymbol{y}$ can be written as

$$p(\boldsymbol{x}|y) = \frac{p(\boldsymbol{x}, y)}{p(y)}, \tag{7.1}$$

and is defined whenever $p(y) \neq 0$. Note that $p(\boldsymbol{x}|y)$ corresponds to $p(\boldsymbol{x}|\boldsymbol{y})$ with $\boldsymbol{y}$ taking a particular value y. In this thesis, the conditional distribution $p(\boldsymbol{x}|\boldsymbol{y})$ refers to the function of $\boldsymbol{x}$ and $\boldsymbol{y}$ that, for each value of y such that $p(y) \neq 0$, returns the pdf (or pmf) $p(\boldsymbol{x}|y)$ on $\boldsymbol{x}$ as defined above.

Remark 7.3 (Conditional cdf)**.** *In this thesis we will sometimes employ the conditional cdf* $P(\boldsymbol{x}|y) = P(\boldsymbol{x}|\boldsymbol{y} = y)$*. In the case of discrete random variable* $\boldsymbol{y}$*, we can write whenever* $\mathbb{P}(\boldsymbol{y} = y) = p(y) \neq 0$*,*

$$P(\boldsymbol{x} = x|\boldsymbol{y} = y) = \mathbb{P}(\boldsymbol{x} \leq x|\boldsymbol{y} = y) = \frac{\mathbb{P}(\boldsymbol{x} \leq x, \boldsymbol{y} = y)}{\mathbb{P}(\boldsymbol{y} = y)} = \sum_{t \leq x} \frac{p(t, y)}{p(y)} = \sum_{t \leq x} p(t|y).$$

In the case of a continuous random variable $\boldsymbol{y}$*, the probability of the conditioning event is always* 0*, that is,* $\mathbb{P}(\boldsymbol{y} = y) = 0$*. We obtain* $P(\boldsymbol{x} = x|\boldsymbol{y} = y)$ *by integration of* $p(x|y)$ *as defined in equation (7.1) (or summation if* $\boldsymbol{x}$ *is discrete) (Wasserman, 2010),*

$$P(\boldsymbol{x} = x|\boldsymbol{y} = y) = \mathbb{P}(\boldsymbol{x} \leq x|\boldsymbol{y} = y) = \int_{-\infty}^{x} p(t|y)\mathrm{d}t,$$

which ensures that

$$\mathbb{P}(\boldsymbol{x} \leq x, \boldsymbol{y} \leq y) = \int_{-\infty}^{y} p(t')\mathbb{P}(\boldsymbol{x} \leq x|\boldsymbol{y} = t')\mathrm{d}t'.$$

The same applies to vector-valued random variables. For a detailed theory on conditioning, we refer the reader to Leadbetter et al. (2014).

Two random variables $\boldsymbol{x}$ and $\boldsymbol{y}$ are independent if the events $\{\boldsymbol{x} \leq x\}$ and $\{\boldsymbol{y} \leq y\}$ are independent events, that is to say

$$\mathbb{P}(\boldsymbol{x} \leq x, \boldsymbol{y} \leq y) = \mathbb{P}(\boldsymbol{x} \leq x)\mathbb{P}(\boldsymbol{y} \leq y).$$

In this case their joint cdf factorises,

$$P(\boldsymbol{x}, \boldsymbol{y}) = P(\boldsymbol{x})P(\boldsymbol{y}),$$

and so does the joint pdf (or joint pmf, or a joint mixed density in the case of a mix of continuous and discrete random variables). Similar to the case of a single random variable, one can compute interesting quantities to describe the relationship between two random variable. For example, the covariance between $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as follows.

Definition 7.6 (Covariance)**.** *The covariance between two random variables $\boldsymbol{x}$ and $\boldsymbol{y}$ is*

$$cov(\boldsymbol{x}, \boldsymbol{y}) = \mathbb{E}[(\boldsymbol{x} - \mathbb{E}[\boldsymbol{x}])(\boldsymbol{y} - \mathbb{E}[\boldsymbol{y}])]\,.$$

Remark 7.4 (Independent identically distributed random variables)**.** *We denote a collection of K random variables as $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K)$. If the random variables are independent and identically distributed, that is, they have the same cdf P, we use the shorthand notation iid. In this case we can consider a unique random variable, in the form of the function $\boldsymbol{x} : \Omega \to \mathcal{X}$. A sample X drawn from $\boldsymbol{X}$ is equivalent to K samples $x_1, \ldots, x_K$ independently drawn from $P(\boldsymbol{x})$, and the K-dimensional cdf of $\boldsymbol{X}$ can be computed as the product of K times the cdf P, denoted $P(\boldsymbol{x})^K$.*

7.3.2 Statistical estimation

Given the distribution $P(\boldsymbol{x})$ of a random variable $\boldsymbol{x}$, we will be interested in knowing some key quantity about P, denoted θ. However, often we do not have access to the full distribution $P(\boldsymbol{x})$ but only to a data set $\mathcal{D}$ consisting of n samples $\mathcal{D} = (x_1, \ldots, x_n)$ drawn from P. We consider in this section that the observations are iid, that is, $\mathcal{D} \sim P(\boldsymbol{x})^n$. Statistical estimation refers to estimating the value of θ based on the samples $x_1, \ldots, x_n$. An *estimator* is a rule to calculate the value of an estimate based on samples, and the value returned by the estimator is the estimate. One can design multiple estimators for the same quantity θ, therefore it is interesting to quantify and compare the quality of different estimators in order to elect the optimal one for a given application.

Two important quantities to evaluate the quality of a statistical estimator are its bias and variance. The bias of an estimator $\hat{\theta}$ is the difference between this estimator's expected value and the true value of the parameter being estimated. It is defined as follows.

$$\text{Bias}(\hat{\theta}, \theta) = \mathbb{E}_{\mathcal{D} \sim P(\boldsymbol{x})^n}\left[\hat{\theta}\right] - \theta.$$

Estimators with zero bias are unbiased estimators. The variance of an estimator measures its dispersion around its expected value, that is,

$$\text{Variance}(\hat{\theta}) = \mathbb{E}_{\mathcal{D} \sim P(\boldsymbol{x})^n}\left[(\hat{\theta} - \mathbb{E}_{\mathcal{D} \sim P(\boldsymbol{x})^n}\left[\hat{\theta}\right])^2\right].$$

The bias-variance decomposition shows that the sum of the variance and the square of the bias is equal to the mean squared error of the estimator, that is,

$$\text{MSE}(\hat{\theta}) = \mathbb{E}_{\mathcal{D} \sim P(\boldsymbol{x})^n}\left[(\hat{\theta} - \theta)^2\right] = \text{Variance}(\hat{\theta}) + \text{Bias}^2(\hat{\theta}, \theta).$$

This reveals a key trade-off, known as the *bias-variance trade-off*. When training a model one has to simultaneously minimise these two sources of errors. A good model has low bias, and low variance. However, we see that different estimators with the same mean square error can have a different bias-variance decomposition, which makes the balance between bias and variance difficult to achieve at training.

7.3.3 Proof of Theorem 2.1

Theorem 2.1 (Strict propriety of the Negative Kernel score with SPD kernels)**.** *We assume $\mathcal{X} \subseteq \mathbb{R}^d$ is countable and finite. Let $K(x, x') : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a strictly positive definite kernel on $\mathcal{X} \times \mathcal{X}$, and $\mathcal{P}$ be the space of probability measures on $\mathcal{B}(\mathcal{X})$. We denote x', x'' two samples, independently drawn, of the random variable $\boldsymbol{x}$ with distribution $Q \in \mathcal{P}$. The Negative Kernel score*

$$S(Q, x) = \mathbb{E}_{x' \sim Q}[K(x, x')] - \frac{1}{2}\mathbb{E}_{x', x'' \sim Q}[K(x', x'')]\,, Q \in \mathcal{P},$$

is strictly proper relative to the class of the probability measures on $\mathcal{B}(\mathcal{X})$ for which $\mathbb{E}_{x', x'' \sim Q}[K(x', x'')]$ is finite.

Proof. In the case of a discrete output variable, the sample space consists of mutually exclusive outcomes $\mathcal{X} = \{x_1, \ldots, x_m\}$. We consider the convex class of probability distribution $\mathcal{P}_m = \{P \text{ such that } p = (p(x_1), \ldots, p(x_m)),\ \forall i \in [1, m], p(x_i) \geq 0, \sum_{i=1}^{m} p(x_i) = 1\}$. Let us K be a non-negative, positive definite kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. The forecaster distribution is $Q(\boldsymbol{x})$ with pmf q. The scoring rule is the collection of m functions:

$$S(Q, x_i) = \mathbb{E}_{x' \sim Q}[K(x_i, x)] - \frac{1}{2}\mathbb{E}_{x', x'' \sim Q}[K(x', x'')]\,, i = 1, \ldots, m.$$

We will prove that S is a (strictly) proper scoring rule if K is a (strictly) definite

kernel. To do so, first we show that G is (strictly) convex when K is (strictly) positive definite. Then, we will show that S is a subgradient of G. This will conclude the proof, using Theorem 2 in Gneiting and Raftery (2007). Let us write the expected score function G,

$$\begin{aligned}
G(Q) = S(Q,Q) &= \mathbb{E}_{x\sim Q}[S(Q,x)] = \sum_{x_i\in\mathcal{X}} S(Q,x_i)q(x_i) \\
&= \sum_{x_i\in\mathcal{X}} \Big[\mathbb{E}_{x'\sim Q}[K(x_i,x')] - \frac{1}{2}\mathbb{E}_{x',x''\sim Q}[K(x',x'')]\Big]q(x_i) \\
&= \sum_{x_i\in\mathcal{X}} \Big[\sum_{x_j\in\mathcal{X}} K(x_i,x_j)q(x_j) - \frac{1}{2}\sum_{x_j\in\mathcal{X}}\sum_{x_k\in\mathcal{X}} K(x_k,x_j)q(x_j)q(x_k)\Big]q(x_i) \\
&= \sum_{x_i\in\mathcal{X}}\sum_{x_j\in\mathcal{X}} K(x_i,x_j)q(x_i)q(x_j) - \frac{1}{2}\sum_{x_j\in\mathcal{X}}\sum_{x_k\in\mathcal{X}} K(x_k,x_j)q(x_j)q(x_k) \\
&= \frac{1}{2}\sum_{x_i\in\mathcal{X}}\sum_{x_j\in\mathcal{X}} K(x_i,x_j)q(x_i)q(x_j) \\
&= \frac{1}{2}\mathbb{E}_{x_i,x_j\sim Q}[K(x_i,x_j)]\,.
\end{aligned}$$

Now, we prove that G is (strictly) convex when K is (strictly) positive definite. Consider a scalar $\lambda\in[0,1]$, and $P,Q\in\mathcal{P}_m$,

$$\begin{aligned}
&G(\lambda P + (1-\lambda)Q) \\
&= \frac{1}{2}\sum_{x_i\in\mathcal{X}}\sum_{x_j\in\mathcal{X}} K(x_i,x_j)[\lambda p(x_i) + (1-\lambda)q(x_i)][\lambda p(x_j) + (1-\lambda)q(x_j)] \\
&= \frac{1}{2}\sum_{x_i\in\mathcal{X}}\sum_{x_j\in\mathcal{X}} K(x_i,x_j)[\lambda^2 p(x_i)p(x_j) + (1-\lambda)^2 q(x_i)q(x_j) \\
&\quad + \lambda(1-\lambda)p(x_i)q(x_j) + \lambda(1-\lambda)p(x_j)q(x_i)] \\
&= \lambda^2 G(P) + (1-\lambda)^2 G(Q) \\
&\quad + \lambda(1-\lambda)\sum_{x_i\in\mathcal{X}}\sum_{x_j\in\mathcal{X}} K(x_i,x_j)p(x_i)q(x_j)
\end{aligned}$$

where we use $K(x_i,x_j) = K(x_j,x_i)$. Writing the difference be-

tween $G(\lambda P + (1-\lambda)Q)$ and $\lambda G(P) - (1-\lambda)G(Q)$, we get,

$$
\begin{aligned}
&G(\lambda P + (1-\lambda)Q) - \lambda G(P) - (1-\lambda)G(Q) \\
&= \lambda(\lambda-1)G(P) + (1-\lambda)((1-\lambda)-1)G(Q) \\
&\quad + \lambda(1-\lambda)\sum_{x_i\in\mathcal{X}}\sum_{x_j\in\mathcal{X}} K(x_i,x_j)p(x_i)q(x_j) \\
&= \lambda(\lambda-1)\Big[G(P)+G(Q) - \sum_{x_i\in\mathcal{X}}\sum_{x_j\in\mathcal{X}} K(x_i,x_j)p(x_i)q(x_j)\Big] \\
&= \lambda(\lambda-1)\Big[\frac{1}{2}\sum_{x_i\in\mathcal{X}}\sum_{x_j\in\mathcal{X}} K(x_i,x_j)(p(x_i)p(x_j)+q(x_i)q(x_j)-2p(x_i)q(x_j))\Big] \\
&= \lambda(\lambda-1)\Big[\frac{1}{2}\sum_{x_i\in\mathcal{X}}\sum_{x_j\in\mathcal{X}} K(x_i,x_j)(p(x_i)-q(x_i))(p(x_j)-q(x_j))\Big], \qquad (7.2)
\end{aligned}
$$

where we use again $K(x_i,x_j) = K(x_j,x_i)$. Since $\lambda \in [0,1]$ we have $\lambda - 1 \leq 0$, therefore $\lambda(\lambda-1) \leq 0$. If K is a (strictly) positive definite kernel, then the term $\frac{1}{2}\sum_{x_i\in\mathcal{X}}\sum_{x_j\in\mathcal{X}} K(x_i,x_j)(p(x_i)-q(x_i))(p(x_j)-q(x_j))$ is (strictly) positive except if $p=q$. It follows that G is (strictly) convex.

We are left to prove that the vector $S(Q)_m$ composed of the elements $S(Q,x_i)$ is a subgradient of G. We recall the definition of a subgradient below, adapted to our setting. That is, $\mathcal{P}_m$ is as defined above, $P,Q \in \mathcal{P}_m$ denote the cdfs and p,q denote the resulting pmfs.

Definition 7.7 (Subgradient)**.** *The vector $S(Q)_m$ is a subgradient of G at the point $Q \in \mathcal{P}_m$ if*

$$G(P) \geq G(Q) + \langle S(Q)_m, p-q\rangle, \ \forall P \in \mathcal{P}_m$$

where $\langle\cdot,\cdot\rangle$ denotes the standard scalar product. In our case, we have the following:

$$G(P) - G(Q) - \langle S(Q)_m, p-q\rangle$$

$$
\begin{aligned}
&= G(P) - G(Q) - \sum_{x_i \in \mathcal{X}} S(Q, x_i)(p(x_i) - q(x_i)) \\
&= G(P) - G(Q) - \sum_{x_i \in \mathcal{X}} S(Q, x_i)p(x_i) + \sum_{x_i \in \mathcal{X}} S(Q, x_i)q(x_i) \\
&= G(P) - G(Q) - \sum_{x_i \in \mathcal{X}} S(Q, x_i)p(x_i) + G(Q) \\
&= G(P) - \sum_{x_i \in \mathcal{X}} S(Q, x_i)p(x_i) \\
&= G(P) - \sum_{x_i \in \mathcal{X}} (\mathbb{E}_{x' \sim Q}[K(x_i, x')] - \frac{1}{2}\mathbb{E}_{x', x'' \sim Q}[K(x', x'')])p(x_i) \\
&= G(P) - \sum_{x_i \in \mathcal{X}} \sum_{x' \in \mathcal{X}} K(x_i, x')p(x_i)q(x') + \frac{1}{2}\mathbb{E}_{x', x'' \sim Q}[K(x', x'')] \\
&= G(P) + G(Q) - \sum_{x_i \in \mathcal{X}} \sum_{x' \in \mathcal{X}} K(x_i, x')p(x_i)q(x') \\
&= \frac{1}{2} \sum_{x_i \in \mathcal{X}} \sum_{x' \in \mathcal{X}} K(x_i, x')(p(x_i) - q(x_i))(p(x') - q(x')) \\
&\geq 0
\end{aligned}
$$

This proves that $S(Q)_m$ is a subgradient of G and we can use Theorem 2 in Gneiting and Raftery (2007) to conclude that S is a (strictly) proper scoring rule when K is a (strictly) positive definite kernel. □

7.4 Appendix for Chapter 3

7.4.1 Proof of Proposition 3.1

Proposition 3.1 (Decomposition of the AD entropy)**.** *Given an input x, the AD entropy of q_x^y, the generalised distribution of $\boldsymbol{z}$ for a value y, can be written as the sum of the negative log-likelihood of y and the AD entropy of p_x^y, the conditional distribution of the latent variable given the output and input,*

$$H_{\alpha,\beta}(q_x^y;\theta) = -\log p(y|x;\theta) + H_{\alpha,\beta}(p_x^y;\theta). \tag{3.4}$$

Proof. The AD entropy of the generalised distribution is

$$\begin{aligned}
H_{\alpha,\beta}(q_x^y;\theta) &= \frac{1}{1-\alpha}\log\Big(\frac{\sum_{z\in\mathcal{Z}} p(y,z|x;\theta)^{\alpha+\beta-1}}{\sum_{z\in\mathcal{Z}} p(y,z|x;\theta)^{\beta}}\Big) \\
&= \frac{1}{1-\alpha}\log\Big(\frac{\sum_{z\in\mathcal{Z}} p(z|y,x;\theta)^{\alpha+\beta-1}p(y|x;\theta)^{\alpha+\beta-1}}{\sum_{z\in\mathcal{Z}} p(z|y,x;\theta)^{\beta}p(y|x;\theta)^{\beta}}\Big) \\
&= \frac{\alpha-1}{1-\alpha}\log p(y|x;\theta) + \frac{1}{1-\alpha}\log\Big(\frac{\sum_{z\in\mathcal{Z}} p(z|y,x;\theta)^{\alpha+\beta-1}}{\sum_{z\in\mathcal{Z}} p(z|y,x;\theta)^{\beta}}\Big) \\
&= -\log p(y|x;\theta) + H_{\alpha,\beta}(p_x^y;\theta).
\end{aligned}$$

□

7.4.2 Proof of Proposition 3.2

Proposition 3.2 (Upper bound on the loss)**.** *The optimisation problem of equation (3.5) minimises an upper bound on the loss $\frac{C}{n}\sum_{i=1}^{n}\Delta(y_i, y_i(\theta))$ where y_i is the ground-truth output for the training input x_i and $y_i(\theta)$ is the predicted output for x_i. This upper bound is tightest when $\epsilon_y \searrow 0$.*

Proof. We will prove it by upper bounding the loss $\Delta(y_i, y_i(\theta))$ for a single training sample pair x_i, y_i. We have $y_i(\theta) = \arg\min_y H_{\alpha,\beta}(q^y_{x_i}; \theta)$ thus,

$$\begin{aligned}\Delta(y_i, y_i(\theta)) &\leq \Delta(y_i, y_i(\theta)) - \epsilon_z H_{\alpha,\beta}(q^{y_i(\theta)}_{x_i}; \theta) + \epsilon_z H_{\alpha,\beta}(q^{y_i}_{x_i}; \theta) \\ &\leq \epsilon_y \log \sum_{y \in \mathcal{Y}} \exp \frac{1}{\epsilon_y} \Big(\Delta(y_i, y) - \epsilon_z H_{\alpha,\beta}(q^y_{x_i}; \theta) \Big) + \epsilon_z H_{\alpha,\beta}(q^{y_i}_{x_i}; \theta).\end{aligned}$$

The first inequality holds by definition of $y_i(\theta)$. The second inequality holds as we show below. Firstly, we have,

$$\Delta(y_i, y_i(\theta)) - \epsilon_z H_{\alpha,\beta}(q^{y_i(\theta)}_{x_i}; \theta) \leq \max_{y \in \mathcal{Y}} (\Delta(y_i, y) - \epsilon_z H_{\alpha,\beta}(q^y_{x_i}; \theta)).$$

For any set $p = (p_1, ..., p_K) \in \mathbb{R}^K$ with $p_{max} = \max_{p_k \in p} p_k$:

$$\begin{aligned}\log \sum_{p_k \in p} \exp(p_k) &= p_{max} + \log \sum_{p_k \in p} \exp(p_k - p_{max}) \\ &= p_{max} + \log(1 + \sum_{\substack{p_k \in p, \\ p_k \neq p_{max}}} \exp(p_k - p_{max})) \geq p_{max} + \log(1) = p_{max},\end{aligned}$$

since log is an increasing function and exp is always positive. This also shows that the bound is tightest when $\epsilon_y \searrow 0$, in which case the log-sum-exp operation results in the maximisation over the output variable values y. Using this result with $p_y = \frac{1}{\epsilon_y}(\Delta(y_i, y) - \epsilon_z H_{\alpha,\beta}(q^y_{x_i}; \theta)), y \in \mathcal{Y}$ gives

$$\begin{aligned}\Delta(y_i, y_i(\theta)) - \epsilon_z H_{\alpha,\beta}(q^{y_i}_{x_i}; \theta) &\leq \max_{y \in \mathcal{Y}} (\Delta(y_i, y) - \epsilon_z H_{\alpha,\beta}(q^y_{x_i}; \theta)) \\ &= \epsilon_y \max_{y \in \mathcal{Y}} \frac{1}{\epsilon_y} (\Delta(y_i, y) - \epsilon_z H_{\alpha,\beta}(q^y_{x_i}; \theta)) \\ &\leq \epsilon_y \log \sum_{y \in \mathcal{Y}} \exp \frac{1}{\epsilon_y} (\Delta(y_i, y) - \epsilon_z H_{\alpha,\beta}(q^y_{x_i}; \theta)).\end{aligned}$$

□

7.4.3 Proof of Proposition 3.3

Proposition 3.3 (Optimisation problem as a DC)**.** *The optimisation problem in equation (3.5) can be equivalently written as a difference of convex (DC) functions for any values of* $\alpha \geq 0, \alpha \neq 1, \beta \geq 0, \alpha+\beta-1 \geq 0$ *using the following formulation,*

$$\begin{aligned}\min_{\theta\in\Theta} \quad & \frac{1}{2}||\theta||^2 + \frac{C}{n}\sum_{i=1}^{n}\Big[\epsilon_y \log\sum_{y\in\mathcal{Y}} \exp\frac{1}{\epsilon_y}\Big(\Delta(y_i,y) \\ & + F^+_{\alpha,\beta}(y,\theta) - F^-_{\alpha,\beta}(y,\theta)\Big) \\ & + G^+_{\alpha,\beta}(y_i,\theta) - G^-_{\alpha,\beta}(y_i,\theta)\Big],\end{aligned}$$

with $F^+_{\alpha,\beta}(y,\theta), F^-_{\alpha,\beta}(y,\theta), G^+_{\alpha,\beta}(y_i,\theta), G^-_{\alpha,\beta}(y_i,\theta)$ *all convex with respect to* θ*.*

Proof. We can write equation (3.5) as follows,

$$\begin{aligned}&\min_{\theta\in\Theta} \frac{1}{2}||\theta||^2 + \frac{C}{n}\sum_{i=1}^{n}\Big[\epsilon_y \log\sum_{y\in\mathcal{Y}} \exp\frac{1}{\epsilon_y}\Big(\Delta(y_i,y) - \epsilon_z H_{\alpha,\beta}(q^y_x;\theta)\Big) + \epsilon_z H_{\alpha,\beta}(q^{y_i}_x;\theta)\Big] \\ &= \min_{\theta\in\Theta} \frac{1}{2}||\theta||^2 + \frac{C}{n}\sum_{i=1}^{n}\Big[\epsilon_y \log\sum_{y\in\mathcal{Y}} \exp\frac{1}{\epsilon_y}\Big(\Delta(y_i,y) \\ &\quad - \frac{\epsilon_z}{1-\alpha}\big(\log\sum_{z\in\mathcal{Z}} p(y,z|x_i;\theta)^{\alpha+\beta-1} - \log\sum_{z\in\mathcal{Z}} p(y,z|x_i;\theta)^{\beta}\big)\Big) \\ &\quad + \frac{\epsilon_z}{1-\alpha}\big(\log\sum_{z\in\mathcal{Z}} p(y_i,z|x_i,\theta)^{\alpha+\beta-1} - \log\sum_{z\in\mathcal{Z}} p(y_i,z|x_i,\theta)^{\beta}\big)\Big] \\ &= \min_{\theta} \frac{1}{2}||\theta||^2 + \frac{C}{n}\sum_{i=1}^{n}\Big[\epsilon_y \log\sum_{y\in\mathcal{Y}} \exp\frac{1}{\epsilon_y}\Big(\Delta(y_i,y) \\ &\quad - \frac{\epsilon_z}{1-\alpha}\log\sum_{z\in\mathcal{Z}} p(y,z|x_i;\theta)^{\alpha+\beta-1} + \frac{\epsilon_z}{1-\alpha}\log\sum_{z\in\mathcal{Z}} p(y,z|x_i;\theta)^{\beta}\Big)\Big) \\ &\quad + \frac{\epsilon_z}{1-\alpha}\log\sum_{z\in\mathcal{Z}} p(y_i,z|x_i,\theta)^{\alpha+\beta-1} - \frac{\epsilon_z}{1-\alpha}\log\sum_{z\in\mathcal{Z}} p(y_i,z|x_i,\theta)^{\beta}\Big].\end{aligned}$$

The log-sum-exp function summing over the latent variable z is convex with respect to θ. Let's assume $\alpha > 1$, then $\frac{1}{1-\alpha} < 0$, and $-\frac{1}{1-\alpha} > 0$. In this case, $F^+_{\alpha,\beta}(y,\theta), F^-_{\alpha,\beta}(y,\theta), G^+_{\alpha,\beta}(y_i,\theta), G^-_{\alpha,\beta}(y_i,\theta)$ are convex with respect to θ. In

the case $\alpha < 1$, $\frac{1}{1-\alpha} > 0$ and we define:

- $F^{+}_{\alpha,\beta}(y,\theta) = \frac{\epsilon_z}{1-\alpha} \log \sum_{z \in \mathcal{Z}} p(y,z|x_i;\theta)^{\beta}$,
- $F^{-}_{\alpha,\beta}(y,\theta) = \frac{\epsilon_z}{1-\alpha} \log \sum_{z \in \mathcal{Z}} p(y,z|x_i;\theta)^{\alpha+\beta-1}$,
- $G^{+}_{\alpha,\beta}(y_i,\theta) = \frac{\epsilon_z}{1-\alpha} \log \sum_{z \in \mathcal{Z}} p(y_i,z|x_i;\theta)^{\alpha+\beta-1}$,
- $G^{-}_{\alpha,\beta}(y_i,\theta) = \frac{\epsilon_z}{1-\alpha} \log \sum_{z \in Z} p(y_i,z|x_i;\theta)^{\beta}$.

Recall our conditional model,

$$p(\boldsymbol{y} = y, \boldsymbol{z} = z|\boldsymbol{x} = x;\theta) := p(y,z|x;\theta) \propto \exp\big(\frac{1}{\epsilon_z}\theta^T\phi(x,y,z)\big).$$

Given this choice for $p(\boldsymbol{y} = y, \boldsymbol{z} = z|\boldsymbol{x} = x;\theta)$, the terms of the form $\log \sum_{z \in \mathcal{Z}} p(y,z|x_i;\theta)$ are convex. The normalisation constant cancel out among the terms $F^{+}_{\alpha,\beta}(y,\theta), F^{-}_{\alpha,\beta}(y,\theta), G^{+}_{\alpha,\beta}(y_i,\theta), G^{-}_{\alpha,\beta}(y_i,\theta)$. The log-sum-exp function summing over the output variable values y is convex and non-decreasing with respect to θ, thus taking the log-sum-exp of a difference of convex is still a difference of convex (Horst et al., 2000, Corollary 4.3). □

7.4.4 Equivalence of the UF with existing objectives

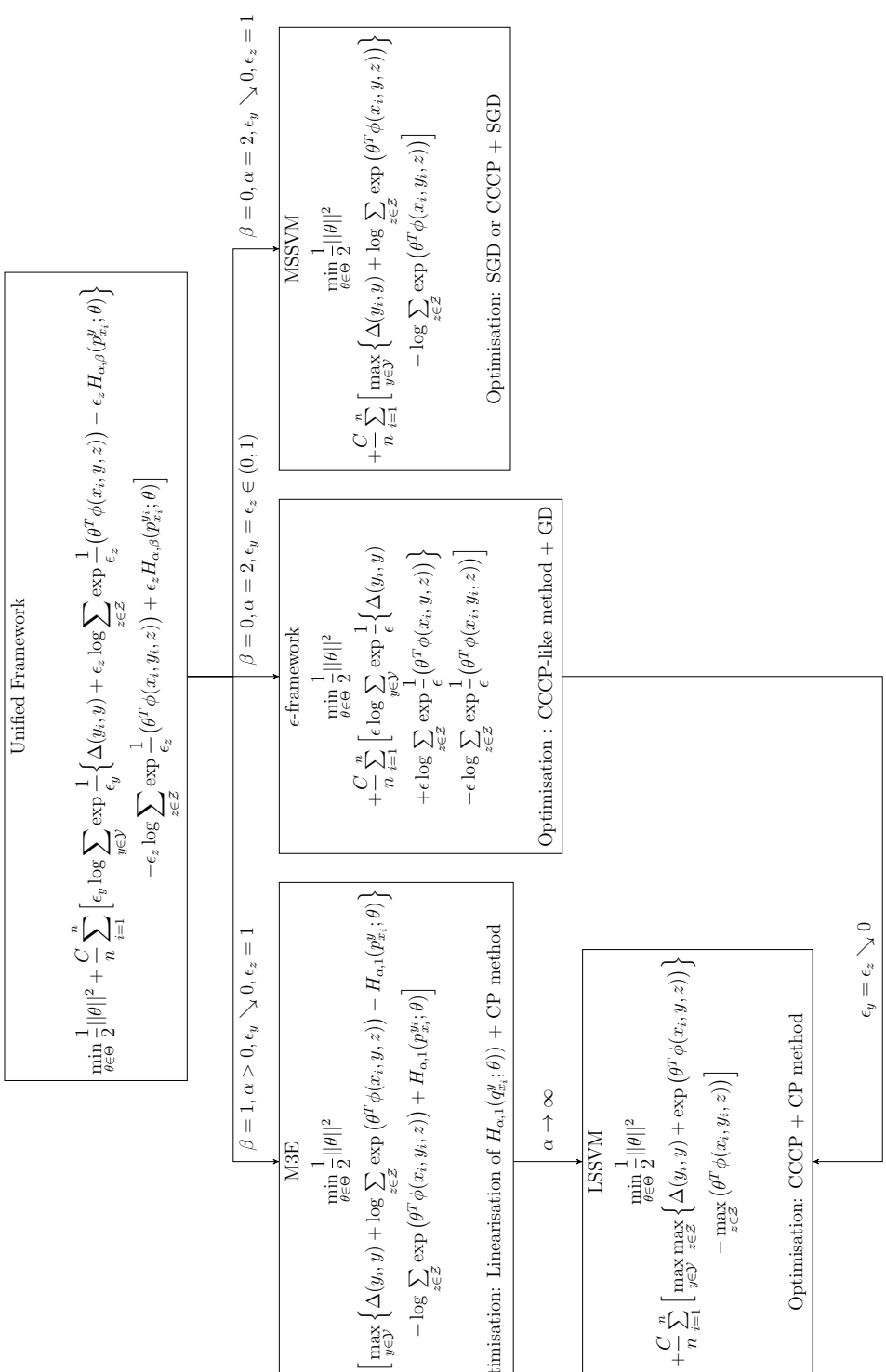


Figure 7.1: Equivalence of the UF's objective with existing ERM methods's objective. For each model we briefly explain their specific optimisation procedure: (Sub-) Gradient Descent (SGD or GD), Concave Convex Procedure (Yuille and Rangarajan, 2002) (CCCP) or Cutting Plane (Joachims et al., 2009) (CP).

7.4.5 Gradient descent with line search procedure in Algorithm 1.

Algorithm 5: Gradient descent with line search procedure in step 3 of Algorithm 1.

Current parameters value: θ_t.
Current objective value: $\text{obj}(\theta^t, \theta^t)$.
$\theta^l = \theta^t$
while $l \leq L$ *and* $\text{stop} = 0$ **do**
    Line search procedure:
    $b = 0$
    $c = 0.5$
    $\eta = \eta_0$
    $\text{stop}_2 = 0$
    $\theta_{\text{LS}} = \theta^t$
    while $a \leq A$ *and* $\text{stop}_2 = 0$ **do**
        $\theta_{\text{LS}} \leftarrow \theta^t - \eta \nabla_\theta \text{obj}(\theta, \theta^t)|_{\theta_t}$
        if $\text{obj}(\theta_{\text{LS}}, \theta^t) < \text{obj}(\theta^t, \theta^t) + c\eta \nabla_\theta \text{obj}(\theta, \theta^t)|_{\theta_t} \nabla_\theta \text{obj}(\theta, \theta^t)|_{\theta_t}$ **then**
            $\text{stop}_2 \leftarrow 1$
        else
            $\eta \leftarrow b\eta$
            $a \leftarrow a + 1$
        end
    end
    $\theta^l = \theta_{\text{LS}}$
    $\delta_{\text{obj}} = \text{obj}(\theta^t, \theta^t) - \text{obj}(\theta^{t+1}, \theta^t)$
    if $0 \leq \delta_{\text{obj}} < C\lambda$ **then**
        $\text{stop} \leftarrow 1$
        $\theta^{t+1} = \theta^l$
    else
        $l \leftarrow l + 1$
    end
end
return θ^{t+1}

7.4.6 Binary action classification

In all tables, the sign $\sim$ means that the parameters of the UF were set to replicate the existing model.

Training details. During our experiments, for each binary action classification tasks, we reweighed the positive samples by the scalar $\frac{|N|}{|P|}$ where N and P are the numbers of negatives and positives samples respectively. This is the weighted loss that we consider in our results analysis.

All models are initialised with LSSVM except for M3E with $\alpha \to \infty$, as its optimisation algorithm is slightly different than the one of LSSVM. M3E specifies a margin between the ground-truth output and all other incorrect outputs, whereas LSSVM considers all outputs (including the ground-truth). We use a random initialisation of the latent variable value for the M3E model with $\alpha \to \infty$.

The convergence tolerance is set to $\lambda = 0.01$. For each model, we tested the following range of parameters: $C = [0.1, 1, 10, 100, 1000]$, $\epsilon = [0.001, 0.01, 0.1, 1]$ (referring to the parameter of the ϵ-framework), $\epsilon_z = [0.001, 0.01, 0.1, 1.0]$, $\alpha = [0.01, 0.1, 2, 100, 1000, 10000]$ and $\beta = [0.0, 0.5, 1.0]$. In terms of computing requirements, all algorithms are comparable. Approximative computing time for all models with cross-validated parameters, averaged on all 10 classes, is of order $5 - 10$ minutes in CPU time on an Intel Xeon X7542 core.

Detailed results. Table 7.1 shows the per class test loss mean on the 5 folds for the 10 actions of the "trainval" data set of the PASCAL VOC 2011 action classification data set. We see that that the performances of UF as a replication and the corresponding existing model are similar. All models perform equivalently except that the output by marginalising the output and hidden variable as done by MSSVM is the less accurate criterion. We do not report p-values of left-tailed t-tests on the models test loss values over the 5 folds since there no statistical significance of outperformance between pairs of models.

Tables 7.2 and 7.3 report the best parameters chosen by cross-validation. The

analysis is not straightforward as we must take into account the values of $\alpha, \beta, \epsilon_z$. However, we see that in 9 classes out of 10 (all but "playing instrument") we have $\beta = 1$ so we recover the M3E model. With our setting, our prediction criterion can be rewritten in these case as follows,

$$\begin{aligned}
&\arg\min_{y\in\mathcal{Y}} \frac{\epsilon_z}{(1-\alpha)} \log\Big(\frac{\sum\limits_{z\in\mathcal{Z}} \exp(\theta^T\phi(x,y,z))^{\alpha/\epsilon_z}}{\sum\limits_{z\in\mathcal{Z}} \exp(\theta^T\phi(x,y,z))^{1/\epsilon_z}}\Big) \\
&= \arg\min_{y\in\mathcal{Y}} \frac{\epsilon_z}{(1-\alpha)} \log \sum_{z\in\mathcal{Z}} \exp(\theta^T\phi(x,y,z))^{\alpha/\epsilon_z} \\
&\quad - \frac{\epsilon_z}{(1-\alpha)} \log \sum_{z\in\mathcal{Z}} \exp(\theta^T\phi(x,y,z))^{1/\epsilon_z} \\
&= \arg\min_{y\in\mathcal{Y}} \frac{\epsilon_z}{(1-\alpha)} \log \sum_{z\in\mathcal{Z}} \exp(\alpha\theta^T\phi(x,y,z))^{1/\epsilon_z} \\
&\quad - \frac{\epsilon_z}{(1-\alpha)} \log \sum_{z\in\mathcal{Z}} \exp(\theta^T\phi(x,y,z))^{1/\epsilon_z}.
\end{aligned}$$

Out of these 9 classes where $\beta = 1$, in 4 classes ("jumping", "phoning","reading", "walking") we have $\alpha = 2$ and a very small value of ϵ_z, which gives a large value of $1/\epsilon_z$. The term ϵ_z in front of the log-sum-exp can be written as $\frac{1}{1/\epsilon_z}$ and $1 - \alpha = 1 - 2 = -1$, which gives:

$$\arg\min_{y\in\mathcal{Y}} -\frac{1}{1/\epsilon_z} \log \sum_{z\in\mathcal{Z}} \exp(2\theta^T\phi(x,y,z))^{1/\epsilon_z} + \frac{1}{1/\epsilon_z} \log \sum_{z\in\mathcal{Z}} \exp(\theta^T\phi(x,y,z))^{1/\epsilon_z}.$$

With a large value of $\frac{1}{\epsilon_z}$ we approximate the log-sum-exp as the maximum function, and this gives:

$$\arg\min_{y\in\mathcal{Y}} -\max_{z\in\mathcal{Z}} 2\theta^T\phi(x,y,z) + \max_{z\in\mathcal{Z}} \theta^T\phi(x,y,z) = \arg\max_{y\in\mathcal{Y}} \max_{z\in\mathcal{Z}} \theta^T\phi(x,y,z).$$

Out of these 9 classes where $\beta = 1$, in 2 other classes ("riding horse", "running")we

have small and equal values of $\alpha = \epsilon_z$,which gives

$$\arg\min_{y\in\mathcal{Y}} \frac{1}{(1-\alpha)/\epsilon_z} \log \sum_{z\in\mathcal{Z}} \exp(\theta^T\phi(x,y,z)) - \frac{1}{(1-\alpha)/\epsilon_z} \log \sum_{z\in\mathcal{Z}} \exp(\theta^T\phi(x,y,z))^{1/\epsilon_z}.$$

Again given the large value of $1/\epsilon_z$ we approximate the log-sum-exp as the maximum function in the second term and the first term $\frac{1}{(1-\alpha)/\epsilon_z} \log \sum_{z\in\mathcal{Z}} \exp(\theta^T\phi(x,y,z))$ is approximated to 0, and this gives:

$$\arg\min_{y\in\mathcal{Y}} -\frac{1}{(1-\alpha)} \max_{z\in\mathcal{Z}} \theta^T\phi(x,y,z) = \arg\max_{y\in\mathcal{Y}} \max_{z\in\mathcal{Z}} \theta^T\phi(x,y,z),$$

as $\alpha \leq 1$. Finally in 2 other classes ("riding bike", "taking photo") out of these 9 we have small but different values of α, ϵ_z, we approximate the log-sum-exp again, which gives

$$\begin{aligned}
&\arg\min_{y\in\mathcal{Y}} -\frac{\alpha}{(1-\alpha)} \max_{z\in\mathcal{Z}} \theta^T\phi(x,y,z) + \frac{1}{(1-\alpha)} \max_{z\in\mathcal{Z}} \theta^T\phi(x,y,z) \\
&\simeq \arg\max_{y\in\mathcal{Y}} \max_{z\in\mathcal{Z}} \theta^T\phi(x,y,z),
\end{aligned}$$

since α is small. This is how we conclude that in most cases the set of UF parameters chosen by cross-validation boils down to a prediction criterion that maximises over the output and latent variables. Similarly, the best α chosen for the M3E models is of high value in that case M3E recovers LSSVM, and the ϵ-framework best parameter ϵ chosen is of small value that also approximates LSSVM.

	jumping	playing instrument	phoning	riding bike	riding horse
LSSVM	45 ± 2.0	57 ± 2.5	55.7 ± 0.82	36 ± 2.6	32 ± 3.0
M3E $\alpha \to \infty$	45 ± 1.9	57 ± 2.5	54.6 ± 0.93	39 ± 4.6	33 ± 3.1
M3E	45 ± 2.6	56 ± 2.2	55.6 ± 0.73	35 ± 3.5	32 ± 1.4
MSSVM	51 ± 1.4	59 ± 3.9	57 ± 2.2	36 ± 3.5	35 ± 3.5
ϵ-framework	44 ± 2.3	57 ± 2.8	55 ± 1.2	36 ± 1.6	31 ± 2.7
UF	43 ± 1.9	57 ± 2.3	53 ± 1.6	32 ± 1.7	31 ± 2.5
UF $\sim$ LSSVM	44 ± 2.3	57 ± 2.5	54 ± 1.5	35 ± 2.9	35 ± 3.1
UF $\sim$ M3E	44 ± 2.3	57 ± 2.6	54 ± 1.5	32 ± 1.8	32 ± 2.8
UF $\sim$ MSSVM	51 ± 1.7	59 ± 3.9	58 ± 2.6	37 ± 3.0	37 ± 4.0
	reading	using computer	running	walking	taking photo
LSSVM	53 ± 1.5	51 ± 1.0	33 ± 3.9	42 ± 1.9	67 ± 3.6
M3E $\alpha \to \infty$	54 ± 1.6	49 ± 1.9	36 ± 3.4	43 ± 1.9	69 ± 3.6
M3E	53 ± 2.1	44 ± 2.1	33 ± 4.0	42 ± 1.8	67 ± 3.3
MSSVM	59 ± 2.4	51 ± 2.1	35 ± 4.2	46 ± 4.6	71 ± 2.6
ϵ-framework	53 ± 1.7	46 ± 2.2	31 ± 3.5	40 ± 1.6	66 ± 4.2
UF	53 ± 1.7	48 ± 3.1	33 ± 3.9	41 ± 1.7	66 ± 3.5
UF $\sim$ LSSVM	53 ± 1.4	50.6 ± 0.71	33 ± 3.7	41 ± 2.0	66 ± 3.5
UF $\sim$ M3E	53 ± 1.4	48 ± 3.1	33 ± 3.9	41 ± 2.0	66 ± 3.5
UF $\sim$ MSSVM	60 ± 2.0	49 ± 1.7	36 ± 3.2	44 ± 2.4	71 ± 2.6

Table 7.1: Per class test loss mean on the 5 folds (in %) $\pm$ standard error of the mean (in %) with cross-validated parameters on the PASCAL VOC 11 data set.

	jumping	playing instrument	phoning	riding bike	riding horse
LSSVM	$C = 10$	$C = 1$	$C = 1$	$C = 10$	$C = 1$
M3E $\alpha \to \infty$	$C = 10$	$C = 1$	$C = 1$	$C = 10$	$C = 1$
M3E	$C = 10, \alpha = 100$	$C = 1, \alpha = 10000$	$C = 1, \alpha = 10000$	$C = 10, \alpha = 0.1$	$C = 10, \alpha = 100$
MSSVM	$C = 100$	$C = 1$	$C = 10$	$C = 10$	$C = 100$
ϵ-framework	$C = 10, \epsilon = 0.01$	$C = 1, \epsilon = 0.1$	$C = 10, \epsilon = 1$	$C = 10, \epsilon = 0.1$	$C = 1, \epsilon = 0.001$
UF	$C = 10, \epsilon_z = 0.1, \alpha = 2, \beta = 1$	$C = 1, \epsilon_z = 0.1, \alpha = 2, \beta = 0.5$	$C = 1, \epsilon_z = 0.001, \alpha = 2, \beta = 1$	$C = 10, \epsilon_z = 0.1, \alpha = 0.01, \beta = 1$	$C = 1, \epsilon_z = 0.1, \alpha = 0.1, \beta = 1$
UF $\sim$ LSSVM	$C = 10$	$C = 1$	$C = 1$	$C = 10$	$C = 1$
UF $\sim$ M3E	$C = 10, \alpha = 100$	$C = 1, \alpha = 1000$	$C = 1, \alpha = 10000$	$C = 10, \alpha = 2$	$C = 1, \alpha = 100$
UF $\sim$ MSSVM	$C = 100$	$C = 1$	$C = 10$	$C = 10$	$C = 10$

Table 7.2: Cross-validated parameters for each model on the first five action classes.

	reading	using computer	running	walking	taking photo
LSSVM	$C = 1$	$C = 1$	$C = 10$	$C = 1$	$C = 1$
M3E $\alpha \to \infty$	$C = 1$	$C = 1$	$C = 10$	$C = 10$	$C = 1$
M3E	$C = 1, \alpha = 100$	$C = 10, \alpha = 0.1$	$C = 10, \alpha = 100$	$C = 1, \alpha = 10000$	$C = 1, \alpha = 100$
MSSVM	$C = 10$	$C = 100$	$C = 100$	$C = 10$	$C = 1$
ϵ-framework	$C = 1, \epsilon = 0.001$	$C = 100, \epsilon = 1$	$C = 10, \epsilon = 0.01$	$C = 10, \epsilon = 0.1$	$C = 1, \epsilon = 0.001$
UF	$C = 1, \epsilon_z = 0.1, \alpha = 2, \beta = 1$	$C = 10, \epsilon_z = 1, \alpha = 2, \beta = 1$	$C = 10, \epsilon_z = 0.01, \alpha = 0.01, \beta = 1$	$C = 1, \epsilon_z = 0.1, \alpha = 2, \beta = 1$	$C = 1, \epsilon_z = 0.001, \alpha = 0.1, \beta = 1$
UF $\sim$ LSSVM	$C = 1$	$C = 1$	$C = 10$	$C = 1$	$C = 1$
UF $\sim$ M3E	$C = 1, \alpha = 10000$	$C = 10, \alpha = 2$	$C = 10, \alpha = 100$	$C = 10, \alpha = 10000$	$C = 1, \alpha = 1000$
UF $\sim$ MSSVM	$C = 10$	$C = 100$	$C = 100$	$C = 100$	$C = 1$

Table 7.3: Cross-validated parameters for each model on remaining five action classes.

7.4.7 Multi-class gesture recognition

Training details. As in the binary classification experiment, all models are initialised with LSSVM except for M3E with $\alpha \to \infty$, as its optimisation algorithm is slightly different than the one of LSSVM (see section 7.4.6).

The convergence tolerance is set to $\lambda = 0.01$. We tested the same range of parameters as in the binary action classification experiment. Approximative computing time for all models with cross-validated parameters, averaged on all noise levels, is of order $2 - 3$ hours in CPU time on an Intel Xeon X7542 core.

Detailed results. Table 7.4 reports the average loss on the testing set for each model with respect to the noise level corrupting the data set (detailed results corresponding to Figures 3.4a and 3.4b). Tables 7.5 to 7.8 show the p-values for the

statistical left-tailed t-test on the models test loss values over the 5 folds performed for all pair of models. We can see that when no noise is added to the data, M3E with $\alpha \to \infty$ is outperformed by M3E, the UF, the UF replicating M3E and the UF replicating MSSVM, with statistical significance at level 0.05. The UF replicating LSSVM is also outperformed by the UF, the UF replicating M3E and the UF replicating MSSVM. As the noise level increases, LSSVM and the ϵ-framework are also outperformed by M3E, MSSVM, the UF, the UF replicating M3E and the UF replicating MSSVM.

	$\sigma = 0$cm	$\sigma = 1$cm	$\sigma = 5$cm	$\sigma = 8$cm
LSSVM	11 ± 1.1	11 ± 1.1	16 ± 1.5	22.8 ± 0.92
M3E $\alpha \to \infty$	11 ± 1.2	10.5 ± 0.40	18.6 ± 0.52	22.8 ± 0.94
M3E	8 ± 1.1	9 ± 1.0	11 ± 2.1	14 ± 1.5
MSSVM	9 ± 1.3	9 ± 1.1	12 ± 1.1	17 ± 1.6
ϵ-framework	10 ± 1.1	11.0 ± 0.74	15 ± 1.6	22.0 ± 0.65
UF	8 ± 1.1	8.6 ± 0.67	11.7 ± 0.85	15 ± 1.6
UF $\sim$ LSSVM	11 ± 1.1	11 ± 1.0	15 ± 1.5	22.8 ± 0.80
UF $\sim$ M3E	8 ± 1.1	9.1 ± 0.59	11.7 ± 0.85	15 ± 1.6
UF $\sim$ MSSVM	8 ± 1.1	9.0 ± 0.69	12 ± 1.0	15 ± 1.2

Table 7.4: Test loss mean on the 5 folds (in %) $\pm$ standard error of the mean (in %) with cross-validated parameters on the MSRC-12 data set, for different noise levels. Best performances in bold.

Table 7.9 shows cross-validated parameters for each model for each noise level. The best parameters for the UF are never boiling down to either LSSVM, MSSVM or the ϵ-framework. In other words the best parameters combination $(\epsilon_z, \alpha, \beta)$ always take in account the AD entropy of the hidden variable.

	LSSVM	M3E $\alpha \to \infty$	M3E	MSSVM	ϵ-framework	UF	UF $\sim$ LSSVM	UF $\sim$ M3E	UF $\sim$ MSSVM
LSSVM	0.0000	0.3750	0.9355	0.8544	0.7283	0.9494	0.4998	0.9494	0.9494
M3E $\alpha \to \infty$	0.6250	0.0000	0.9552	0.8984	0.8117	0.9644	0.6268	0.9644	0.9644
M3E	0.0645	**0.0448**	0.0000	0.3474	0.1520	0.5682	0.0613	0.5682	0.5682
MSSVM	0.1456	0.1016	0.6526	0.0000	0.2898	0.7059	0.1422	0.7059	0.7059
ϵ-framework	0.2717	0.1883	0.8480	0.7102	0.0000	0.8796	0.2676	0.8796	0.8796
UF	0.0506	**0.0356**	0.4318	0.2941	0.1204	0.0000	**0.0479**	0.5000	0.5000
UF $\sim$ LSSVM	0.5002	0.3732	0.9387	0.8578	0.7324	0.9521	0.0000	0.9521	0.9521
UF $\sim$ M3E	0.0506	**0.0356**	0.4318	0.2941	0.1204	0.5000	**0.0479**	0.0000	0.5000
UF $\sim$ MSSVM	0.0506	**0.0356**	0.4318	0.2941	0.1204	0.5000	**0.0479**	0.5000	0.0000

Table 7.5: p-values on the MSRC-12 testing data set, $\sigma = 0cm$

	LSSVM	M3E $\alpha \to \infty$	M3E	MSSVM	ϵ-framework	UF	UF $\sim$ LSSVM	UF $\sim$ M3E	UF $\sim$ MSSVM
LSSVM	0.0000	0.7739	0.9203	0.9127	0.6534	0.9683	0.4998	0.9443	0.9493
M3E $\alpha \to \infty$	0.2261	0.0000	0.8700	0.8527	0.3178	0.9779	0.2051	0.9538	0.9501
M3E	0.0797	0.1300	0.0000	0.5015	0.0970	0.6693	0.0697	0.5009	0.5444
MSSVM	0.0873	0.1473	0.4985	0.0000	0.1100	0.6574	0.0782	0.4991	0.5398
ϵ-framework	0.3466	0.6822	0.9030	0.8900	0.0000	0.9772	0.3354	0.9530	0.9550
UF	**0.0317**	0.0221	0.3307	0.3426	0.0228	0.0000	**0.0236**	0.2760	0.3381
UF $\sim$ LSSVM	0.5002	0.7949	0.9303	0.9218	0.6646	0.9764	0.0000	0.9566	0.9602
UF $\sim$ M3E	0.0557	**0.0462**	0.4991	0.5009	**0.0470**	0.7240	**0.0434**	0.0000	0.5591
UF $\sim$ MSSVM	0.0507	**0.0499**	0.4556	0.4602	**0.0450**	0.6619	**0.0398**	0.4409	0.0000

Table 7.6: p-values on the MSRC-12 data set, $\sigma = 1cm$

	LSSVM	M3E $\alpha \to \infty$	M3E	MSSVM	ϵ-framework	UF	UF $\sim$ LSSVM	UF $\sim$ M3E	UF $\sim$ MSSVM
LSSVM	0.0000	0.0771	0.9376	0.9587	0.5495	0.9780	0.5255	0.9780	0.9466
M3E $\alpha \to \infty$	0.9229	0.0000	0.9875	0.9987	0.9325	0.9999	0.9320	0.9999	0.9991
M3E	0.0624	**0.0125**	0.0000	0.3890	0.0751	0.4763	0.0673	0.4763	0.3238
MSSVM	**0.0413**	**0.0013**	0.6110	0.0000	0.0559	0.6480	**0.0457**	0.6480	0.3948
ϵ-framework	0.4505	0.0675	0.9249	0.9441	0.0000	0.9691	0.4752	0.9691	0.9280
UF	**0.0220**	**0.0001**	0.5237	0.3520	**0.0309**	0.0000	**0.0242**	0.5000	0.2408
UF $\sim$ LSSVM	0.4745	0.0680	0.9327	0.9543	0.5248	0.9758	0.0000	0.9758	0.9409
UF $\sim$ M3E	**0.0220**	**0.0001**	0.5237	0.3520	0.0309	0.5000	**0.0242**	0.0000	0.2408
UF $\sim$ MSSVM	0.0534	**0.0009**	0.6762	0.6052	0.0720	0.7592	0.0591	0.7592	0.0000

Table 7.7: p-values on the MSRC-12 data set, $\sigma = 5cm$

	LSSVM	M3E $\alpha \to \infty$	M3E	MSSVM	ϵ-framework	UF	UF $\sim$ LSSVM	UF $\sim$ M3E	UF $\sim$ MSSVM
LSSVM	0.0000	0.5026	0.9985	0.9885	0.7603	0.9972	0.5000	0.9972	0.9990
M3E $\alpha \to \infty$	0.4974	0.0000	0.9985	0.9883	0.7541	0.9972	0.4973	0.9972	0.9989
M3E	**0.0015**	**0.0015**	0.0000	0.1308	**0.0028**	0.4034	**0.0015**	0.4034	0.2900
MSSVM	**0.0115**	**0.0117**	0.8692	0.0000	**0.0205**	0.8100	**0.0113**	0.8100	0.7672
ϵ-framework	0.2397	0.2459	0.9972	0.9795	0.0000	0.9949	0.2209	0.9949	0.9981
UF	**0.0028**	**0.0028**	0.5966	0.1900	**0.0051**	0.0000	**0.0029**	0.5000	0.3935
UF $\sim$ LSSVM	0.5000	0.5027	0.9985	0.9887	0.7791	0.9971	0.0000	0.9971	0.9990
UF $\sim$ M3E	**0.0028**	**0.0028**	0.5966	0.1900	**0.0051**	0.5000	**0.0029**	0.0000	0.3935
UF $\sim$ MSSVM	**0.0010**	**0.0011**	0.7100	0.2328	**0.0019**	0.6065	**0.0010**	0.6065	0.0000

Table 7.8: p-values on the MSRC-12 data set, $\sigma = 8cm$

	$\sigma = 0$cm	$\sigma = 1$cm	$\sigma = 5$cm	$\sigma = 8$cm
LSSVM	$C = 100$	$C = 100$	$C = 100$	$C = 100$
M3E $\alpha \to \infty$	$C = 100$	$C = 100$	$C = 100$	$C = 100$
M3E	$C = 1000, \alpha = 0.01$	$C = 1000, \alpha = 0.01$	$C = 1000, \alpha = 0.01$	$C = 100, \alpha = 2$
MSSVM	$C = 100$	$C = 1000$	$C = 1000$	$C = 1000$
ϵ-framework	$C = 100, \epsilon = 0.1$	$C = 100, \epsilon = 0.1$	$C = 100, \epsilon = 0.1$	$C = 100, \epsilon = 0.1$
UF	$C = 1000, \epsilon_z = 1, \alpha = 0.01, \beta = 1$	$C = 1000, \epsilon_z = 1, \alpha = 2, \beta = 0.5$	$C = 1000, \epsilon_z = 1, \alpha = 0.1, \beta = 1$	$C = 1000, \epsilon_z = 1, \alpha = 0.1, \beta = 1$
UF $\sim$ LSSVM	$C = 100$	$C = 100$	$C = 100$	$C = 100$
UF $\sim$ M3E	$C = 1000, \alpha = 0.01$	$C = 1000, \alpha = 0.01$	$C = 1000, \alpha = 0.1$	$C = 1000, \alpha = 0.1$
UF $\sim$ MSSVM	$C = 1000$	$C = 1000$	$C = 1000$	$C = 1000$

Table 7.9: Cross-validated parameters for each model for each noise level.

7.5 Appendix for Chapter 4

7.5.1 Proof of Theorem 4.1

Theorem 4.1 (DISCO Nets Loss gradient theorem)**.** *For a finite space $\mathcal{Y}$, under some regularity conditions (detailed in the proof), the gradient $\nabla_\theta \mathrm{F}_\Delta(P, Q, P_x)$ is:*

$$\begin{aligned}\nabla_\theta \mathrm{F}_\Delta(P, Q, P_x) &= \pm \lim_{\epsilon \searrow 0} \frac{1}{\epsilon} \nabla_\theta \mathrm{F}^\epsilon_\Delta(P, Q, P_x) \\ &= \pm \lim_{\epsilon \searrow 0} \frac{1}{\epsilon} \Big(\nabla_\theta \mathrm{DIV}^\epsilon_\Delta(P, Q, P_x) - \frac{1}{2} \nabla_\theta \mathrm{DIV}^\epsilon_\Delta(Q, Q, P_x) \Big),\end{aligned}$$

where

$$\begin{aligned}&\nabla_\theta \mathrm{DIV}^\epsilon_\Delta(P, Q, P_x) \\ &\quad = \mathbb{E}_{x \sim P_x(\boldsymbol{x})} \big[\mathbb{E}_{y \sim P(\boldsymbol{y}|x)} \big[\mathbb{E}_{z \sim P_z(\boldsymbol{z})} \big[\nabla_\theta G_{y_p}(x, z; \theta) - \nabla_\theta G_{\hat{y}}(x, z; \theta) \big] \big] \big] , \\ &\hat{y} = \arg\max_{\tilde{y} \in \mathcal{Y}} G_{\tilde{y}}(x, z; \theta), \\ &y_p = \arg\max_{\tilde{y} \in \mathcal{Y}} G_{\tilde{y}}(x, z; \theta) \pm \epsilon \Delta(\tilde{y}, y), \\ &\nabla_\theta \mathrm{DIV}^\epsilon_\Delta(Q, Q, P_x) \\ &\quad = 2 \ \mathbb{E}_{x \sim P_x(\boldsymbol{x})} \big[\mathbb{E}_{z \sim P_z(\boldsymbol{z})} \big[\mathbb{E}_{z' \sim P_z(\boldsymbol{z})} \big[\nabla_\theta G_{y_q}(x, z; \theta) - \nabla_\theta G_{\hat{y}}(x, z; \theta) \big] \big] \big] , \\ &\hat{y}' = \arg\max_{\tilde{y} \in \mathcal{Y}} G_{\tilde{y}}(x, z'; \theta), \\ &y_q = \arg\max_{\tilde{y} \in \mathcal{Y}} G_{\tilde{y}}(x, z; \theta) \pm \epsilon \Delta(\tilde{y}, \hat{y}').\end{aligned}$$

Proof. We will prove the Loss gradient theorem for DISCO Nets in the positive update direction $\nabla_\theta \mathrm{F}_\Delta(P, Q, P_x) = \lim_{\epsilon \searrow 0} \frac{1}{\epsilon} \nabla_\theta \mathrm{F}^\epsilon_\Delta(P, Q, P_x)$, the proof in the negative direction follows a similar reasoning. We assume $\mathcal{Y}$ to be finite with $n_y = |\mathcal{Y}|$ and no tie in maximisation. Without loss of generality, for clarity purposes in this proof we assume a 1-dimensional output y (hence the grades are scalar-valued).

Part 1: The difficulty with DIV$_\Delta(Q, Q, P_x)$.

First, let us write the true gradient.

$$\nabla_\theta \mathrm{F}_\Delta(P, Q, P_x) = \nabla_\theta \mathrm{DIV}_\Delta(P, Q, P_x) - \frac{1}{2}\nabla_\theta \mathrm{DIV}_\Delta(Q, Q, P_x).$$

The approximation of the term $\nabla_\theta \mathrm{DIV}_\Delta(P, Q, P_x)$ by $\nabla_\theta \mathrm{DIV}^\epsilon_\Delta(P, Q, P_x)$ is directly obtained by using the loss-gradient theorem of Song et al. (2016). We repeat this theorem here, adapted to our terminology.

Theorem 7.1 (General Loss gradient theorem)**.** *When given a finite set $\mathcal{Y}$, a grading function $G_y(x;\theta)$ of $(x, y, \theta) \in \mathcal{X} \times \mathcal{Y} \times \Theta$, a set of input data $x \sim P(\boldsymbol{x})$ and output data $y \sim P(\boldsymbol{y}|\boldsymbol{x})$, a task-loss $\Delta(y, \hat{y})$, then, under some mild regularity conditions (see Song et al. (2016)), the direct loss gradient has the following form:*

$$\begin{aligned}
&\nabla_\theta \mathbb{E}_{x\sim P(\boldsymbol{x})}\left[\mathbb{E}_{y\sim P(\boldsymbol{y}|x)}[\Delta(y, \hat{y})]\right] \\
&\quad = \pm \lim_{\epsilon\to 0} \frac{1}{\epsilon}\mathbb{E}_{x\sim P(\boldsymbol{x})}\left[\mathbb{E}_{y\sim P(\boldsymbol{y}|x)}\left[\nabla_\theta G_{y_p}(x;\theta) - \nabla_\theta G_{\hat{y}}(x;\theta)\right]\right], \\
&\textit{where} \\
&\hat{y} = \arg\max_{\tilde{y}\in\mathcal{Y}} G_{\tilde{y}}(x;\theta), \\
&y_p = \arg\max_{\tilde{y}\in\mathcal{Y}} G_{\tilde{y}}(x;\theta) \pm \epsilon\Delta(\tilde{y}, y).
\end{aligned}$$

Proof. See Song et al. (2016) supplementary material. □

In our case, recall the expression of $\mathrm{DIV}_\Delta(P, Q, P_x)$.

$$\begin{aligned}
\mathrm{DIV}_\Delta(P, Q, P_x) &= \mathbb{E}_{x\sim P_x(\boldsymbol{x})}\left[\mathbb{E}_{y\sim P(\boldsymbol{y}|x)}\left[\mathbb{E}_{y'\sim Q(\boldsymbol{y}|x;\theta)}[\Delta(y, y')]\right]\right], \\
&= \mathbb{E}_{x\sim P_x(\boldsymbol{x})}\left[\mathbb{E}_{y\sim P(\boldsymbol{y}|x)}\left[\mathbb{E}_{z\sim P_z(\boldsymbol{z})}[\Delta(y, \hat{y})]\right]\right], \\
&\text{with } \hat{y} = \arg\max_{\tilde{y}\in\mathcal{Y}} G_{\tilde{y}}(x, z;\theta).
\end{aligned}$$

We apply Theorem 7.1 to our setting, considering the input pair (x, z), this gives:

$$\nabla_\theta \mathbb{E}_{x\sim P_x(\boldsymbol{x})}\left[\mathbb{E}_{y\sim P(\boldsymbol{y}|x)}\left[\mathbb{E}_{z\sim P_z(\boldsymbol{z})}[\Delta(y,\hat{y})]\right]\right]$$
$$= \lim_{\epsilon\searrow 0}\frac{1}{\epsilon}\mathbb{E}_{x\sim P_x(\boldsymbol{x})}\left[\mathbb{E}_{y\sim P(\boldsymbol{y}|x)}\left[\mathbb{E}_{z\sim P_z(\boldsymbol{z})}\left[\nabla_\theta G_{y_p}(x,z;\theta) - \nabla_\theta G_{\hat{y}}(x,z;\theta)\right]\right]\right],$$

where

$$\hat{y} = \arg\max_{\tilde{y}\in\mathcal{Y}} G_{\tilde{y}}(x,z;\theta),$$

$$y_p = \arg\max_{\tilde{y}\in\mathcal{Y}} G_{\tilde{y}}(x,z;\theta) + \epsilon\Delta(\tilde{y},y).$$

We denote

$$\nabla_\theta \mathrm{DIV}^\epsilon_\Delta(P,Q,P_x) = \mathbb{E}_{x\sim P_x(\boldsymbol{x})}\left[\mathbb{E}_{y\sim P(\boldsymbol{y}|x)}\left[\mathbb{E}_{z\sim P_z(\boldsymbol{z})}\left[\nabla_\theta G_{y_p}(x,z;\theta) - \nabla_\theta G_{\hat{y}}(x,z;\theta)\right]\right]\right],$$

and the results follows for $\nabla_\theta \mathrm{DIV}^\epsilon_\Delta(P,Q,P_x)$:

$$\nabla_\theta \mathbb{E}_{x\sim P_x(\boldsymbol{x})}\left[\mathbb{E}_{y\sim P(\boldsymbol{y}|x)}\left[\mathbb{E}_{z\sim P_z(\boldsymbol{z})}[\Delta(y,\hat{y})]\right]\right] = \lim_{\epsilon\searrow 0}\frac{1}{\epsilon}\nabla_\theta \mathrm{DIV}^\epsilon_\Delta(P,Q,P_x).$$

Another way of seeing that we can apply the Theorem 7.1 from Song et al. (2016) is that $\mathrm{DIV}_\Delta(P,Q,P_x)$ is computed using a single grades set $G_{\mathcal{Y}}(x,z;\theta)$ computed for the sample z. The difficulty arises when we consider $\mathrm{DIV}_\Delta(Q,Q,P_x)$:

$$\mathrm{DIV}_\Delta(Q,Q,P_x) = \mathbb{E}_{x\sim P_x(\boldsymbol{x})}\left[\mathbb{E}_{z\sim P_z(\boldsymbol{z})}\left[\mathbb{E}_{z'\sim P_z(\boldsymbol{z})}[\Delta(\hat{y},\hat{y}')]\right]\right],$$

$$\hat{y} = \arg\max_{\tilde{y}\in\mathcal{Y}} G_{\tilde{y}}(x,z;\theta),$$

$$\hat{y}' = \arg\max_{\tilde{y}\in\mathcal{Y}} G_{\tilde{y}}(x,z';\theta).$$

In this case both output values $\hat{y}, \hat{y}'$ are functions of θ, that is, both expectations are with respect to the parametrised distribution $Q(\boldsymbol{y}|x;\theta)$. Therefore the first argument in Δ is not fixed and a change $\theta \to \theta + \epsilon\delta\theta$ results in a change in both

outputs. This was not the case when considering $\mathrm{DIV}_\Delta(P, Q, P_x)$, where the first argument is $y \sim P(\boldsymbol{y}|x)$, independent of θ. Again, another way to express the difficulty is to see that we now have to deal with two grades sets $G_{\mathcal{Y}}(x, z; \theta)$ and $G_{\mathcal{Y}}(x, z'; \theta)$, computed respectively with the samples z and z'.

Part 2: Rewrite the result to prove

We explicitly write the dependance of $\hat{y}$ to $(x, z; \theta)$ by writing $\hat{y}(x, z; \theta)$. By definition of the gradient, we write

$$\begin{aligned}
&\delta\theta^T \nabla_\theta \mathrm{DIV}_\Delta(Q, Q, P_x) \\
&\quad = \delta\theta^T \nabla_\theta \mathbb{E}_{x\sim P_x(\boldsymbol{x})}\left[\mathbb{E}_{z\sim P_z(\boldsymbol{z})}\left[\mathbb{E}_{z'\sim P_z(\boldsymbol{z})}[\Delta(\hat{y}(x, z; \theta), \hat{y}(x, z'; \theta))]\right]\right] \\
&\quad = \lim_{\epsilon \searrow 0} \frac{1}{\epsilon} \mathbb{E}_{x\sim P_x(\boldsymbol{x})}\Big[\mathbb{E}_{z\sim P_z(\boldsymbol{z})}\left[\mathbb{E}_{z'\sim P_z(\boldsymbol{z})}[\Delta(\hat{y}(x, z; \theta + \epsilon\delta\theta), \hat{y}(x, z'; \theta + \epsilon\delta\theta))]\right] \\
&\qquad\qquad - \mathbb{E}_{z\sim P_z(\boldsymbol{z})}\left[\mathbb{E}_{z'\sim P_z(\boldsymbol{z})}[\Delta(\hat{y}(x, z; \theta), \hat{y}(x, z'; \theta))]\right]\Big].
\end{aligned}$$

For clarity we denote

$$\begin{aligned}
E_1 &= \mathbb{E}_{z\sim P_z(\boldsymbol{z})}\left[\mathbb{E}_{z'\sim P_z(\boldsymbol{z})}[\Delta(\hat{y}(x, z; \theta + \epsilon\delta\theta), \hat{y}(x, z'; \theta + \epsilon\delta\theta))]\right] \\
E_2 &= \mathbb{E}_{z\sim P_z(\boldsymbol{z})}\left[\mathbb{E}_{z'\sim P_z(\boldsymbol{z})}[\Delta(\hat{y}(x, z; \theta), \hat{y}(x, z'; \theta))]\right].
\end{aligned}$$

So we have

$$\delta\theta^T \nabla_\theta \mathrm{DIV}_\Delta(Q, Q, P_x) = \lim_{\epsilon \searrow 0} \frac{\mathbb{E}_{x\sim P_x(\boldsymbol{x})}[E_1 - E_2]}{\epsilon}. \tag{7.3}$$

Therefore, we will show that

$$\lim_{\epsilon \searrow 0} \frac{1}{\epsilon} \delta\theta^T \nabla_\theta \mathrm{DIV}^\epsilon_\Delta(Q, Q, P_x) = \lim_{\epsilon \searrow 0} \frac{\mathbb{E}_{x\sim P_x(\boldsymbol{x})}[E_1 - E_2]}{\epsilon}, \tag{7.4}$$

which will lead us to

$$\lim_{\epsilon \searrow 0} \frac{1}{\epsilon} \delta\theta^T \nabla_\theta \mathrm{DIV}^\epsilon_\Delta(Q, Q, P_x) = \delta\theta^T \nabla_\theta \mathrm{DIV}_\Delta(Q, Q, P_x), \tag{7.5}$$

If we show equation (7.5) for any vector $\delta\theta \in \mathbb{R}^d$ where d is the dimensionality of θ, we equivalently show the following.

$$\lim_{\epsilon \searrow 0} \frac{1}{\epsilon} \nabla_\theta \mathrm{DIV}^\epsilon_\Delta(Q, Q, P_x) = \nabla_\theta \mathrm{DIV}_\Delta(Q, Q, P_x). \tag{7.6}$$

We add and substract $E_3 = \mathbb{E}_{z\sim P_z(\boldsymbol{z})}\big[\mathbb{E}_{z'\sim P_z(\boldsymbol{z})}[\Delta(\hat{y}(x, z; \theta), \hat{y}(x, z'; \theta + \epsilon\delta\theta))]\big]$ and consider the following expression

$$\lim_{\epsilon \searrow 0} \frac{\mathbb{E}_{x\sim P_x(\boldsymbol{x})}[E_1 - E_3 + E_3 - E_2]}{\epsilon}.$$

We will prove

$$\boxed{\lim_{\epsilon \searrow 0} \frac{1}{\epsilon} \delta\theta^T \nabla_\theta \mathrm{DIV}^\epsilon_\Delta(Q, Q, P_x) = \lim_{\epsilon \searrow 0} \frac{\mathbb{E}_{x\sim P_x(\boldsymbol{x})}[E_1 - E_3 + E_3 - E_2]}{\epsilon}.}$$

Part 3: Considering $\lim_{\epsilon \searrow 0} \frac{\mathbb{E}_{x\sim P_x(\boldsymbol{x})}[E_1 - E_3]}{\epsilon}$***.***

Let us look at $\lim_{\epsilon \searrow 0} \frac{\mathbb{E}_{x\sim P_x(\boldsymbol{x})}[E_1 - E_3]}{\epsilon}$.

$$\begin{aligned} E_1 - E_3 = \mathbb{E}_{z\sim P_z(\boldsymbol{z})}[\mathbb{E}_{z'\sim P_z(\boldsymbol{z})}[\Delta(\hat{y}(x, z; \theta + \epsilon\delta\theta), \hat{y}(x, z'; \theta + \epsilon\delta\theta)) \\ - \Delta(\hat{y}(x, z; \theta), \hat{y}(x, z'; \theta + \epsilon\delta\theta))]]. \end{aligned}$$

The difficulty is that both outputs change when $\epsilon \searrow 0$. Hence, we cannot directly apply Theorem 7.1. Using a reasoning similar to McAllester et al. (2010); Song et al. (2016), we will show the following.

$$\lim_{\epsilon \searrow 0} \frac{\mathbb{E}_{x \sim P_x(\boldsymbol{x})}[E_1 - E_3]}{\epsilon} =$$

$$\lim_{\epsilon \searrow 0} \frac{1}{\epsilon} \delta\theta^T \mathbb{E}_{x \sim P_x(\boldsymbol{x})} \Big[\mathbb{E}_{z \sim P_z(\boldsymbol{z})} [\mathbb{E}_{z' \sim P_z(\boldsymbol{z})} [\nabla_\theta G_{y_q}(x, z; \theta) - \nabla_\theta G_{\hat{y}}(x, z; \theta)]] \Big],$$

$$\hat{y}' = \arg\max_{\tilde{y} \in \mathcal{Y}} G_{\tilde{y}}(x, z'; \theta),$$

$$\hat{y} = \arg\max_{\tilde{y} \in \mathcal{Y}} G_{\tilde{y}}(x, z; \theta),$$

$$y_q = \arg\max_{\tilde{y} \in \mathcal{Y}} G_{\tilde{y}}(x, z; \theta) + \epsilon \Delta(\tilde{y}, \hat{y}').$$

First, let us rewrite $\mathbb{E}_{x \sim P_x(\boldsymbol{x})}[E_1 - E_3]$.

$$E_1 - E_3 = \mathbb{E}_{z \sim P_z(\boldsymbol{z})} [\mathbb{E}_{z' \sim P_z(\boldsymbol{z})} [\Delta(\hat{y}(x, z; \theta + \epsilon\delta\theta), \hat{y}(x, z'; \theta + \epsilon\delta\theta)) - \Delta(\hat{y}(x, z; \theta), \hat{y}(x, z'; \theta + \epsilon\delta\theta))]].$$

We denote by $\Delta_l^{j,i} = \Delta(y_j, y_l) - \Delta(y_i, y_l)$. This is the difference in losses encountered if, when $\theta \to \theta + \epsilon\delta\theta$, we move

- $\hat{y}(x, z; \theta) = y_i$ to $\hat{y}(x, z; \theta + \epsilon\delta\theta) = y_j$,
- while $\hat{y}(x, z'; \theta + \epsilon\delta\theta)$ is fixed at y_l.

We denote $V_{i,j,l}$ the corresponding subspace of $(\mathcal{X}, \mathcal{Z}, \mathcal{Z})$ for this move, expressed as follows,

$$V_{i,j,l} = \{(x, z, z') : \hat{y}(x, z; \theta + \epsilon\delta\theta) = y_j, \hat{y}(x, z; \theta) = y_i, \hat{y}(x, z'; \theta + \epsilon\delta\theta) = y_l\}.$$

This allows us to write $\mathbb{E}_{x \sim P_x(\boldsymbol{x})}[E_1 - E_3]$ in a simpler form:

$$\mathbb{E}_{x \sim P_x(\boldsymbol{x})}[E_1 - E_3] = \sum_{y_l \in \mathcal{Y}} \sum_{\substack{y_i, y_j \in \mathcal{Y} \\ i \neq j}} \Delta_l^{j,i} \mathbb{E}_{x \sim P_x(\boldsymbol{x})} \left[\mathbb{E}_{z \sim P_z(\boldsymbol{z})} \left[\mathbb{E}_{z' \sim P_z(\boldsymbol{z})} \left[1_{V_{i,j,l}} \right] \right] \right].$$

We rewrite $V_{i,j,l}$ in term of the grades set $G_{\mathcal{Y}}(x, z; \theta)$ and the loss Δ.

Part 4: Rewrite $V_{i,j,l}$**.**

We denote by $\delta G(x, z; \theta)^{i,j} = G_{y_i}(x, z; \theta) - G_{y_j}(x, z; \theta)$ the difference of grades between y_i and y_j, for the input pair (x, z) and parameters θ. If $\hat{y}(x, z; \theta) = y_i$, this means that y_i is the maximal graded output for the pair (x, z) and model parameters θ. We can write this as $\forall k \in 1, \ldots, n_y : \delta G(x, z; \theta)^{i,k} > 0$. Similarly $\hat{y}(x, z; \theta + \epsilon\delta\theta) = y_j$ means that y_j is the maximal graded output for the pair (x, z) and model parameters $\theta + \epsilon\delta\theta$. We can write this as $\forall k \in 1, \ldots, n_y : \delta G(x, z; \theta + \epsilon\delta\theta)^{j,k} > 0$. For clarity in the remainder of the proof we always denote $\forall k$ as a shorthand of $\forall k \in 1, \ldots, n_y$.
Therefore, we write the subspace $V_{i,j,l}$ as

$$\begin{aligned} V_{i,j,l} = \{(x, z, z') :& \forall k, \delta G(x, z; \theta + \epsilon\delta\theta)^{j,k} > 0, \\ & \forall k, \delta G(x, z; \theta)^{i,k} > 0, \\ & \hat{y}(x, z'; \theta + \epsilon\delta\theta) = y_l\} \end{aligned}$$

We decompose $\delta G(x, z; \theta + \epsilon\delta\theta)^{j,k}$ into its first order Taylor expansion, that is, $\delta G(x, z; \theta + \epsilon\delta\theta)^{j,k} = \delta G(x, z; \theta)^{j,k} + \epsilon\delta\theta^T \nabla_\theta \delta G(x, z; \theta)^{j,k} + o(\epsilon)$ and write the subspace $V_{i,j,l}$ as follows,

$$\begin{aligned} V_{i,j,l} = \{(x, z, z') :& \forall k \ \delta G(x, z; \theta)^{j,k} > -\epsilon\delta\theta^T \nabla_\theta \delta G(x, z; \theta)^{j,k} - o(\epsilon), \\ & \forall k, \delta G(x, z; \theta)^{i,k} > 0, \\ & \hat{y}(x, z'; \theta + \epsilon\delta\theta) = y_l\}. \end{aligned}$$

We explicitly separate the case $y_k = y_i$ and $y_k = y_j$.

$$\begin{aligned} V_{i,j,l} = \{(x, z, z') :& \forall k \neq i \ \delta G(x, z; \theta)^{j,k} > -\epsilon \delta\theta^T \nabla_\theta \delta G(x, z; \theta)^{j,k} - o(\epsilon), \\ & \forall k \neq j, \delta G(x, z; \theta)^{i,k} > 0, \\ & \delta G(x, z; \theta)^{j,i} > -\epsilon \delta\theta^T \nabla_\theta \delta G(x, z; \theta)^{j,i} - o(\epsilon), \qquad (7.7) \\ & \delta G(x, z; \theta)^{i,j} > 0, \qquad (7.8) \\ & \hat{y}(x, z'; \theta + \epsilon\delta\theta) = y_l\}. \end{aligned}$$

Noting that $\delta G(x, z; \theta)^{i,j} = -\delta G(x, z; \theta)^{j,i}$, we write equation (7.7) as $\delta G(x, z; \theta)^{i,j} < -\epsilon \delta\theta^T \nabla_\theta \delta G(x, z; \theta)^{i,j} + o(\epsilon)$, and equation (7.7) together with equation (7.8) become

$$0 < \delta G(x, z; \theta)^{i,j} < -\epsilon \delta\theta^T \nabla_\theta \delta G(x, z; \theta)^{i,j} + o(\epsilon).$$

The subspace $V_{i,j,l}$ is now[1]

$$\begin{aligned} V_{i,j,l} = \{(x, z, z') :& \forall k \neq i \ \delta G(x, z; \theta)^{j,k} > -\epsilon \delta\theta^T \nabla_\theta \delta G(x, z; \theta)^{j,k} + o(\epsilon), \\ & \forall k \neq j, \delta G(x, z; \theta)^{i,k} > 0, \\ & 0 < \delta G(x, z; \theta)^{i,j} < -\epsilon \delta\theta^T \nabla_\theta \delta G(x, z; \theta)^{i,j} + o(\epsilon), \\ & \hat{y}(x, z'; \theta + \epsilon\delta\theta) = y_l\} \end{aligned}$$

If $\hat{y}(x, z'; \theta + \epsilon\delta\theta) = y_l$, this means that y_l is the maximal graded output for the pair (x, z') and model parameters $\theta + \epsilon\delta\theta$, that is, $\forall k' \in \mathcal{Y} : \delta G(x, z'; \theta + \epsilon\delta\theta)^{l,k'} > 0$. We use the first order Taylor expansion of $\delta G(x, z'; \theta + \epsilon\delta\theta)^{l,k'}$ and the subspace $V_{i,j,l}$

[1]The classic notation o refers to a function of ϵ that is an infinitesimal of ϵ, that is, $\frac{o(\epsilon)}{\epsilon} \to 0$ when $\epsilon \to 0$. From now on, we will use positive signs in front of $o(\epsilon)$, as this notation refers to any function that is infinitesimal, therefore taking the negative is also infinitesimal.

is as follows,

$$\begin{aligned} V_{i,j,l} = \{(x,z,z') :& \forall k \neq i, \delta G(x,z;\theta)^{j,k} > -\epsilon\delta\theta^T\nabla_\theta\delta G(x,z;\theta)^{j,k} + o(\epsilon), \\ & \forall k \neq j, \delta G(x,z;\theta)^{i,k} > 0, \\ & 0 < \delta G(x,z;\theta)^{i,j} < -\epsilon\delta\theta^T\nabla_\theta\delta G(x,z;\theta)^{i,j} + o(\epsilon), \\ & \forall k', \delta G(x,z';\theta)^{l,k'} + \epsilon\delta\theta^T\nabla_\theta\delta G(x,z';\theta)^{l,k'} + o(\epsilon) > 0\}. \end{aligned}$$

For clarity, we use the shorthand notations:

$$\begin{aligned} a &= -\delta\theta^T\nabla_\theta\delta G(x,z;\theta)^{i,j}, \\ b_{j,k} &= -\delta\theta^T\nabla_\theta\delta G(x,z;\theta)^{j,k}, k \neq i, \\ c_{l,k'} &= -\delta\theta^T\nabla_\theta\delta G(x,z';\theta)^{l,k'}, \end{aligned}$$

and we write

$$\begin{aligned} V_{i,j,l} = \{(x,z,z') :& \forall k \neq i, \delta G(x,z;\theta)^{j,k} > \epsilon b_{j,k} + o(\epsilon), \\ & \forall k \neq j, \delta G(x,z;\theta)^{i,k} > 0, \\ & 0 < \delta G(x,z;\theta)^{i,j} < \epsilon a + o(\epsilon), \\ & \forall k', \delta G(x,z';\theta)^{l,k'} > \epsilon c_{l,k'} + o(\epsilon)\}. \end{aligned}$$

Let us sum up our findings so far, we have shown that:

$$\mathbb{E}_{x\sim P_x(\boldsymbol{x})}[E_1 - E_3] = \sum_{y_l\in\mathcal{Y}} \sum_{\substack{y_i,y_j\in\mathcal{Y} \\ i\neq j}} \Delta_l^{j,i} \mathbb{E}_{x\sim P_x(\boldsymbol{x})}\left[\mathbb{E}_{z\sim P_z(\boldsymbol{z})}\left[\mathbb{E}_{z'\sim P_z(\boldsymbol{z})}\left[1_{V_{i,j,l}}\right]\right]\right], \tag{7.9}$$

where

$$V_{i,j,l} = \{(x,z,z') :\forall k \neq i, \delta G(x,z;\theta)^{j,k} > \epsilon b_{j,k} + o(\epsilon),$$

$$\forall k \neq j, \delta G(x, z; \theta)^{i,k} > 0,$$
$$0 < \delta G(x, z; \theta)^{i,j} < \epsilon a + o(\epsilon),$$
$$\forall k', \delta G(x, z'; \theta)^{l,k'} > \epsilon c_{l,k'} + o(\epsilon)\}.$$

Part 5: Taking the limit.

Since $\boldsymbol{x}, \boldsymbol{z}$ are random variables, $\delta\theta^T \nabla_\theta \delta G(\boldsymbol{x}, \boldsymbol{z}; \theta)^{i,j}, i, j \in 1, \ldots, n_y$ is a random variable, which we denote $\delta\theta^T \nabla_\theta \delta G^{i,j}$ for clarity. The expectation with respect to $(\boldsymbol{x}, \boldsymbol{z})$ can be written with respect to $\delta\theta^T \nabla_\theta \delta G^{i,j}$ using the Law of the Unconscious Statistician (LOTUS) described in section 2.3.1. This is also the case for the terms $\delta G(\boldsymbol{x}, \boldsymbol{z}; \theta)^{i,j}$. Therefore, we consider the following random variables (recall $n_y = |\mathcal{Y}|$)[2].

- $\delta\theta^T \nabla_\theta \delta G^{i,1}, \ldots, \delta\theta^T \nabla_\theta \delta G^{j,n_y}$, i.e., $a, b_{i,1}, \ldots, b_{j,n_y}$,
- $\delta\theta^T \nabla_\theta \delta G^{l,1} \ldots; \delta\theta^T \nabla_\theta \delta G^{l,n_y}$, i.e., $c_{j,1}, \ldots, c_{j,n_y}$,
- $\delta G^{i,1}, \ldots, \delta G^{j,n_y}$,
- $\delta G^{l,1}, \ldots, \delta G^{l,n_y}$.

Similar to McAllester et al. (2010), we assume any joint distribution ρ on these random variables can be expressed as:

- A distribution μ on $a, b_{i,1}, \ldots, b_{j,n_y}, c_{j,1}, \ldots, c_{j,n_y}$,
- A conditional density function that we denote $f_{i,j,l}$ for clarity, as follows,

$$f_{i,j,l} := f(\delta G^{i,1}, \ldots, \delta G^{j,n_y}, \delta G^{l,1}, \ldots, \delta G^{l,n_y} | a, b_{i,1}, \ldots, b_{j,n_y}, c_{j,1}, \ldots, c_{j,n_y}).$$

[2]We do not denote the following random variables in bold for clarity purposes.

This allows us to write the expectation of the subspace $V_{i,j,l}$:

$$\mathbb{E}_\rho\big[1_{V_{i,j,l}}\big] = \mathbb{E}_\mu\Big[\int_0^{a\epsilon+o(\epsilon)}\int_{b_{j,1}\epsilon+o(\epsilon)}^{\infty}\dots\int_{b_{j,n_y}\epsilon+o(\epsilon)}^{\infty}\int_0^{\infty}\dots\int_0^{\infty}\int_{c_{l,1}\epsilon+o(\epsilon)}^{\infty}\dots\int_{c_{l,n_y}\epsilon+o(\epsilon)}^{\infty} f_{i,j,l}\mathrm{d}\delta G^{l,1},\dots,\mathrm{d}\delta G^{l,n_y}\mathrm{d}\delta G^{i,1},\dots,\mathrm{d}\delta G^{j,n_y}\Big].$$

Under the assumption that the range of the expectation with respect to μ is bounded, and that the integrand is continuous and bounded (detailed at the end of the proof), we exchange the limit and expectation sign and write

$$\begin{aligned}&\lim_{\epsilon\searrow 0}\frac{1}{\epsilon}\mathbb{E}_\rho\big[1_{V_{i,j,l}}\big]\\ &= \mathbb{E}_\mu\Big[\lim_{\epsilon\searrow 0}\frac{1}{\epsilon}\int_0^{a\epsilon+o(\epsilon)}\int_{b_{j,1}\epsilon+o(\epsilon)}^{\infty}\dots\int_{b_{j,n_y}\epsilon+o(\epsilon)}^{\infty}\int_0^{\infty}\dots\int_0^{\infty}\int_{c_{l,1}\epsilon+o(\epsilon)}^{\infty}\dots\int_{c_{l,n_y}\epsilon+o(\epsilon)}^{\infty} f_{i,j,l}\mathrm{d}\delta G^{l,1},\dots,\mathrm{d}\delta G^{l,n_y}\mathrm{d}\delta G^{i,1},\dots,\mathrm{d}\delta G^{j,n_y}\Big].\end{aligned}$$

At this point we use Lemma 2 from Song et al. (2016), which we repeat here.

Lemma 7.1 (Limit of multiple integrals)**.** *Suppose a continuous function $f(t_0,t_1,\dots,t_n)$, then for a scalar value $d_0>0$ and scalar values $d_1,\dots,d_n$:*

$$\begin{aligned}&\lim_{\epsilon\searrow 0}\frac{1}{\epsilon}\int_0^{d_0\epsilon+o(\epsilon)}\int_{d_1\epsilon+o(\epsilon)}^{\infty}\dots\int_{d_n\epsilon+o(\epsilon)}^{\infty} f(t_0,t_1,\dots,t_n)\mathrm{d}t_0\mathrm{d}t_1\dots\mathrm{d}t_n\\ &=\lim_{\epsilon\searrow 0}\frac{1}{\epsilon}d_0\int_0^{\epsilon}\int_0^{\infty}\dots\int_0^{\infty} f(t_0,t_1,\dots,t_n)\mathrm{d}t_0\mathrm{d}t_1\dots\mathrm{d}t_n.\end{aligned}$$

Proof. See Song et al. (2016) supplementary material. □

While Lemma 7.1 specifies $d_0>0$, Song et al. (2016) effectively use $\forall d_0\in\mathbb{R}$:

$$\begin{aligned}&\lim_{\epsilon\searrow 0}\frac{1}{\epsilon}\int_0^{d_0\epsilon+o(\epsilon)}\int_{d_1\epsilon+o(\epsilon)}^{\infty}\dots\int_{d_n\epsilon+o(\epsilon)}^{\infty} f(t_0,t_1,\dots,t_n)\mathrm{d}t_0\mathrm{d}t_1\dots\mathrm{d}t_n\\ &=\lim_{\epsilon\searrow 0}\frac{1}{\epsilon}(d_0)^+\int_0^{\epsilon}\int_0^{\infty}\dots\int_0^{\infty} f(t_0,t_1,\dots,t_n)\mathrm{d}t_0\mathrm{d}t_1\dots\mathrm{d}t_n.\end{aligned}$$

where $(d_0)^+ = \max(d_0, 0)$. Indeed, in the specific case where f is a density function over $(\boldsymbol{t}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_n)$ we can write:

$$\begin{aligned}&\lim_{\epsilon \searrow 0} \frac{1}{\epsilon} \int_0^{d_0\epsilon + o(\epsilon)} \int_{d_1\epsilon+o(\epsilon)}^{\infty} \ldots \int_{d_n\epsilon+o(\epsilon)}^{\infty} f(t_0, t_1, \ldots, t_n) \mathrm{d}t_0 \mathrm{d}t_1 \ldots \mathrm{d}t_n \\ &= \lim_{\epsilon \searrow 0} \frac{1}{\epsilon} \mathbb{E}\left[1_{\substack{t_0 \in [0, d_0\epsilon + o(\epsilon)], \\ t_i \in [d_i\epsilon + o(\epsilon), +\infty)], i>0}} \right].\end{aligned}$$

where the expectation is taken with respect to $(\boldsymbol{t}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_n)$. In the case $d_0 < 0$, the interval $[0, d_0\epsilon + o(\epsilon)]$ will be empty for ϵ sufficiently small, and the expectation will be 0. Lemma 1 in McAllester et al. (2010) states a similar result. With Lemma 7.1, we have

$$\begin{aligned}\lim_{\epsilon \searrow 0} \frac{1}{\epsilon} \mathbb{E}_\rho\left[1_{V_{i,j,l}}\right] = \mathbb{E}_\mu \Big[\lim_{\epsilon \searrow 0} \frac{1}{\epsilon} (a)^+ \int_0^{\epsilon} \int_0^{\infty} \ldots \int_0^{\infty} \int_0^{\infty} \ldots \int_0^{\infty} \int_0^{\infty} \ldots \int_0^{\infty} \\ f_{i,j,l} \mathrm{d}\delta G^{l,1}, \ldots, \mathrm{d}\delta G^{l,n_y} \mathrm{d}\delta G^{i,1}, \ldots, \mathrm{d}\delta G^{j,n_y} \Big].\end{aligned}$$

Note that when we use Lemma 7.1, $d_0, d_1, \ldots, d_n$ correspond to scalar values taken by the random variables $a, b_{i,1}, \ldots, b_{j,n_y}, c_{j,1}, \ldots, c_{j,n_y}$ as we are inside the expectation with respect to μ, while $\boldsymbol{t}_0$ is the random variable $\delta G^{i,j}$ and $\boldsymbol{t}_1, \ldots, \boldsymbol{t}_n$ are $\delta G^{i,k},\ \forall k \neq j$, $\delta G^{j,k},\ \forall k \neq i$, and $\delta G^{l,k},\ \forall k$.

We group expectations for a change from y_i to y_j, and from y_j to y_i, noting that the set $V_{j,i,l}$ corresponding to a change from y_j to y_i is written as:

$$\begin{aligned}V_{j,i,l} = \{(x, z, z') : &\forall k \neq j, \delta G(x, z; \theta)^{i,k} > \epsilon b_{i,k} + o(\epsilon), \\ &\forall k \neq i, \delta G(x, z; \theta)^{j,k} > 0, \\ &\delta G(x, z; \theta)^{j,i} > 0, \\ &\delta G(x, z; \theta)^{i,j} > a\epsilon + o(\epsilon), \\ &\forall k', \delta G(x, z'; \theta)^{l,k'} > \epsilon c_{l,k'} + o(\epsilon)\}, \\ = \{(x, z, z') : &\forall k \neq j, \delta G(x, z; \theta)^{i,k} > \epsilon b_{i,k} + o(\epsilon),\end{aligned}$$

$$\forall k \neq i, \delta G(x,z;\theta)^{j,k} > 0,$$
$$a\epsilon + o(\epsilon) < \delta G(x,z;\theta)^{i,j} < 0,$$
$$\forall k', \delta G(x,z';\theta)^{l,k'} > \epsilon c_{l,k'} + o(\epsilon)\},$$

Similar to Lemma 7.1 and Lemma 1 in McAllester et al. (2010) states, one can show that $\forall d_0 \in \mathbb{R}$, for a continuous bounded density function f,

$$\begin{aligned}&\lim_{\epsilon \searrow 0} \frac{1}{\epsilon} \int_{d_0\epsilon+o(\epsilon)}^{0} \int_{d_1\epsilon+o(\epsilon)}^{\infty} \dots \int_{d_n\epsilon+o(\epsilon)}^{\infty} f(t_0,t_1,\dots,t_n)\mathrm{d}t_0\mathrm{d}t_1\dots\mathrm{d}t_n \\ &= \lim_{\epsilon \searrow 0} \frac{1}{\epsilon} - (d_0)^{-} \int_0^{\epsilon} \int_0^{\infty} \dots \int_0^{\infty} f(t_0,t_1,\dots,t_n)\mathrm{d}t_0\mathrm{d}t_1\dots\mathrm{d}t_n.\end{aligned}$$

with $(d_0)^{-} = \min(d_0, 0)$. Thereby, we can write:

$$\begin{aligned}\lim_{\epsilon \searrow 0} \frac{1}{\epsilon} \mathbb{E}_\rho\big[1_{V_{j,i,l}}\big] = \mathbb{E}_\mu\Big[\lim_{\epsilon \searrow 0} \frac{1}{\epsilon}(-(a)^{-}) \int_0^{\epsilon} \int_0^{\infty} \dots \int_0^{\infty} \int_0^{\infty} \dots \int_0^{\infty} \int_0^{\infty} \dots \int_0^{\infty} \\ f_{i,j,l}\mathrm{d}\delta G^{l,1}, \dots, \mathrm{d}\delta G^{l,n_y}\mathrm{d}\delta G^{i,1}, \dots, \mathrm{d}\delta G^{j,n_y}\Big].\end{aligned}$$

We sum the two expressions, noting that $a = (a)^{+} + (a)^{-}$,

$$\begin{aligned}&\Delta_l^{j,i}\mathbb{E}_\mu\left[\lim_{\epsilon \searrow 0} \frac{1}{\epsilon} 1_{V_{i,j,l}}\right] + \Delta_l^{i,j}\mathbb{E}_\mu\left[\lim_{\epsilon \searrow 0} \frac{1}{\epsilon} 1_{V_{j,i,l}}\right] \\ &= \Delta_l^{j,i}\mathbb{E}_\mu\left[\lim_{\epsilon \searrow 0} \frac{1}{\epsilon} 1_{V_{i,j,l}}\right] - \Delta_l^{j,i}\mathbb{E}_\mu\left[\lim_{\epsilon \searrow 0} \frac{1}{\epsilon} 1_{V_{j,i,l}}\right] \\ &= \Delta_l^{j,i}\mathbb{E}_\mu\Big[\lim_{\epsilon \searrow 0} \frac{1}{\epsilon}((a)^{+} + (a)^{-}) \int_0^{\epsilon} \int_0^{\infty} \dots \int_0^{\infty} \int_0^{\infty} \dots \int_0^{\infty} \int_0^{\infty} \dots \int_0^{\infty} \\ &\quad f_{i,j,l}\mathrm{d}\delta G^{l,1}, \dots, \mathrm{d}\delta G^{l,n_y}\mathrm{d}\delta G^{i,1}, \dots, \mathrm{d}\delta G^{j,n_y}], \\ &= \Delta_l^{j,i}\mathbb{E}_\mu\Big[\lim_{\epsilon \searrow 0} \frac{1}{\epsilon} a \int_0^{\epsilon} \int_0^{\infty} \dots \int_0^{\infty} \int_0^{\infty} \dots \int_0^{\infty} \int_0^{\infty} \dots \int_0^{\infty} \\ &\quad f_{i,j,l}\mathrm{d}\delta G^{l,1}, \dots, \mathrm{d}\delta G^{l,n_y}\mathrm{d}\delta G^{i,1}, \dots, \mathrm{d}\delta G^{j,n_y}]. \end{aligned} \tag{7.10}$$

Using equation (7.9) and equation (7.10) together gives:

$$\lim_{\epsilon \searrow 0} \frac{\mathbb{E}_{x \sim P_x(\boldsymbol{x})}[E_1 - E_3]}{\epsilon} =$$
$$\sum_{y_l \in \mathcal{Y}} \sum_{\substack{y_i, y_j \in \mathcal{Y} \\ i \neq j}} \Delta_l^{j,i} \mathbb{E}_\mu \Big[\lim_{\epsilon \searrow 0} \frac{1}{\epsilon} a \int_0^\epsilon \int_0^\infty \dots \int_0^\infty \int_0^\infty \dots \int_0^\infty \int_0^\infty \dots \int_0^\infty$$
$$f_{i,j,l} \mathrm{d}\delta G^{l,1}, \dots, \mathrm{d}\delta G^{l,n_y} \mathrm{d}\delta G^{i,1}, \dots, \mathrm{d}\delta G^{j,n_y}]. \tag{7.11}$$

Recall that we want to prove:

$$\begin{aligned}
&\lim_{\epsilon \searrow 0} \frac{\mathbb{E}_{x \sim P_x(\boldsymbol{x})}[E_1 - E_3]}{\epsilon} = \\
&\quad \lim_{\epsilon \searrow 0} \frac{1}{\epsilon} \delta\theta^T \mathbb{E}_{x \sim P_x(\boldsymbol{x})} \Big[\mathbb{E}_{z \sim P_z(\boldsymbol{z})} [\mathbb{E}_{z' \sim P_z(\boldsymbol{z})} [\nabla_\theta G_{y_q}(x, z; \theta) \\
&\qquad - \nabla_\theta G_{\hat{y}}(x, z; \theta)]] \Big], \\
&\hat{y} = \arg\max_{\tilde{y} \in \mathcal{Y}} G_{\tilde{y}}(x, z; \theta), \\
&\hat{y}' = \arg\max_{\tilde{y} \in \mathcal{Y}} G_{\tilde{y}}(x, z'; \theta), \\
&y_q = \arg\max_{\tilde{y} \in \mathcal{Y}} G_{\tilde{y}}(x, z; \theta) + \epsilon \Delta(\tilde{y}, \hat{y}').
\end{aligned} \tag{7.12}$$

We will now show that the right hand side of equation (7.12) is also equal to the right hand side of equation (7.11). Let us denote the term in the limit in the right hand side of equation (7.12) as:

$$\mathrm{RHS} = \delta\theta^T \mathbb{E}_{x \sim P_x(\boldsymbol{x})} \Big[\mathbb{E}_{z \sim P_z(\boldsymbol{z})} [\mathbb{E}_{z' \sim P_z(\boldsymbol{z})} [\nabla_\theta G_{y_q}(x, z; \theta) - \nabla_\theta G_{\hat{y}}(x, z; \theta)]] \Big].$$

Part 6: Rewrite using subspace $U_{i,j,l}$**.**

First, note that $\nabla_\theta G_{y_j}(x, z; \theta) - \nabla_\theta G_{y_i}(x, z; \theta)$ is the difference of gradients between y_j and y_i, for the input pair (x, z) and parameters θ. It is equal to the gradient of the difference in scores, $\nabla_\theta \delta G(x, z; \theta)^{i,j}$,

that appeared in the previous parts when we used the Taylor expansion $\delta G(x,z;\theta+\epsilon\delta\theta)^{i,j} = \delta G(x,z;\theta)^{j,i} + \epsilon\delta\theta^T\nabla_\theta\delta G(x,z;\theta)^{j,i} + o(\epsilon)$.

This allows us to rewrite RHS as follows,

$$\begin{aligned}\text{RHS} &= \mathbb{E}_{x\sim P_x(\boldsymbol{x})}\Big[\mathbb{E}_{z\sim P_z(\boldsymbol{z})}\Big[\mathbb{E}_{z'\sim P_z(\boldsymbol{z})}\Big[\delta\theta^T\Big(\nabla_\theta G_{y_q(x,z;\theta)}(x,z;\theta) - \nabla_\theta G_{\hat{y}(x,z;\theta)}(x,z;\theta)\Big)\Big]\Big]\Big] \\ &= \sum_{y_l\in\mathcal{Y}}\sum_{\substack{y_i,y_j\in\mathcal{Y}\\ i\neq j}} \mathbb{E}_{x\sim P_x(\boldsymbol{x})}\big[\mathbb{E}_{z\sim P_z(\boldsymbol{z})}\big[\mathbb{E}_{z'\sim P_z(\boldsymbol{z})}\big[\delta\theta^T\delta\nabla_\theta G(x,z;\theta)^{j,i}1_{U_{i,j,l}}\big]\big]\big]\,. \qquad (7.13)\end{aligned}$$

where $U_{i,j,l} = \{(x,z,z') : y_q(x,z;\theta) = y_j, \hat{y}(x,z;\theta) = y_i, \hat{y}(x,z';\theta) = y_l\}$.

Recall that we defined $\Delta_l^{j,k} = \Delta(y_j,y_l) - \Delta(y_k,y_l)$. Using the same type of reasoning as for $V_{i,j,l}$, we have (i) $y_q(x,z;\theta) = y_j$ means $\forall k, \delta G(x,z;\theta)^{j,k} + \epsilon\Delta_l^{j,k} > 0$, (ii) $\hat{y}(x,z;\theta) = y_i$ means $\forall k, \delta G(x,z;\theta)^{i,k} > 0$ and (iii) $\hat{y}(x,z';\theta) = y_l$ means $\forall k, \delta G(x,zl;\theta)^{l,k} > 0$. Thereby, we rewrite $U_{i,j,l}$ as follows,

$$\begin{aligned}U_{i,j,l} = \{(x,z,z') :&\forall k\neq i, \delta G(x,z;\theta)^{j,k} > -\epsilon\Delta_l^{j,k},\\ &\forall k\neq j, \delta G(x,z;\theta)^{i,k} > 0,\\ &\delta G(x,z;\theta)^{j,i} > -\epsilon\Delta_l^{j,i},\\ &\delta G(x,z;\theta)^{i,j} > 0,\\ &\forall k', \delta G(x,z';\theta)^{l,k'} > 0\},\\ = \{(x,z,z') :&\forall k\neq i, \delta G(x,z;\theta)^{j,k} > -\epsilon\Delta_l^{j,k},\\ &\forall k\neq j, \delta G(x,z;\theta)^{i,k} > 0,\\ &\delta G(x,z;\theta)^{i,j} < \epsilon\Delta_l^{j,i},\\ &\delta G(x,z;\theta)^{i,j} > 0,\\ &\forall k', \delta G(x,z';\theta)^{l,k'} > 0\},\\ = \{(x,z,z') :&\forall k\neq i, \delta G(x,z;\theta)^{j,k} > -\epsilon\Delta_l^{j,k},\\ &\forall k\neq j, \delta G(x,z;\theta)^{i,k} > 0,\end{aligned}$$

$$0 < \delta G(x,z;\theta)^{i,j} < \epsilon\Delta_l^{j,i},$$
$$\forall k', \delta G(x,z';\theta)^{l,k'} > 0\}.$$

Again, we will consider the random variables

- $\delta\theta^T\nabla_\theta\delta G^{i,1},\dots,\delta\theta^T\nabla_\theta\delta G^{j,n_y}$, i.e., $a, b_{i,1},\dots,b_{j,n_y}$,
- $\delta\theta^T\nabla_\theta\delta G^{l,1}\dots;\delta\theta^T\nabla_\theta\delta G^{l,n_y}$, i.e., $c_{j,1},\dots,c_{j,n_y}$,
- $\delta G^{i,1},\dots,\delta G^{j,n_y}$,
- $\delta G^{l,1},\dots,\delta G^{l,n_y}$,

the distributions ρ,μ and the conditional density $f_{i,j,l}$.

$$\mathbb{E}_\rho\big[\delta\theta^T\nabla_\theta\delta G(x,z;\theta)^{j,i}1_{U_{i,j,l}}\big] = \mathbb{E}_\rho\big[\delta\theta^T(-\nabla_\theta\delta G(x,z;\theta)^{i,j})1_{U_{i,j,l}}\big]$$
$$= \mathbb{E}_\mu\Big[a\int_0^{\epsilon\Delta_l^{i,j}}\int_{-\epsilon\Delta_l^{j,1}}^{\infty}\dots\int_{-\epsilon\Delta_l^{j,n_y}}^{\infty}\int_0^\infty\dots\int_0^\infty\int_0^\infty\dots\int_0^\infty$$
$$f_{i,j,l}\mathrm{d}\delta G^{l,1},\dots,\mathrm{d}\delta G^{l,n_y}\mathrm{d}\delta G^{i,1},\dots,\mathrm{d}\delta G^{j,n_y}\Big].$$

recalling that $a = -\delta\theta^T\nabla_\theta\delta G(x,z;\theta)^{i,j}$. Note that while only $\delta\theta^T\nabla_\theta\delta G(x,z;\theta)^{i,j}$ appears in $\mathbb{E}_\rho\big[\delta\theta^T\nabla_\theta\delta G(x,z;\theta)^{i,j}1_{U_{i,j,l}}\big]$, we make $f_{i,j,l}$ appear and marginalise it over the extra random variables $\delta\theta^T\nabla_\theta\delta G$. Under the assumption that the range of the expectation with respect to μ is bounded, and that the integrand is continuous and bounded (detailed at the end of the proof), we exchange the limit and expectation sign and use Lemma 7.1 again. This gives us:

$$\mathbb{E}_\rho\big[\delta\theta^T\nabla_\theta\delta G(x,z;\theta)^{j,i}1_{U_{i,j,l}}\big] = \lim_{\epsilon\searrow 0}\frac{1}{\epsilon}\mathbb{E}_\rho\big[a1_{U_{i,j,l}}\big]$$
$$= \mathbb{E}_\mu\Big[a\lim_{\epsilon\searrow 0}\frac{1}{\epsilon}(\Delta_l^{j,i})^+\int_0^\epsilon\int_0^\infty\dots\int_0^\infty\int_0^\infty\dots\int_0^\infty\int_0^\infty\dots\int_0^\infty$$
$$f_{i,j,l}\mathrm{d}\delta G^{l,1},\dots,\mathrm{d}\delta G^{l,n_y}\mathrm{d}\delta G^{i,1},\dots,\mathrm{d}\delta G^{j,n_y}\Big].$$

A change from y_j to y_i is written as the following set:

$$\begin{aligned}
U_{j,l,l} = \{(x,z,z') :& \forall k \neq j, \delta G(x,z;\theta)^{i,k} > -\epsilon\Delta_l^{i,k}, \\
& \forall k \neq i, \delta G(x,z;\theta)^{j,k} > 0, \\
& 0 < \delta G(x,z;\theta)^{j,i} < \epsilon\Delta_l^{i,j}, \\
& \forall k', \delta G(x,z';\theta)^{l,k'} > 0\}, \\
= \{(x,z,z') :& \forall k \neq j, \delta G(x,z;\theta)^{i,k} > -\epsilon\Delta_l^{i,k}, \\
& \forall k \neq i, \delta G(x,z;\theta)^{i,k} > 0, \\
& 0 < \delta G(x,z;\theta)^{j,i} < -\epsilon\Delta_l^{j,i}, \\
& \forall k', \delta G(x,z';\theta)^{l,k'} > 0\}, \\
= \{(x,z,z') :& \forall k \neq j, \delta G(x,z;\theta)^{i,k} > -\epsilon\Delta_l^{i,k}, \\
& \forall k \neq i, \delta G(x,z;\theta)^{j,k} > 0, \\
& 0 > \delta G(x,z;\theta)^{i,j} > \epsilon\Delta_l^{j,i}, \\
& \forall k', \delta G(x,z';\theta)^{l,k'} > 0\}.
\end{aligned}$$

Thereby, we can write:

$$\begin{aligned}
&\mathbb{E}_\rho\big[\delta\theta^T\nabla_\theta\delta G(x,z;\theta)^{i,j}1_{U_{j,i,l}}\big] \\
&\quad= \mathbb{E}_\mu\Big[\delta\theta^T\nabla_\theta\delta G(x,z;\theta)^{i,j}\lim_{\epsilon\searrow 0}\frac{1}{\epsilon} - (\Delta_l^{j,i})^- \int_0^\epsilon\int_0^\infty\cdots\int_0^\infty\int_0^\infty\cdots\int_0^\infty\int_0^\infty\cdots\int_0^\infty \\
&\qquad f_{i,j,l}\mathrm{d}\delta G^{l,1},\ldots,\mathrm{d}\delta G^{l,n_y}\mathrm{d}\delta G^{i,1},\ldots,\mathrm{d}\delta G^{j,n_y}\Big], \\
&\quad= \mathbb{E}_\mu\Big[(-a)\lim_{\epsilon\searrow 0}\frac{1}{\epsilon} - (\Delta_l^{j,i})^- \int_0^\epsilon\int_0^\infty\cdots\int_0^\infty\int_0^\infty\cdots\int_0^\infty\int_0^\infty\cdots\int_0^\infty \\
&\qquad f_{i,j,l}\mathrm{d}\delta G^{l,1},\ldots,\mathrm{d}\delta G^{l,n_y}\mathrm{d}\delta G^{i,1},\ldots,\mathrm{d}\delta G^{j,n_y}\Big], \\
&\quad= \mathbb{E}_\mu\Big[a(\Delta_l^{j,i})^-\lim_{\epsilon\searrow 0}\frac{1}{\epsilon}\int_0^\epsilon\int_0^\infty\cdots\int_0^\infty\int_0^\infty\cdots\int_0^\infty\int_0^\infty\cdots\int_0^\infty \\
&\qquad f_{i,j,l}\mathrm{d}\delta G^{l,1},\ldots,\mathrm{d}\delta G^{l,n_y}\mathrm{d}\delta G^{i,1},\ldots,\mathrm{d}\delta G^{j,n_y}\Big].
\end{aligned}$$

When we group expectations for a change from y_i to y_j, and from y_j to y_i, we will use again $(\Delta_l^{j,i}) = (\Delta_l^{j,i})^+ + (\Delta_l^{j,i})^-$. Recall equation (7.13), that is, that we use the subspace $U_{i,j,l}$ to rewrite RHS. Taking the limit $\lim_{\epsilon \searrow 0} \frac{1}{\epsilon}$RHS gives,

$$\begin{aligned}\lim_{\epsilon \searrow 0} \frac{1}{\epsilon}\text{RHS} &= \sum_{y_l \in \mathcal{Y}} \sum_{\substack{y_i, y_j \in \mathcal{Y} \\ i \neq j}} \lim_{\epsilon \searrow 0} \frac{1}{\epsilon} \mathbb{E}_\rho \big[\delta\theta^T \nabla_\theta \delta G(x, z; \theta)^{j,i} 1_{U_{i,j,l}}\big] \\ &= \sum_{y_l \in \mathcal{Y}} \sum_{\substack{y_i, y_j \in \mathcal{Y} \\ i \neq j}} \mathbb{E}_\mu [\Delta_l^{j,i} a \lim_{\epsilon \searrow 0} \frac{1}{\epsilon} \int_0^\epsilon \int_0^\infty \cdots \int_0^\infty \int_0^\infty \cdots \int_0^\infty \int_0^\infty \cdots \int_0^\infty \\ &\qquad f_{i,j,l} \mathrm{d}\delta G^{l,1}, \ldots, \mathrm{d}\delta G^{l,n_y} \mathrm{d}\delta G^{i,1}, \ldots, \mathrm{d}\delta G^{j,n_y} \Big]. \end{aligned} \tag{7.14}$$

Equation (7.14) is the right hand side of equation (7.11), therefore both left hand sides are equals. We have shown that:

$$\begin{aligned}&\lim_{\epsilon \searrow 0} \frac{\mathbb{E}_{x \sim P_x(\boldsymbol{x})}[E_1 - E_3]}{\epsilon} = \\ &\quad \lim_{\epsilon \searrow 0} \frac{1}{\epsilon} \delta\theta^T \mathbb{E}_{x \sim P_x(\boldsymbol{x})} \Big[\mathbb{E}_{z \sim P_z(\boldsymbol{z})} [\mathbb{E}_{z' \sim P_z(\boldsymbol{z})} [\nabla_\theta G_{y_q}(x, z; \theta) \\ &\qquad - \nabla_\theta G_{\hat{y}}(x, z; \theta)]]\Big], \\ &\hat{y} = \arg\max_{\tilde{y} \in \mathcal{Y}} G_{\tilde{y}}(x, z; \theta), \\ &\hat{y}' = \arg\max_{\tilde{y} \in \mathcal{Y}} G_{\tilde{y}}(x, z'; \theta), \\ &y_q = \arg\max_{\tilde{y} \in \mathcal{Y}} G_{\tilde{y}}(x, z; \theta) + \epsilon \Delta(\tilde{y}, \hat{y}'). \end{aligned} \tag{7.15}$$

Part 7: Considering $\lim_{\epsilon \searrow 0} \frac{\mathbb{E}_{x \sim P_x(\boldsymbol{x})}[E_3 - E_2]}{\epsilon}$

Let us look at $\lim_{\epsilon \searrow 0} \frac{\mathbb{E}_{x \sim P_x(\boldsymbol{x})}[E_3 - E_2]}{\epsilon}$.

$$\begin{aligned}E_3 - E_2 = \mathbb{E}_{z \sim P_z(\boldsymbol{z})} [\mathbb{E}_{z' \sim P_z(\boldsymbol{z})} [&\Delta(\hat{y}(x, z; \theta), \hat{y}(x, z'; \theta + \epsilon\delta\theta)) \\ &- \Delta(\hat{y}(x, z; \theta), \hat{y}(x, z'; \theta))]].\end{aligned}$$

Using a reasoning similar to the one have derive for the term $\lim_{\epsilon \searrow 0} \frac{\mathbb{E}_{x \sim P_x(\boldsymbol{x})}[E_1 - E_3]}{\epsilon}$, one can show that:

$$\lim_{\epsilon \searrow 0} \frac{\mathbb{E}_{x \sim P_x(\boldsymbol{x})}[E_3 - E_2]}{\epsilon} = \lim_{\epsilon \searrow 0} \frac{1}{\epsilon} \delta\theta^T \mathbb{E}_{x \sim P_x(\boldsymbol{x})} \left[\mathbb{E}_{z \sim P_z(\boldsymbol{z})} \left[\mathbb{E}_{z' \sim P_z(\boldsymbol{z})} \left[\nabla_\theta G_{y'_q}(x, z'; \theta) - \nabla_\theta G_{\hat{y}'}(x, z'; \theta) \right] \right] \right], \tag{7.16}$$

where

$$\hat{y} = \arg\max_{\tilde{y} \in \mathcal{Y}} G_{\tilde{y}}(x, z; \theta),$$

$$\hat{y}' = \arg\max_{\tilde{y} \in \mathcal{Y}} G_{\tilde{y}}(x, z'; \theta),$$

$$y'_q = \arg\max_{\tilde{y} \in \mathcal{Y}} G_{\tilde{y}}(x, z'; \theta) + \epsilon \Delta(\tilde{y}, \hat{y}).$$

For this term, the proof would be more similar to the proof of Theorem 7.1 in Song et al. (2016), as the first argument of the loss, that is, $\hat{y}(x, z; \theta)$, does not change when $\epsilon \searrow 0$.

Part 8: Summing the two results.

We now sum together the two results obtained, that are equation (7.16) and equation (7.15). We repeat them here:

$$\lim_{\epsilon \searrow 0} \frac{\mathbb{E}_{x \sim P_x(\boldsymbol{x})}[E_1 - E_3]}{\epsilon} = \lim_{\epsilon \searrow 0} \frac{1}{\epsilon} \delta\theta^T \mathbb{E}_{x \sim P_x(\boldsymbol{x})} \Big[\mathbb{E}_{z \sim P_z(\boldsymbol{z})} [\mathbb{E}_{z' \sim P_z(\boldsymbol{z})} [\nabla_\theta G_{y_q}(x, z; \theta) - \nabla_\theta G_{\hat{y}}(x, z; \theta)]] \Big]$$

$$\lim_{\epsilon \searrow 0} \frac{\mathbb{E}_{x \sim P_x(\boldsymbol{x})}[E_3 - E_2]}{\epsilon} = \lim_{\epsilon \searrow 0} \frac{1}{\epsilon} \delta\theta^T \mathbb{E}_{x \sim P_x(\boldsymbol{x})} \Big[\mathbb{E}_{z \sim P_z(\boldsymbol{z})} \left[\mathbb{E}_{z' \sim P_z(\boldsymbol{z})} \left[\nabla_\theta G_{y'_q}(x, z'; \theta) - \nabla_\theta G_{\hat{y}'}(x, z'; \theta) \right] \right] \Big],$$

where

$$\hat{y} = \arg\max_{\tilde{y}\in\mathcal{Y}} G_{\tilde{y}}(x, z; \theta),$$

$$\hat{y}' = \arg\max_{\tilde{y}\in\mathcal{Y}} G_{\tilde{y}}(x, z'; \theta),$$

$$y_q = \arg\max_{\tilde{y}\in\mathcal{Y}} G_{\tilde{y}}(x, z; \theta) + \epsilon\Delta(\tilde{y}, \hat{y}'),$$

$$y'_q = \arg\max_{\tilde{y}\in\mathcal{Y}} G_{\tilde{y}}(x, z'; \theta) + \epsilon\Delta(\tilde{y}, \hat{y}).$$

We assume that $\lim_{\epsilon\searrow 0} \frac{\mathbb{E}_{x\sim P_x(\boldsymbol{x})}[E_1 - E_3]}{\epsilon}$ and $\lim_{\epsilon\searrow 0} \frac{\mathbb{E}_{x\sim P_x(\boldsymbol{x})}[E_3 - E_2]}{\epsilon}$ are finite. We note that z and z' play a symmetric role, that is,

$$\lim_{\epsilon\searrow 0} \frac{\mathbb{E}_{x\sim P_x(\boldsymbol{x})}[E_1 - E_3]}{\epsilon} = \lim_{\epsilon\searrow 0} \frac{\mathbb{E}_{x\sim P_x(\boldsymbol{x})}[E_3 - E_2]}{\epsilon}.$$

This gives us our final result:

$$\begin{aligned}
\lim_{\epsilon\searrow 0} \frac{\mathbb{E}_{x\sim P_x(\boldsymbol{x})}[E_1 - E_2]}{\epsilon} &= \lim_{\epsilon\searrow 0} \frac{\mathbb{E}_{x\sim P_x(\boldsymbol{x})}[E_1 - E_3 + E_3 - E_2]}{\epsilon} \\
&= \lim_{\epsilon\searrow 0} \frac{\mathbb{E}_{x\sim P_x(\boldsymbol{x})}[E_1 - E_3]}{\epsilon} + \lim_{\epsilon\searrow 0} \frac{\mathbb{E}_{x\sim P_x(\boldsymbol{x})}[E_3 - E_2]}{\epsilon} \\
&= 2 \lim_{\epsilon\searrow 0} \frac{\mathbb{E}_{x\sim P_x(\boldsymbol{x})}[E_1 - E_3]}{\epsilon}.
\end{aligned}$$

If we define:

$$\nabla_\theta \mathrm{DIV}^\epsilon_\Delta(Q, Q, P_x) = 2\mathbb{E}_{x\sim P_x(\boldsymbol{x})}\left[\mathbb{E}_{z\sim P_z(\boldsymbol{z})}\left[\mathbb{E}_{z'\sim P_z(\boldsymbol{z})}\left[\nabla_\theta G_{y_q}(x, z; \theta) - \nabla_\theta G_{\hat{y}}(x, z; \theta)\right]\right]\right],$$

we have

$$\lim_{\epsilon\searrow 0} \frac{\mathbb{E}_{x\sim P_x(\boldsymbol{x})}[E_1 - E_2]}{\epsilon} = \delta\theta^T \lim_{\epsilon\searrow 0} \nabla_\theta \mathrm{DIV}^\epsilon_\Delta(Q, Q, P_x).$$

As explained in equation (7.3) to equation (7.6), we defined E_1, E_2 such that,

$$\delta\theta^T \nabla_\theta \mathrm{DIV}_\Delta(Q, Q, P_x) = \lim_{\epsilon \searrow 0} \frac{\mathbb{E}_{x \sim P_x(\boldsymbol{x})}[E_1 - E_2]}{\epsilon}.$$

This leads us to

$$\lim_{\epsilon \searrow 0} \frac{1}{\epsilon} \delta\theta^T \nabla_\theta \mathrm{DIV}^\epsilon_\Delta(Q, Q, P_x) = \delta\theta^T \nabla_\theta \mathrm{DIV}_\Delta(Q, Q, P_x),$$

which is what we wanted to prove.

Part 9: Conditions of applications of the Loss gradient theorem for DISCO Nets.

Our proof of the extension of the Loss gradient theorem to the DISCO Nets' setting requires similar assumptions as McAllester et al. (2010); Song et al. (2016). First, we required that the joint density ρ can be expressed as a bounded conditional density function f and a distribution μ. Second, to assume that we can exchange limits and expectations, the integrand must be continuous and bounded. Therefore the integral of the conditional density f must be bounded and continuous with respect to the terms $\delta\theta^T \nabla_\theta \delta G$, and the grades set and their gradient with respect to θ must be bounded and continuous with respect to x, z. The range of the integral must be bounded, which again requires the gradient of the grades set to be bounded. In practise, we parametrise the grades set with a neural network. Depending on the network architecture, the gradients of the grades set might not be continuous, or even exist (in this case we compute subgradients). In this case we still use our approximation to learn the model's parameters.

Part 10: Convergence of the Loss gradient theorem.

We do not derive an asymptotic convergence guarantee for the Loss gradient theorem for DISCO Nets. Note that our objective function is non-convex with re-

spect to the model's parameters. McAllester et al. (2010) mentions that taking steps updates with a learning rate η_t and approximation parameter ϵ_t, one should be able to prove that the original Direct Loss Minimisation method (McAllester et al., 2010) converges to a local optimum of the objective function provided that $\eta_t \to 0$, $\epsilon_t \to 0$, $\sum_{t=0}^{\infty} \eta_t \epsilon_t \to 0$.

□

7.5.2 Hand pose estimation

We provide in this section additional details on the experiment on hand pose estimation of section 4.5.

Data preprocessing. We perform the same data preprocessing as in Oberweger et al. (2015a,b), using the DeepPrior software (Oberweger, 2015). Specifically, we extract a fixed-size metric cube around the hand from the depth image. We resize the depth values within the cube to a 128×128 patch and normalised them in $[-1, 1]$. Pixels deeper than the back of the cube and missing depth values are both set to a depth of 1.

MEU method. The prediction is made as follows,

$$y_{\text{MEU}} = \underset{k=1,\dots,K}{\arg\min} \sum_{k'=1}^{K} \Delta_{\text{metric}}(y_k, y_{k'})$$

For example, for a given input x, we need to choose a pointwise output to evaluate the Mean Joint Euclidean Error (MeJEE). We sample K candidate ouputs values for the input x, and elect:

$$y_{\text{MEU-MeJEE}} = \underset{k=1,\dots,K}{\arg\min} \sum_{k'=1}^{K} \Delta_{\text{MeJEE}}(y_k, y'_k) = \underset{k=1,\dots,K}{\arg\min} \sum_{k=1}^{K} \frac{1}{J} \sum_{j=1}^{J} \|y_k^j - {y'_k}^j\|_2$$

where y^j denotes the 3-dimensional vector of values corresponding to the 3D coordinates of the joint j (among the $3 \times J = 42$ values of all coordinates y). Then when we evaluate MeJEE, the loss encountered on the example x is $\Delta_{\text{MeJEE}}(y, y_{\text{MEU-MeJEE}})$ where y is the ground-truth output that corresponds to x.

Pointwise evaluation metrics. We denote by n_{test} the number of test example pairs (x_i, y_i), and $y_{\Delta_{\text{metric}},i}$ is the prediction, specific to the metric, for the i^{th} input example x_i. J is the number of joints of the hand.

MeJEE (Mean Joint Euclidean Error) is the per-joint Euclidean distance between the pointwise pose and the goundtruth, averaged by J and n_{test}, computed as:

$$\Delta_{\text{MeJEE}} = \frac{1}{J} \sum_{j=1}^{J} \|.\|_2,$$
$$\text{MeJEE} = \frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} \frac{1}{J} \sum_{j=1}^{J} \|y_i^j - y^j_{\Delta_{\text{MeJEE}},i}\|_2.$$

MaJEE (Max Joint Euclidean Error) is the per-joint maximal Euclidean distance between the pointwise pose and the goundtruth averaged by n_{test}, computed as:

$$\Delta_{\text{MaJEE}} = \max_{j \in [1,J]} \|.\|_2,$$
$$\text{MaJEE} = \frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} \max_{j \in [1,J]} \|y_i^j - y^j_{\Delta_{\text{MaJEE}},i}\|_2.$$

FF(d) (Fraction of Frames) is the fraction of test examples that have all predicted joints of the pointwise pose below a given maximum Euclidean distance d in mm from the ground-truth. Note the $-$ sign since for FF higher is better, the MEU method will take the output with maximal utility.

$$\Delta_{\text{FF}} = -1_{\max_{j \in [1,J]} \|.\|_2 \leq d},$$

$$\mathrm{FF}(d) = \frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} 1_{\max_{j\in[1,J]} \|y_i^j - y_{\Delta_{\mathrm{FF}},i}^j\|_2 \leq d}$$

Architecture. The input depth image x is fed to 2 convolutional layers, each having 8 filters, with kernels of size 5×5, with stride 1, followed by Rectified Linear Units (ReLUs) and Max Pooling layers of kernel size 3×3 and stride 3. A third and last convolutional layer has 8 filters, with kernels of size 5×5, with stride 1, followed by a Rectified Linear Unit. The ouput of the convolution is concatenated to the random noise vector z of size $d_z = 200$, drawn from a uniform distribution in $[-1, 1]$. The result of the concatenation is fed to 2 dense layers of output size 1024, with ReLUs, and a third dense layer that outputs the candidate pose $y \in \mathbb{R}^{3\times J}$. For the non-probabilistic BASE_σ model no noise is concatenated as only a pointwise estimate is produced.

Training details. Back-propagation is used with Stochastic Gradient Descent with a batchsize of 256. The learning rate is fixed to $\lambda = 0.01$ and we use a momentum of $m = 0.9$ (Polyak, 1964). We also add L2-regularisation controlled by the parameter C. We use $C = [0.0001, 0.001]$. Note that DISCO Nets report consistent performances across the different values C, contrary to BASE. We use 3 different random seeds to initialize each model network parameters. We report the performance of each model with its best cross-validated seed and C. We train all models for 400 epochs, as it results in a change of less than 3% in the value of the loss on the validation data set for BASE.

All network weights are initialised at random with a Gaussian distribution of mean 0 and standard deviation 0.01, all biases are initialised to 0. During training, the number of candidates outputs generated by the probabilistic models DISCO_γ is $K = 2$. This is sufficient to compute a value of the unbiased estimator of the gradient in step 5 of Algorithm 2. Note that in step 2 of Algorithm 2 we must

draw new noise samples $(z_{i,1}, \ldots, z_{i,K})$ for each training example at each iteration. Indeed, let us consider the iteration t on a training example x_i. The values of the noise sampled for x_i at iteration $t-1$, are used in the estimation of the gradient, and thus influence the update $\theta^t \leftarrow \theta^{t-1}$. Thus, we need to draw new samples to ensure that the K candidate outputs for x_i, $G(x_i, z_{i,k}; \theta_t), k = 1, \ldots, K$, remain independent given x_i and θ^t.

7.5.3 Psychology attributes ratings prediction

We provide in this section additional details on the experiment on psychology attributes ratings prediction of section 4.7.

Data preprocessing. The US10k Face Data Set (Bainbridge et al., 2013) consists of 20 psychological attributes all paired with their antonyms, we list them in Table 7.10. To obtain ratings on the face images from Amazon Mechanical Turk workers, the set of attributes and their antonyms was separated in two splits, split_1 and split_2, and ratings were gathered using two survey version. Each image is annotated for split_1 by 15 workers, and for split_2 by 15 other workers. There were $1,274$ workers altogether. Ratings range from 1 (not at all) to 9 (extremely). Each question included a pop-up window with a definition of the attribute, and the survey included a catch question to eliminate workers who were answering at random. In our setting, for a given face image x, we want the same worker to have annotated the whole set of considered attributes y. Therefore, contrary to (Bainbridge et al., 2013), we do not merge the two surveys and use split_1, which is composed of the attributes in bold in Table 7.10. We remove annotations where the catch question was incorrectly answered, and consider the minimum number of annotations per image available after this preprocessing, that is 12 annotations for each of the $2,222$ images. We split the data set into $1,022$ images for training, 600 for vali-

dation and 600 for testing. The face photographs are JPEGs with 256-pixel height. We resize the images to 227×227 pixels in order to suit the network architecture and normalise the images by subtracting the per-pixel mean and dividing by the per-pixel standard deviation, computed on the training set.

interesting/**boring**	**calm**/agressive
caring/**cold**	**confident**/uncertain
humble/**egotistic**	emotionally stable/**unstable**
intelligent/unintelligent	sociable/**introverted**
kind/mean	**responsible**/irresponsible
trustworthy/untrustworthy	attractive/**unattractive**
happy/**unhappy**	normal/**weird**
typical/**atypical**	memorable/**forgettable**
familiar/**unfamiliar**	**common**/uncommon
emotional/**unemotional**	friendly/**unfriendly**

Table 7.10: Psychology attributes from Bainbridge et al. (2013). The attributes in bold are the one we consider.

Architecture. The input depth image x is fed to a convolutional layer with 30 filters, with kernel of size 11×11 and stride 4, followed by Local Response Normalisation (LNR), RELUs and Max Pooling layers of kernel size 3×3 and stride 2. The resulting feature map is fed to a second convolutional layer with 40 filters, with kernel of size 5×5 and stride 1, followed by LNR, RELUs and Max Pooling layers of kernel size kernel size 3×3 and stride 2. The resulting eature map is fed to a third convolutional layer with 40 filters, with kernel of size 3×3 and stride 1, followed by RELUs units. This layer outputs a matrix of features flattened along the last dimension into a vector noted $f \in \mathbb{R}^{3240}$. The output f is concatenated to the random noise vector z of size $d_z = 200$, drawn from a uniform distribution in $[-1, 1]$. The result of the concatenation is fed to a dense layer of output size 1240, followed by RELUs units, and a second dense layer output size 180. The output is reshaped into $\mathbb{R}^{20\times9}$. Finally, we use a softmax activation function on the last dimension (corresponding to the 9 possible values of an attribute grade).

Training details. We update the model parameters either with Stochastic Gradient Descent (SGD) with learning rates $\lambda \in [0.001, 0.01]$ or SGD with momentum $\rho = 0.95$ learning rate $\lambda = 0.001$ (best performing method depends on the models, but performances are consistent). Interestingly, we found that models perform better with a mini-batch size of 10. Finally, we use a fixed value of $\epsilon = 1$ and the negative update direction. During training, the number of candidates outputs generated by all models is $K = 2$. For $\text{DISCO}_{\gamma=1/2}$, we compute the unbiased estimators in steps 4 and 5 of Algorithm 3 in the following manner. At each iteration, for each input x_i, among the 12 samples of the training data set, we randomly sample 1 ground-truth output. We draw from the model $K = 2$ samples. We randomly sample one sample among the 2 generated samples to be compared to the ground-truth output in step 4 of Algorithm 3 and compare the 2 samples in step 5. For $\text{DISCO}_{\gamma=0}$, one randomly drawn sample is compared to the ground-truth sample. For CNN-CE, the two samples drawn are used to compute the marginalised distribution.

Pseudo-polynomial time algorithm for exact loss-augmented inference. Steps 4 and 5 of Algorithm 3 requires to solve, for each sample x_i, $K + K(K-1)$ loss-augmented inference problems. Using dynamic programming methods, we can find exact solutions to the loss-augmented problems in pseudo-polynomial time.

First, let us restate the MinMax kernel between two outputs $\boldsymbol{y}, \boldsymbol{y'} \in \mathcal{Y} = \{0, ..., R\}^A$. If we consider range valued outputs $y \in \mathcal{Y} = \{0, ..., R\}^A$, that is, $y = (y^1, \ldots, y^A), y^1, \ldots, y^A \in \{0, \ldots, R\}$, the MinMax kernel between two outputs $y, y' \in \mathcal{Y} \times \mathcal{Y}$ is

$$\mathrm{K}_{\text{MinMax}}(y, y') = \frac{\sum_{a=1}^{A} \min(y^a, y'^a)}{\sum_{a=1}^{A} \max(y^a, y'^a)} \tag{7.17}$$

The loss-augmented inference problem that we have to solve are of the following form (in the positive update direction, the algorithm is similar in the negative

update direction),

$$\arg\max_{\tilde{y}\in\mathcal{Y}} G_{\tilde{y}}(x, z, \theta) + \epsilon\Delta(\tilde{y}, y). \tag{7.18}$$

Replacing the loss with its expression, $\Delta(y, y') = 1 - \mathrm{K}_{\mathrm{MinMax}}(y, y')$, equation (7.18) is equivalent to:

$$\arg\max_{\tilde{y}\in\mathcal{Y}} G_{\tilde{y}}(x, z, \theta) - \epsilon\mathrm{K}_{\mathrm{MinMax}}(\tilde{y}, y). \tag{7.19}$$

First, we notice that when the denominator in equation (7.17) (which sums over the maximum on each dimension a) is a constant, the loss-augmented inference of equation (7.19) can be decomposed over the dimensions. Second, the value of the denominator is lower bounded as follows,

$$\sum_{a=1}^{A} \max(y^a, y'^a) \geq \sum_{a=1}^{A} y^a = C,$$

where denote $C = \sum_{a=1}^{A} y^a$. Similarly, the denominator is upper bounded as follows,

$$\sum_{a=1}^{A} \max(y^a, y'^a) \leq \sum_{a=1}^{A} R = A \times R.$$

Let us consider $c = \sum_{a=1}^{A} \max(y^a, y'^a) \in \{C, ..., R \times A\}$ and fix the denominator to remain equal to c. Therefore, there is a total "budget", denoted $b = c - C$ to assign among each element of $\tilde{y}$. This is the commonly known Knapsack Problem, which can be solved in pseudo polynomial time. Using dynamic programming methods, we compute the possible assigment of b that maximises $G_{\tilde{y}}(x, z, \theta) - \epsilon\mathrm{K}_{\mathrm{MinMax}}(\tilde{y}, y)$, there are A attributes and the budget is bounded by $R \times A$ so this runs in time $\mathrm{O}\big(RA^2\big)$. We loop over the values of c, which is bounded by $R \times A$. This gives an exact solution to the loss-augmented problems in time $\mathrm{O}\big(R^2A^3\big)$.

7.5.4 Discrimination ability of the Negative kernel score

Pinson and Tastu (2013) refer to the discrimination ability of a scoring rule as its ability to evaluate the differences between two densities. Specifically, they define the discrimination ability of a scoring rule as follows (for positively oriented scoring rules),

Definition 7.8 (Discrimination ability of a scoring rule)**.** *The discrimination ability of a scoring rule S is the property such that if Q_1 is genuinely of higher quality than Q_2, then $S(Q_1, y) > S(Q_2, y)$ for any observation $y \sim P(\boldsymbol{y})$. The scoring rule is said to have high discrimination ability if a differences in the prediction quality of the modelling distribution Q_1 and Q_2 results in a significant difference between $S(Q_1, y)$ and $S(Q_2, y)$. Contrarily, if the difference between $S(Q_1, y)$ and $S(Q_2, y)$ is null, the scoring rule is said to have no discrimination ability.*

Intuitively, a high discrimination ability is a desirable property for a scoring rule both in the context of parameter estimation and model evaluation. As noted in Pinson and Tastu (2013), being strictly proper does not ensure that $S(Q_1, \mathcal{D}) > S(Q_2, \mathcal{D})$ for any dataset $\mathcal{D}$ drawn from P. However, we could expect strictly proper scoring rules to discriminate forecasting distributions that are close to the true distribution, as the divergence approaches its unique minimum. In this context, Pinson and Tastu (2013) investigate the discrimination ability of the Energy score with $\beta = 1$, corresponding to the Euclidean norm. Their conclusions show that the Energy score discriminates well distributions that differ in their means, but difference in scores values are much smaller for distributions that differ in their variance parameters. Importantly, the Energy Score hardly discriminates distributions that differ in their structure, that is, their covariance parameters.

In order to analyse the results of our psychology attribute ratings prediction experiment, we perform a similar analysis on the Negative kernel score with dis-

crete data. Similar to Pinson and Tastu (2013), we assume that there is a true process that follows the true distribution $P(\boldsymbol{y})$, which we access via 1000 samples drawn from P. Specifically, $\boldsymbol{y}$ is a two-dimensional variable and P is a bivariate distribution. We consider two forecasting models (i) a model that incorrectly captures the expected value of the process, denoted Q_{mean}; and (ii) a model that incorrectly captures the covariance between the two dimensions of $\boldsymbol{y}$, denoted Q_{cov}. We draw 1000 samples from Q_{mean} and Q_{cov} and compare their scores with respect to the samples drawn from P. Our goal is to evaluate how each type of error (mean or covariance) affects the Negative kernel score. To do so, we consider the relative change RC in the Negative kernel score:

$$\text{RC} = \frac{S(P,P) - S(Q,P)}{S(P,P)}.$$

where S is the Negative kernel scoring rule associated with the MinMax kernel, and $S(Q,P)$ is the associated expected Negative kernel score, that is,

$$\begin{aligned} S(P,P) &= \frac{1}{2}\mathbb{E}_{y\sim P(\boldsymbol{y})}\left[\mathbb{E}_{y'\sim P(\boldsymbol{y})}[\text{K}_{\text{MinMax}}(y,y')]\right], \\ S(Q,P) &= \mathbb{E}_{y\sim P(\boldsymbol{y})}\left[\mathbb{E}_{y'\sim Q(\boldsymbol{y})}[\text{K}_{\text{MinMax}}(y,y')]\right] \\ &\quad - \frac{1}{2}\mathbb{E}_{y'\sim Q(\boldsymbol{y})}\left[\mathbb{E}_{y''\sim Q(\boldsymbol{y})}[\text{K}_{\text{MinMax}}(y',y'')]\right]. \end{aligned}$$

Experimental setting. We perform our analysis on distributions of $\boldsymbol{y}$ and not conditioned on an input. We employ multivariate Poisson distribution, as this discrete distribution has the appealing following properties (Johnson et al., 1997).

Remark 7.5 (Properties of the multivariate Poisson distribution)**.** *Let $\boldsymbol{y}_i \equiv$ Poisson(λ_i), $\forall i \in 0,\ldots,m$. Define the random variables $\boldsymbol{w}_j = \boldsymbol{y}_j + \boldsymbol{y}_0$, $\forall j = 1,\ldots,m$, then $(\boldsymbol{w}_1,\ldots,\boldsymbol{w}_m)$ jointly follows a multivariate Poisson distribution. Furthermore, the marginal distribution of $\boldsymbol{w}_j, \forall j = 1,\ldots,m$*

is a Poisson distribution of parameter $\lambda_j + \lambda_0$*, that is,*

$$\boldsymbol{w}_j \equiv Poisson(\lambda_j + \lambda_0). \tag{7.20}$$

The covariance between each pair of random variable is λ_0*, in other words,*

$$cov(\boldsymbol{w}_j, \boldsymbol{w}_k) = \lambda_0, \forall j = 1, \ldots, m, k = 1, \ldots, m, k \neq j. \tag{7.21}$$

We construct the true distribution $P(\boldsymbol{w}_1, \boldsymbol{w}_2)$ to be a bivariate Poisson distribution of parameters $(\lambda_0, \lambda_1 = \lambda, \lambda_2 = \lambda)$, specifically,

$$\begin{aligned}
&\boldsymbol{y}_0 \equiv \text{Poisson}(\lambda_0), && \boldsymbol{w}_1 \equiv \text{Poisson}(\lambda + \lambda_0),\\
&\boldsymbol{y}_1 \equiv \text{Poisson}(\lambda), && \boldsymbol{w}_2 \equiv \text{Poisson}(\lambda + \lambda_0),\\
&\boldsymbol{y}_2 \equiv \text{Poisson}(\lambda), && \text{cov}(\boldsymbol{w}_1, \boldsymbol{w}_2) = \lambda_0.\\
&\boldsymbol{w}_1 = \boldsymbol{y}_1 + \boldsymbol{y}_0, &&\\
&\boldsymbol{w}_2 = \boldsymbol{y}_2 + \boldsymbol{y}_0, &&
\end{aligned}$$

We consider two models, respectively represented by the distributions $Q_{\epsilon_{\text{mean}}}$ and $Q_{\epsilon_{\text{cov}}}$. The first forecasting distribution $Q_{\epsilon_{\text{mean}}}(\boldsymbol{w}_1, \boldsymbol{w}_2)$ does not capture the correct marginal means of the distributions, but accurately captures the variance and covariance parameters. The resulting $Q_{\epsilon_{\text{mean}}}$ is no longer a bivariate Poisson distribution, since the Poisson distribution family is not closed under linear trans-

formation (Winkelmann, 2008).

$$\begin{aligned}
&\boldsymbol{y}_0 \equiv \text{Poisson}(\lambda_0), && \mathbb{E}[\boldsymbol{w}_1] = (\lambda + \lambda_0) - \epsilon_{\text{mean}}(\lambda + \lambda_0),\\
&\boldsymbol{y}_1 \equiv \text{Poisson}(\lambda), && \mathbb{E}[\boldsymbol{w}_2] = (\lambda + \lambda_0) - \epsilon_{\text{mean}}(\lambda + \lambda_0),\\
&\boldsymbol{y}_2 \equiv \text{Poisson}(\lambda), && \mathbb{V}[\boldsymbol{w}_1] = \lambda + \lambda_0,\\
&\boldsymbol{w}_1 = \boldsymbol{y}_1 + \boldsymbol{y}_0 - \epsilon_{\text{mean}}(\lambda + \lambda_0), && \mathbb{V}[\boldsymbol{w}_2] = \lambda + \lambda_0,\\
&\boldsymbol{w}_2 = \boldsymbol{y}_2 + \boldsymbol{y}_0 - \epsilon_{\text{mean}}(\lambda + \lambda_0), && \text{cov}(\boldsymbol{w}_1, \boldsymbol{w}_2) = \lambda_0.
\end{aligned}$$

While it is true that the distribution $Q_{\epsilon_{\text{mean}}}$ suffers from the fact that the resulting marginals are not Poisson distributions, for large values of λ the Poisson distribution approximates the Normal distribution of mean and variance equal to λ. This is a known result and the proof relies on the Central Limit theorem. As the Normal distribution is closed under linear transformation, we expect this disadvantaging effect on $Q_{\epsilon_{\text{mean}}}$ to decrease as λ increases.

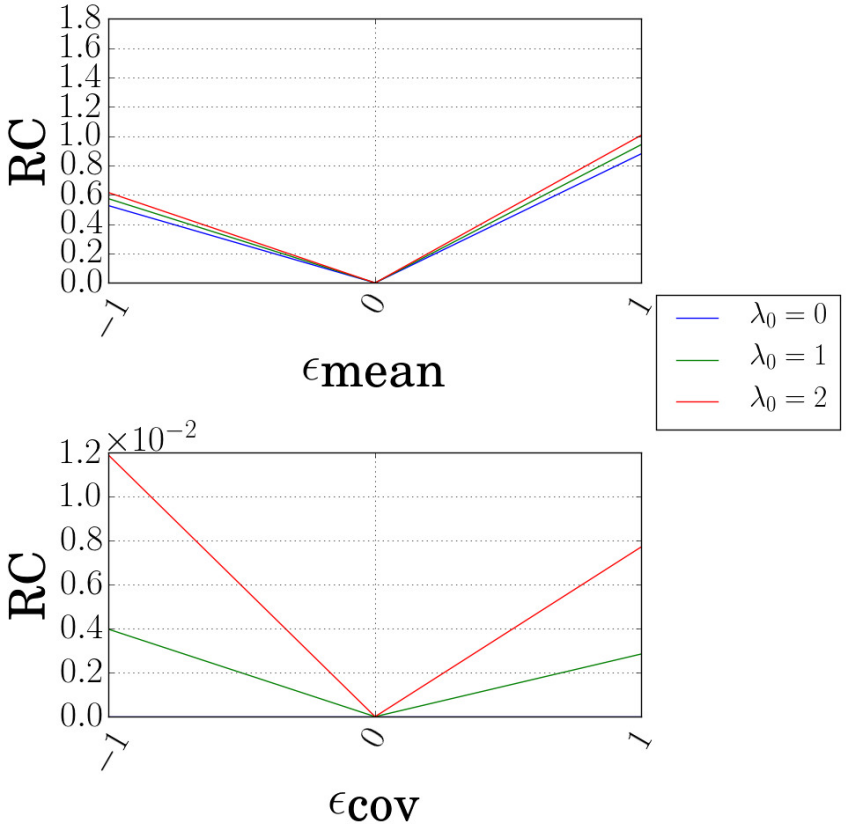
The second forecasting distribution $Q_{\epsilon_{\text{cov}}}(\boldsymbol{w}_1, \boldsymbol{w}_2)$ incorrectly models the covariance, that is, λ_0. We increase the parameter λ so that the marginals follow the same Poisson distributions as the true distribution. The resulting $Q_{\epsilon_{\text{cov}}}$ remains a bivariate Poisson distribution, with Poisson marginals, that is,

$$\begin{aligned}
&\boldsymbol{y}_0 \equiv \text{Poisson}(\lambda_0 - \epsilon_{\text{cov}}\lambda_0), && \boldsymbol{w}_1 \equiv \text{Poisson}(\lambda + \lambda_0),\\
&\boldsymbol{y}_1 \equiv \text{Poisson}(\lambda + \epsilon_{\text{cov}}\lambda_0), && \boldsymbol{w}_2 \equiv \text{Poisson}(\lambda + \lambda_0),\\
&\boldsymbol{y}_2 \equiv \text{Poisson}(\lambda + \epsilon_{\text{cov}}\lambda_0), && \text{cov}(\boldsymbol{w}_1, \boldsymbol{w}_2) = \lambda_0 - \epsilon_{\text{cov}}\lambda_0.\\
&\boldsymbol{w}_1 = \boldsymbol{y}_1 + \boldsymbol{y}_0,\\
&\boldsymbol{w}_2 = \boldsymbol{y}_2 + \boldsymbol{y}_0,
\end{aligned}$$

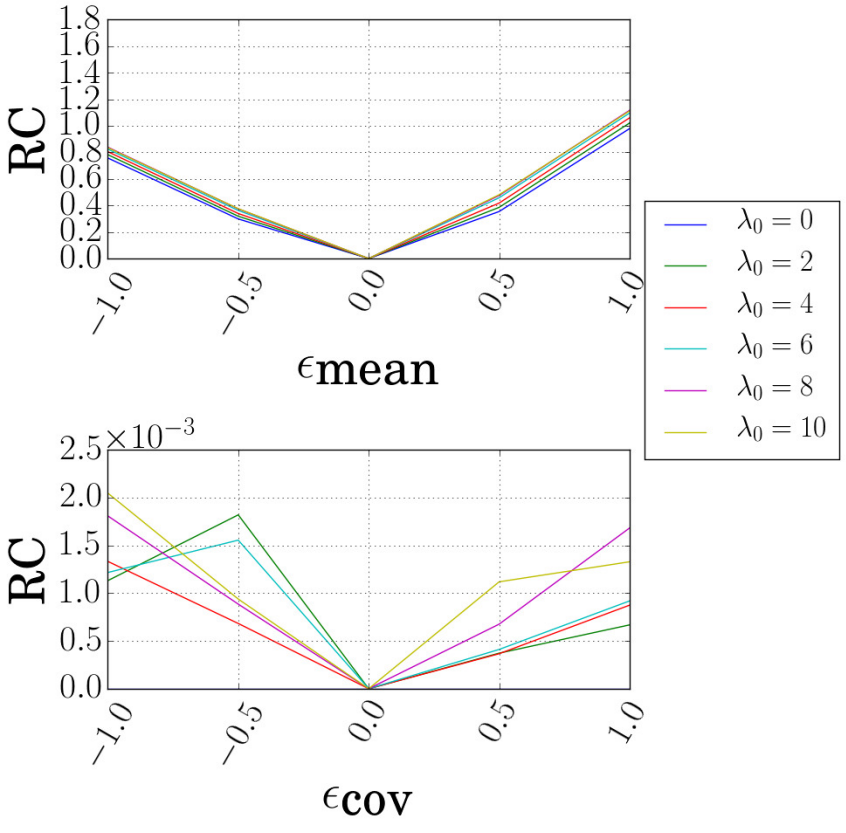
with $\epsilon_{\text{mean}}, \epsilon_{\text{cov}} \in [0, 1]$.

Figures 7.2 and 7.3 show the resulting changes in RC for the different values of λ.

Comparing the scale of variations in the relative changes, we see that the Negative kernel score is more sensitive to an error in the estimation of the marginal means (upper plots) than to the covariance parameter (lower plots), by factors of 100 to 10000. This observation supports our conclusion from the psychology attributes ratings prediction experiment. By using a loss-calibrated training objective based on the non-decomposable MinMax kernel, DISCO Nets capture structure in the learned distribution. However this does not lead to a statistically significant gain in quantitative performance at test time, compared to the CNN-CE model. The CNN-CE model is not penalised for failing to capture covariances between the attributes ratings, as the Negative kernel score hardly discriminates between distributions that differ in their covariances.

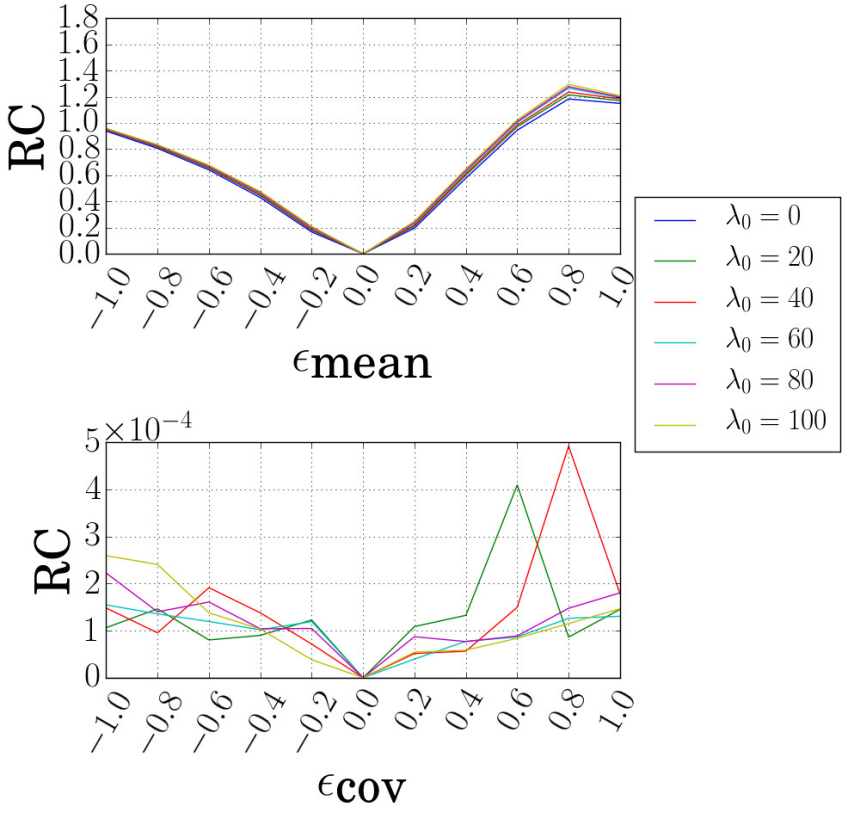


(a) $\lambda = 2$

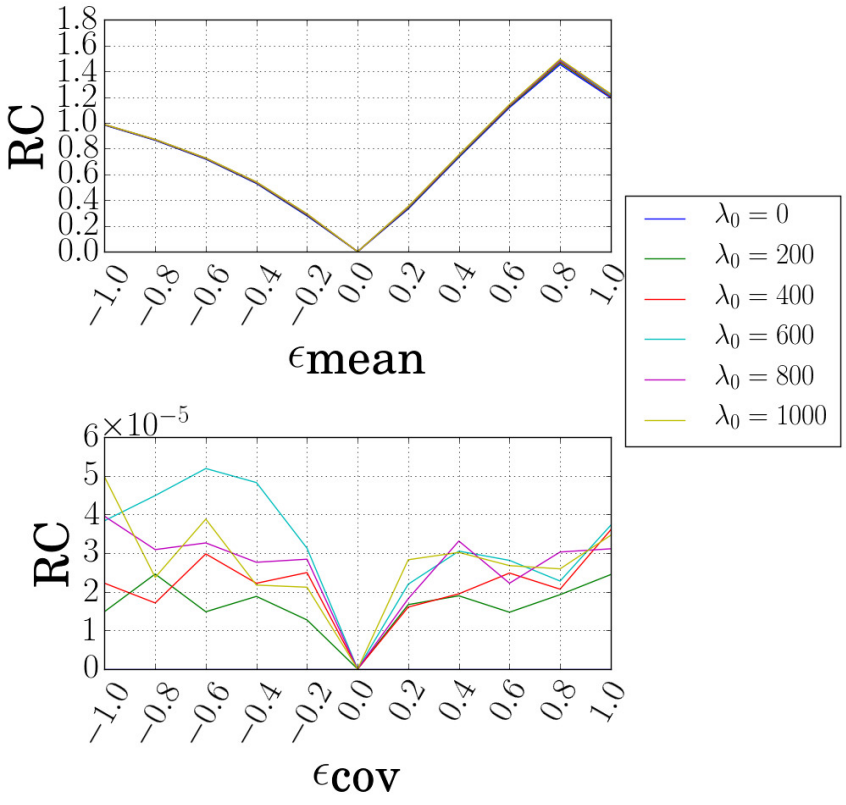


(b) $\lambda = 10$

Figure 7.2: Discrimination ability of the Negative kernel score for different values of λ*, and the two types of error.*



(a) $\lambda = 100$



(b) $\lambda = 1000$

Figure 7.3: Discrimination ability of the Negative kernel score for different values of λ*, and the two types of error.*

Training details. In our experiments, the ranges of values of the parameters $\lambda_0, \epsilon_{\text{mean}}, \epsilon_{\text{cov}}$ are chosen so that $Q_{\epsilon_{\text{mean}}}$ samples remain integers, and we threshold samples drawn from at $Q_{\epsilon_{\text{mean}}}$ at 0.

Additional analysis. In 7.2b, as the covariance λ_0 decreases, the values RC decrease. In 7.3b the values RC are independent of the covariance. Recall that $Q_{\epsilon_{\text{mean}}}$ is not a bivariate Poisson distribution, as the mean and variance of each marginal are different. Looking at distributions modelled by $Q_{\epsilon_{\text{mean}}}$, we note that the difference between the marginal means and variances of $\boldsymbol{w}_j, j = 1, 2$ is $\mathbb{E}[\boldsymbol{w}_j] - \mathbb{V}[\boldsymbol{w}_j] = -\epsilon_{\text{mean}}(\lambda + \lambda_0)$. As λ_0 decreases, this difference gets closer to 0 and the marginals get closer to Poisson distribution. therefore $Q_{\epsilon_{\text{mean}}}$ is less penalised for not modelling a Poisson distribution. This effect disappears for large values of λ, as the Poisson distribution approximates the Gaussian distribution, which is closed under linear transformation. Finally, we attribute the drop in relative changes RC from $\epsilon_{\text{mean}} = 0.8$ to $\epsilon_{\text{mean}} = 1.0$ to the thresholding of the samples at 0.

7.6 Appendix for Chapter 5

7.6.1 Derivations for the product of normals densities method

We construct the probability density function of the random variable $\boldsymbol{c}_G$ by multiplying $|G|$ normal density functions, each of them evaluating the probability of the variable $\boldsymbol{c}_G$ taking the value c_G, given $\boldsymbol{x}_i = x_i$, $\forall i \in G$, as follows,

$$q(\boldsymbol{c}_G = c_G | \boldsymbol{X}_G = X_G; \phi_c) \propto \prod_{i \in G} q(\boldsymbol{c}_G = c_G | \boldsymbol{x}_i = x_i; \phi_c)$$

We assume $q(\boldsymbol{c}_G | \boldsymbol{x}_i = x_i; \phi_c)$ to be a Normal distribution $\mathcal{N}(\mu_i, \Sigma_i)$. The normalisation constant is the resulting product marginalised over all possible values of $\boldsymbol{c}_G$. The result of the product of $|G|$ normal density functions is proportional to the density function of a Normal distribution of mean μ_G and variance Σ_G.

$$\mu_G^T \Sigma_G^{-1} = \sum_{i \in G} \mu_i^T \Sigma_i^{-1}, \ \Sigma_G^{-1} = \sum_{i \in G} \Sigma_i^{-1}.$$

We show below how we derive the expressions of mean μ_G and variance Σ_G.

$$\begin{aligned}
&\prod_{i \in G} q(\boldsymbol{c}_G = c_G | \boldsymbol{x}_i = x_i; \phi_c) \\
&= \prod_{i \in G} \frac{1}{\sqrt{(2\pi)^d |\Sigma_i|}} \exp\Big(-\frac{1}{2}(c_G - \mu_i)^T \Sigma_i^{-1} (c_G - \mu_i) \Big) \\
&= K_1 \exp\Big(-\frac{1}{2} \sum_{i \in G} (c_G - \mu_i)^T \Sigma_i^{-1} (c_G - \mu_i) \Big) \\
&= K_1 \exp\Big(-\frac{1}{2} (\sum_{i \in G} c_G^T \Sigma_i^{-1} c_G + \mu_i^T \Sigma_i \mu_i - 2\mu_i^T \Sigma_i^{-1} c_G) \Big) \\
&= K_1 K_2 \exp\Big(\sum_{i \in G} \mu_i^T \Sigma_i^{-1} c_G - \frac{1}{2} c_G^T \Sigma_i^{-1} c_G \Big)
\end{aligned}$$

$$
\begin{aligned}
&= K_1 K_2 \exp\Big(\sum_{i\in G} \mu_i^T \Sigma_i^{-1} c_G - \frac{1}{2} c_G^T \sum_{i\in G} \Sigma_i^{-1} c_G\Big) \\
&= K_1 K_2 \exp\Big(\mu_G^T \Sigma_G^{-1} c_G - \frac{1}{2} c_G^T \Sigma_G^{-1} c_G\Big) \\
&= K_1 K_2 \exp\Big(-\frac{1}{2}(c_G^T \Sigma_G^{-1} c_G - 2\mu_G^T \Sigma_G^{-1} c_G)\Big) \\
&= K_1 K_2 \exp\Big(-\frac{1}{2}(c_G^T \Sigma_G^{-1} c_G - 2\mu_G^T \Sigma_G^{-1} c_G + \mu_G^T \Sigma_G^{-1} \mu_G - \mu_G^T \Sigma_G^{-1} \mu_G)\Big) \\
&= K_1 K_2 \exp(\frac{1}{2}\mu_G^T \Sigma_G^{-1} \mu_G) \exp\Big(-\frac{1}{2}(c_G^T \Sigma_G^{-1} c_G - 2\mu_G^T \Sigma_G^{-1} c_G + \mu_G^T \Sigma_G^{-1} \mu_G\Big) \\
&= K_1 K_2 K_3 \exp\Big(-\frac{1}{2}(c_G^T \Sigma_G^{-1} c_G - 2\mu_G^T \Sigma_G^{-1} c_G + \mu_G^T \Sigma_G^{-1} \mu_G\Big) \\
&= K_1 K_2 K_3 K_4 \frac{1}{\sqrt{(2\pi)^d |\Sigma_G|}} \exp\Big(-\frac{1}{2}(c_G - \mu_G)^T \Sigma_G^{-1} (c_G - \mu_G)\Big),
\end{aligned}
$$

where d is the dimensionality of c_G and

$$
\begin{aligned}
K_1 &= \prod_{i\in G} \frac{1}{\sqrt{(2\pi)^d |\Sigma_i|}}, \\
K_2 &= \exp(-\frac{1}{2} \sum_{i\in G} \mu_i^T \Sigma_i^{-1} \mu_i), \\
K_3 &= \exp(\frac{1}{2} \mu_G^T \Sigma_G^{-1} \mu_G), \\
K_4 &= \sqrt{(2\pi)^d |\Sigma_G|}.
\end{aligned}
$$

This is a normal distribution, scaled by $K_1 K_2 K_3 K_4$, of mean μ_G and variance Σ_G, expressed as:

$$
\begin{aligned}
\Sigma_G^{-1} &= \sum_{i\in G} \Sigma_i^{-1}, \\
\mu_G^T \Sigma_G^{-1} &= \sum_{i\in G} \mu_i^T \Sigma_i^{-1}.
\end{aligned}
$$

However, the constant of normalisation disappears when we rescale the resulting

product in order for the resulting product to integrate to 1.

$$
\begin{aligned}
& q(\boldsymbol{c}_G = c_G | \boldsymbol{X}_G = X_G; \phi_c) \\
&= \frac{K_1 K_2 K_3 K_4 \frac{1}{\sqrt{(2\pi)^d |\Sigma_G|}} \exp\Big(-\frac{1}{2}(c_G - \mu_G)^T \Sigma_G^{-1} (c_G - \mu_G)\Big)}{\int_{c_G} K_1 K_2 K_3 K_4 \frac{1}{\sqrt{(2\pi)^d |\Sigma_G|}} \exp\Big(-\frac{1}{2}(c_G - \mu_G)^T \Sigma_G^{-1} (c_G - \mu_G)\Big) \mathrm{d}c_G} \\
&= \frac{\exp\Big(-\frac{1}{2}(c_G - \mu_G)^T \Sigma_G^{-1} (c_G - \mu_G)\Big)}{\int_{c_G} \exp\Big(-\frac{1}{2}(c_G - \mu_G)^T \Sigma_G^{-1} (c_G - \mu_G)\Big) \mathrm{d}c_G} \\
&= \frac{1}{\sqrt{(2\pi)^d |\Sigma_G|}} \exp\Big(-\frac{1}{2}(c_G - \mu_G)^T \Sigma_G^{-1} (c_G - \mu_G)\Big).
\end{aligned}
$$

7.6.2 Mixture of Normal densities method and results

We discuss here a method to construct the variational approximation $q(\boldsymbol{c}_G = c_G | \boldsymbol{X}_G = \boldsymbol{X}_G; \phi_c)$ as a mixture of $|G|$ density functions, each of them evaluating the probability of the variable $\boldsymbol{c}_G$ taking the value c_G, given $\boldsymbol{x}_i = x_i, \forall i \in G$. This is an alternative to the product of Normal densities method, and is expressed as,

$$
q(\boldsymbol{c}_G = c_G | \boldsymbol{X}_G = \boldsymbol{X}_G; \phi_c) = \frac{1}{|G|} \sum_{i \in G} q(\boldsymbol{c}_G = c_G | \boldsymbol{x}_i = x_i; \phi_c).
$$

We assume $q(\boldsymbol{c}_G | \boldsymbol{x}_i = x_i; \phi_c)$ to be a Normal distribution $\mathcal{N}(\mu_i, \Sigma_i)$. Let us consider the term $\mathrm{KL}(q(\boldsymbol{c}_G | X_G; \phi_c) || p(\boldsymbol{c}_G))$, which is expressed as follows,

$$
\begin{aligned}
\mathrm{KL}(q(\boldsymbol{c}_G | X_G; \phi_c) || p(\boldsymbol{c}_G)) &= \mathbb{E}_{q(\boldsymbol{c}_G | \boldsymbol{X}_G; \phi_c)} [\log q(\boldsymbol{c}_G | X_G; \phi_c) - \log p(\boldsymbol{c}_G)] \\
&= \mathbb{E}_{q(\boldsymbol{c}_G | X_G; \phi_c)} [\log q(\boldsymbol{c}_G | X_G; \phi_c)] - \mathbb{E}_{q(\boldsymbol{c}_G | X_G; \phi_c)} [\log p(\boldsymbol{c}_G)] \,.
\end{aligned}
$$

The term $\mathrm{KL}(q(\boldsymbol{c}_G | X_G; \phi_c) || p(\boldsymbol{c}_G))$ cannot be computed in closed form. We estimate

it by sampling L samples $c_{g,l} \sim q(\boldsymbol{c}_G|X_G;\phi_c)$ and computing the estimate:

$$\frac{1}{L}\sum_{l=1}^{L}\log\frac{1}{|G|}\sum_{i\in G}q(c_{g,l}|x_i;\phi_c) - \frac{1}{L}\sum_{l=1}^{L}\log p(c_{g,l}).$$

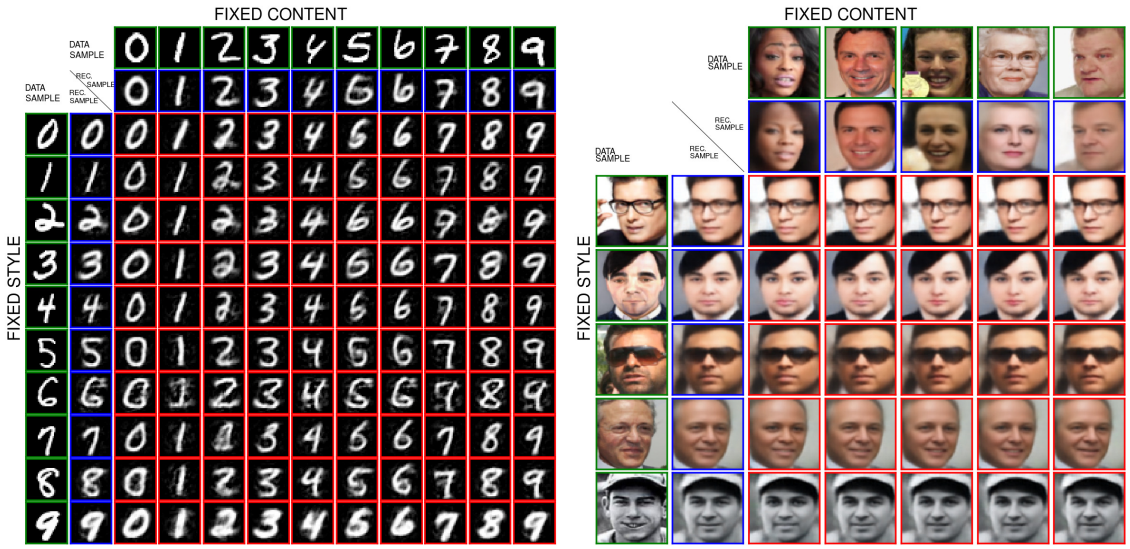
In our experiments, we used $L = |G|$, and used the samples drawn to compute the first term of the objective function, that is $\sum_{i\in G}\mathbb{E}_{q(\boldsymbol{c}_G|X_G;\phi_c)}\left[\mathbb{E}_{q(\boldsymbol{s}_i|x_i;\phi_s)}[\log p(x_i|\boldsymbol{c}_G,\boldsymbol{s}_i;\theta)]\right]$. Figure 7.4 shows the results of the mixture of Normal densities method on the swapping task. We intentionally show the same images as the ML-VAE with the product of Normal densities method for comparison purposes. Qualitative evaluation indicates a better disentanglement with the product of Normal densities method. On MS-Celeb-1M, the mixture of Normal densities method seems to store information about the grouping into the style: when style gets transferred, the facial features along the columns (which should remain consistent as it is the same identity) tend to change. Nevertheless, we present it as it might be better suited to other data sets and other tasks.

7.6.3 Bias of the ML-VAE estimate objective

We detail the bias induced by taking a subset of the samples in a group as mentioned in section 5.3.2. Let us write the entire Group ELBO, as follows,

$$\begin{aligned}\mathcal{L}(X_G;\theta,\phi_c,\phi_s) = &\sum_{i\in G}\mathbb{E}_{c_G\sim q(\boldsymbol{c}_G|X_G;\phi_c)}\left[\mathbb{E}_{s_i\sim q(\boldsymbol{s}_i|x_i;\phi_s)}[\log p(x_i|c_G,s_i;\theta)]\right]\\ &-\sum_{i\in G}\mathrm{KL}(q(\boldsymbol{s}_i|x_i;\phi_s)||p(\boldsymbol{s}_i)) - \mathrm{KL}(q(\boldsymbol{c}_G|X_G;\phi_c)||p(\boldsymbol{c}_G)).\end{aligned}$$

Let us take a subsample H of G and consider the estimator $\hat{\mathcal{L}}(X_H;\theta,\phi_c,\phi_s)$, where H denotes the fact that we use a subsample X_H of X_G. Note that since we have a single random variable $\boldsymbol{c}_G$ for the entire group, its index remains G, but $q(\boldsymbol{c}_G|X_H;\phi_c)$ is



(a) MNIST, testing data set. *(b) MS-Celeb-1M, testing data set.*

Figure 7.4: ML-VAE with mixture of Normal densities method. Swapping, the first row and the first column show the test data sample input (green boxes), the second row and column are reconstructed samples (blue boxes). In the remaining rows and columns, each row is a fixed style and each column is a fixed content (red boxes). Compared to ML-VAE with the product of Normal densities method, the mixture of Normal densities method seems to store information about the grouping into the style.

computed using $X_H \subseteq X_G$.

$$\hat{\mathcal{L}}(X_H;\theta,\phi_c,\phi_s) = \sum_{i\in H\subseteq G} \mathbb{E}_{c_G\sim q(\boldsymbol{c}_G|X_H;\phi_c)}\left[\mathbb{E}_{s_i\sim q(\boldsymbol{s}_i|x_i;\phi_s)}[\log p(x_i|c_G,s_i;\theta)]\right] - \sum_{i\in H\subseteq G} \mathrm{KL}(q(\boldsymbol{s}_i|x_i;\phi_s)||p(\boldsymbol{s}_i)) - \mathrm{KL}(q(\boldsymbol{c}_G|X_H;\phi_c)||p(\boldsymbol{c}_G)).$$

In the case of the product of Normal densities method, we construct the content variational posterior distribution $q(\boldsymbol{c}_G|X_H;\phi_c)$ in a non-linear manner. Both the first term of the Group ELBO and the third term $\mathrm{KL}(q(\boldsymbol{c}_G|X_H;\phi_c)||p(\boldsymbol{c}_G))$ cannot be cannot be simply weighted by the size of the subsampled group to remove the bias. Using the mixture of Normal densities, in the third term of $\hat{\mathcal{L}}(X_H;\theta,\phi_c,\phi_s)$, that is, the Kullback-Leibler divergence term $\mathrm{KL}(q(\boldsymbol{c}_G|X_H;\phi_c)||p(\boldsymbol{c}_G))$ the logarithm of the posterior of the content will not decompose as the sum of logarithms, and the

estimator will be biased too. Thereby, the bias will depend on the method employed.

7.6.4 Disentanglement of digit and face images

In all experiments we use the Adam optimiser (Kingma and Welling, 2014) with $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1e-8$. We use Normal distributions with diagonal covariance matrix for the distributions $q(\boldsymbol{S}_G|X_G;\phi_s), q(\boldsymbol{c}_G|x_i;\phi_c)$. We also use a Normal distribution with diagonal covariance matrix for the posterior $p(\boldsymbol{x}_i|c_G, s_i;\theta)$. We train the model for 2000 epochs on MNIST and 250 epochs on MS-Celeb-1M. When we compare the original VAE and the ML-VAE we use early stopping on the validation set for MS-Celeb-1M as the architecture is larger and prone to over-fitting. In the specific case of Stochastic Variational Inference (SVI) (Hoffman et al., 2013), for MNIST we train the model for 2000 epochs and proceed to 40000 epochs of inference at test-time, and for MS-Celeb-1M we use 500 training epochs and proceed to 200 epochs of inference at test-time

MNIST (LeCun et al., 1998) architecture. We use an encoder network composed of a first linear layer e_0 that takes as input a 1×784-dimensional MNIST image x_i, that is, x_i is an observation of $\boldsymbol{x}_i$. Layer e_0 has 500 hidden units and the hyperbolic tangent activation function. After layer e_0 we separate the network into 4 linear layers, $e_{m,s}, e_{v,s}$ and $e_{m,c}, e_{v,c}$, each of size $500 \times d$ where d is the dimension of the latent representation for style and content. The layers $e_{m,s}, e_{v,s}$ take as input the output of e_0 and output respectively the mean and log-variance of the Normal distribution $q(\boldsymbol{s}_i|x_i;\phi_s)$. The layers $e_{m,c}, e_{v,c}$ take as input the output of l_0 and output respectively the mean and variance of the Normal distribution $q(\boldsymbol{c}_G|x_i;\phi_c)$.

We construct $q(\boldsymbol{c}_G|X_G;\phi_c)$ from $q(\boldsymbol{c}_G|x_i;\phi_c), \forall i \in G$. Let us denote G the group which x_i belongs to. As explained in step 10 of Algorithm 4, for each input x_i we draw a sample $c_{G,i} \sim q(\boldsymbol{c}_G|X_G = X_G;\phi_c)$ for the content latent representation and a

sample $s_i \sim q(\boldsymbol{s}_i|x_i;\phi_s)$ of the style latent representation. We concatenate $(c_{G,i}, s_i)$ into a $2 \times d$-dimensional vector that is fed to the decoder.

The decoder network is composed of a first linear layer d_0 that takes as input the $2 \times d$-dimensional vector $(c_{G,i}, s_i)$. Layer d_0 has 500 hidden units and the hyperbolic tangent activation function. A second linear layer d_2 takes as input the output of d_0 and outputs a 784-dimensional vector representating the parameters of the Normal distribution $p(\boldsymbol{x}_i|c_{G,i}, s_i;\theta)$. We use $d = 10$ for a total latent representation size of respectively 20.

MS-Celeb-1M (Guo et al., 2016) architecture. We use an encoder network composed of a four convolutional layers e_1, e_2, e_3, e_4 composed of respectively $64, 128, 256$ and 512 filters, with kernels of size 4×4 with stride 2. All four layers are followed by Batch Normalisation and Rectified Linear Units (ReLUs) activation functions. The fifth and sixth layers e_5, e_6 are linear layers with 256 hidden units, followed by Batch Normalisation and Concatenated Rectified Linear Unit (CReLUs) activation functions. Similar to the MNIST architecture, after layer e_6 we separate the network into 4 linear layers,$e_{m,s}$, $e_{v,s}$ and ,$e_{m,c}$, $e_{v,c}$ each of size $256 \times 2 \times d$ where d is the dimension of the latent representations for style and content. The layers for the log-variances are followed by the tangent hyperbolic activation function and multiplied by 5. Similar to the MNIST experiment, we construct the latent representation and draw samples from it as explained in Algorithm 4.

The decoder network is composed of 3 deconvolutional layers d_1, d_2, d_3, composed of respectively $256, 128, 64$ filters, with kernels of sizes 4×4 with stride 2. All four layers are followed by Batch Normalisation and Rectified Linear Units (ReLUs) activation functions. The seventh and eight layers are deconvolutional layers composed of 3 filters, with kernels of size 3×3 with stride 1. They output respectively the mean and log-variance of the Normal distribution $p(\boldsymbol{x}_i|c_{G,i}, s_i;\theta)$. The

layer for the log-variances is followed by the tangent hyperbolic activation function and multiplied by 5. We use in our experiments $d = 50$ for a total latent representation size 100. We use padding in the convolutional and deconvolutional layers to match the data size.

ML-VAE with product of Normal densities method without accumulating evidence results. Figure 7.5 show the results of swapping and interpolation of the ML-VAE with the product of Normal densities method without accumulating evidence (strategy 1).

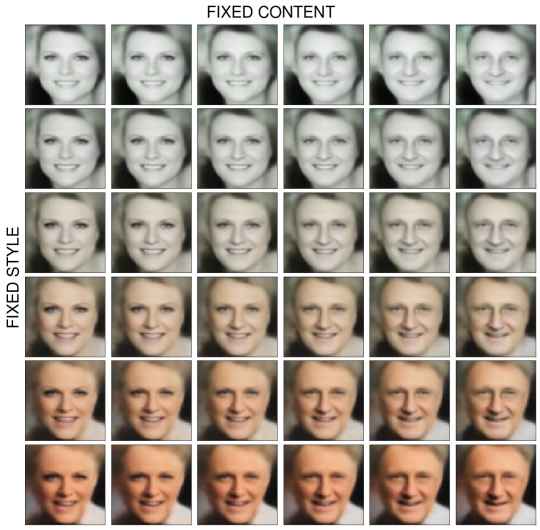
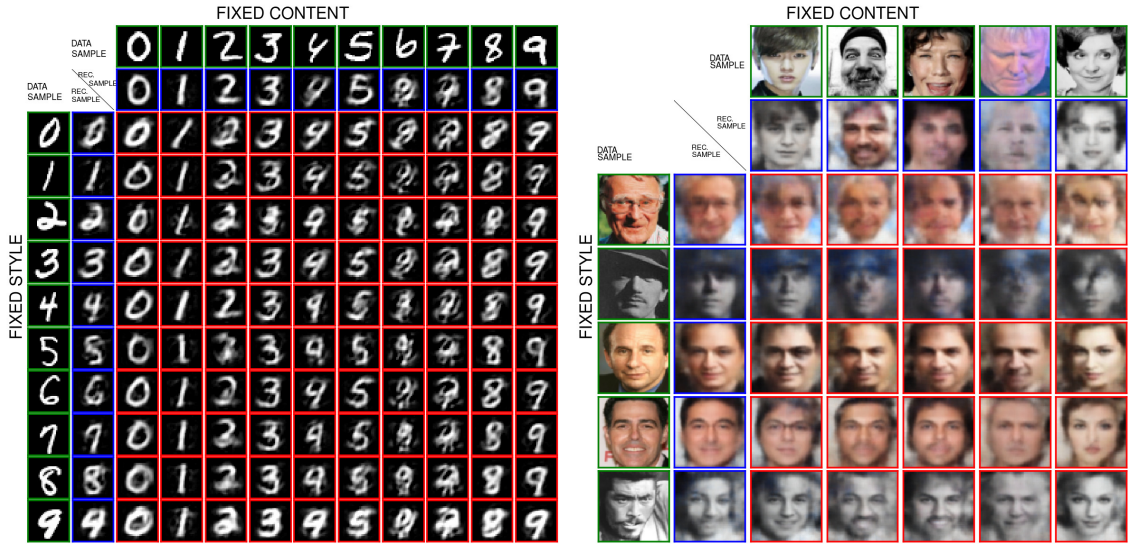


Figure 7.5: ML-VAE with the product of Normal densities method without accumulating evidence. Interpolation, from top left to bottom right, rows correspond to a fixed style and interpolating on the content, columns correspond to a fixed content and interpolating on the style. We see that interpolation results without accumulated evidence (strategy 1) are also convincing.

Stochastic Variational Inference (SVI) results We compare in our experiments with Stochastic Variational Inference (SVI) (Hoffman et al., 2013). In the case of SVI, the encoder is an embedding layer mapping each sample x_i in a group G to the non-shared parameters $\phi_{s,i}$ of its style latent representation $q(\boldsymbol{s}_i|\phi_{s,i})$ and to the non-shared parameters $\phi_{c,G}$ of its group content latent representation $q(\boldsymbol{c}_G|\phi_{c,G})$.

The decoder is the same as for the ML-VAE and VAE models.

We show in Figure 7.6 the qualitative results of Stochastic Variational Inference (SVI) (Hoffman et al., 2013) on the swapping evaluation. We see that while SVI disentangles the style and the content, the resulting image quality is poor. In the case of MS-Celeb-1M, we trained SVI for twice the number of epochs of the other models, that is in total 500 epochs. SVI does not share the parameters ϕ_c, ϕ_s among observations at training, hence takes a longer time to train. It is the first disadvantadge of SVI compared to VAE-based model. At test-time, the SVI model requires expensive iterative inference. In the case of MS-Celeb-1M we used 200 epochs of test inference, it is possible that more epochs of test-time inference would have lead to better quality images but this already shows the limits of non-amortised variational inference and the advantages of the ML-VAE.



(a) MNIST, testing data set. *(b) MS-Celeb-1M, testing data set.*

Figure 7.6: Stochastic Variational Inference (SVI). Swapping, first row and first column are test data samples (green boxes), second row and column are reconstructed samples (blue boxes) and the rest are swapped reconstructed samples (red boxes). Each row is fixed style and each column is a fixed content.

Quantitative evaluation details. We explain in section 5.4 how we quantitatively evaluate the disentanglement performance of our model. The classifier is a neural network composed of two linear layers of 256 hidden units each. The first layer is followed by a tangent hyperbolic activation function. The second layer is followed by a softmax activation function. We use the Adam optimiser (Kingma and Welling, 2014) with $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1e-8$. Note that the training, validation and testing sets for the classifier are all composed of test images, and each set must have K samples per classes; hence our choice of K and number of classes for MS-Celeb-1M. In the case of MNIST, there are only 10 classes, therefore when k is small we would take only a small number of images to test the classifier. Therefore we perform 5 trials of test procedures of the classifier, each trial using different test images, and report the mean performance. Figure 7.7 shows the quantitative evaluation for k up to $k = K = 100$ on MNIST.

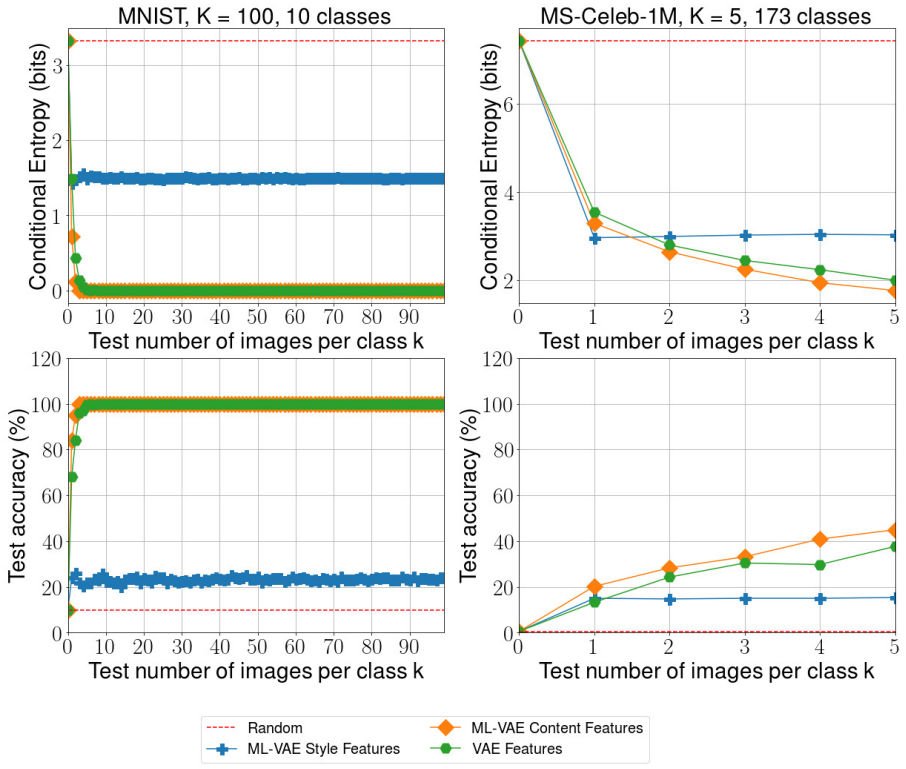


Figure 7.7: Full quantitative evaluation of the ML-VAE. Accuracy (higher is better) and conditional entropy (lower is better).

Bibliography

Abbasnejad, E., Dick, A. R., and van den Hengel, A. Infinite variational autoencoder for semi-supervised learning. *arxiv:1611.07800*, 2016. (pp. 107.)

Aczél, J. and Daróczy, Z. Charakterisierung der entropien positiver ordnung und der Shannonschen entropie. *Acta Mathematica Hungarica*, 1963. (pp. 38 and 39.)

Alemi, A. A., Fischer, I., Dillon, J. V., and Murphy, K. P. Deep variational information bottleneck. *ICLR*, 2017. (pp. 107.)

Allamanis, M., Chanthirasegaran, P., Kohli, P., and Sutton, C. Learning continuous semantic representations of symbolic expressions. *ICML*, 2017. (pp. 108.)

Bahdanau, D., Serdyuk, D., Brakel, P., Ke, N. R., Chorowski, J., and Bengio, Y. Task loss estimation for sequence prediction. *arxiv:1511.06456*, 2016. (pp. 7.)

Bainbridge, W. A., Isola, P., and Oliva, A. The intrinsic memorability of face photographs. *Journal of Experimental Psychology: General*, 2013. (pp. xiii, 81, 84, 92, 93, 132, 186, and 187.)

BakIr, G., Hofman, T., Schölkopf, B., Smola, A. J., Taskar, B., and Vishwanathan, S. V. *Predicting Structured Data*. MIT Press, 2007. (pp. 33.)

Behl, A., Jawahar, C. V., and Kumar, M. P. Optimizing Average Precision using weakly supervised data. *CVPR*, 2014. (pp. 34 and 50.)

Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, Aug 2013. (pp. 103.)

Bernardo, J. M. Expected information as expected utility. *The Annals of Statistics*, 7(3):891–921, May 1979. (pp. 14.)

Blaschko, M. B. and Lampert, C. H. Learning to localize objects with structured output regression. *ECCV*, 2008. (pp. 33.)

Borgwardt, K. M., Gretton, A., Rasch, M. J., Kriegel, H.-P., Schölkopf, B., and Smola, A. J. Integrating structured biological data by Kernel Maximum Mean Discrepancy. *Bioinformatics (ISMB)*, 22(14):e49–e57, 2006. (pp. 16.)

Bouchacourt, D., Nowozin, S., and Kumar, M. P. Entropy-based latent structured output prediction. *ICCV*, 2015. (pp. v, 8, and 57.)

Bouchacourt, D., Kumar, M. P., and Nowozin, S. DISCO Nets: DISsimilarity COefficient Networks. *NIPS*, 2016. (pp. v, 8, 61, and 79.)

Bouchacourt, D., Tomioka, R., and Nowozin, S. Multi-level variational autoencoder: Learning disentangled representations from grouped observations. *arxiv:1705.08841*, 2017. (pp. v and 8.)

Bouchard, M., Jousselme, A.-L., and Doré, P.-E. A proof for the positive definiteness of the Jaccard index matrix. *International Journal of Approximate Reasoning*, 54(5):615–626, 2013. (pp. 91.)

Bousmalis, K., Silberman, N., Dohan, D., Erhan, D., and Dilip Krishnan, D. Unsupervised pixel-level domain adaptation with generative adversarial networks. *CVPR*, 2017. (pp. 108.)

Boykov, Y., Veksler, O., and Zabih, R. Fast approximate energy minimization via graph cuts. *PAMI*, 23(11):1222–1239, 2001. (pp. 57.)

Bromiley, P. A., Thacker, N. A., and Bouhova-Thacker, E. *Shannon Entropy, Renyi Entropy, and Information*, 2010. URL `http://www.tina-vision.net/docs/memos/2004-004.pdf`. (pp. 39.)

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. *NIPS*, 2016. (pp. 107.)

Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., and Abbeel, P. Variational lossy autoencoder. *ICLR*, 2017. (pp. 107.)

Cisse, M., Adi, Y., Neverova, N., and Keshet, J. Houdini: Fooling deep structured prediction models. *arxiv:1707.05373*, 2017. (pp. 7.)

Dawid, A. P. The geometry of proper scoring rules. *Annals of the Institute of Statistical Mathematics*, 59(1):77–93, 2007. (pp. 16.)

Dawid, A. P. and Musio, M. Bayesian model selection based on proper scoring rules. *Bayesian Analysis*, 10:479–499, 2015. (pp. 13.)

Dawid, A. P., Musio, M., and Columbu, S. A note on Bayesian model selection for discrete data using proper scoring rules. *arxiv:1703.03353*, 2017. (pp. 13.)

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 1977. (pp. 20 and 34.)

Denton, E. and Birodkar, V. Unsupervised learning of disentangled representations from video. *arxiv:1705.10915*, 2017. (pp. 108.)

Denton, E., Chintala, S., Szlam, A., and Fergus, R. Deep generative image models using a laplacian pyramid of adversarial networks. *NIPS*, 2015. (pp. 28.)

Denton, E., Gross, S., and Fergus, R. Semi-supervised learning with context-conditional generative adversarial networks. *arxiv:1606.01583*, 2016. (pp. 61.)

Donahue, C., Balsubramani, A., McAuley, J., and Lipton, Z. C. Semantically decomposing the latent spaces of generative adversarial networks. *arxiv:1705.07904*, 2017. (pp. 108.)

Durand, T., Thome, N., and Cord, M. WELDON: Weakly supervised learning of deep convolutional neural networks. *CVPR*, 2016. (pp. 57.)

Dziugaite, G. K., Roy, D. M., and Ghahramani, Z. Training generative neural networks via maximum mean discrepancy optimization. *UAI*, 2015. (pp. 29, 61, 100, and 130.)

Edwards, H. and Storkey, A. J. Censoring representations with an adversary. *ICLR*, 2016. (pp. 107.)

Edwards, H. and Storkey, A. J. Towards a neural statistician. *ICLR*, 2017. (pp. 109.)

Esteban, M. D. and Morales, D. A summary on entropy statistics. *Kybernetika*, 31 (4):337–346, 1995. (pp. 39.)

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 2010. (pp. 4, 18, 41, and 49.)

Felzenszwalb, P., McAllester, D., and Ramanan, D. A discriminatively trained, multiscale, deformable part model. *CVPR*, 2008. (pp. 24, 32, and 34.)

Felzenszwalb, P. F., Girshick, R., and McAllester, D. A. *Discriminatively trained deformable part models, release 4*, 2010. URL `http://cs.brown.edu/~pff/latent-release4/`. (pp. 50.)

Fothergill, S., Mentis, H. M., Kohli, P., and Nowozin, S. Instructing people for training gestural interactive systems. *CHI*, 2012. (pp. 52 and 53.)

Fu, T.-C., Liu, Y.-C., Chiu, W.-C., Wang, S.-D., and Wang, Y.-C. F. Learning cross-domain disentangled deep representation with supervision from a single domain. *arxiv:1705.01314*, 2017. (pp. 108.)

Fukumizu, K., Song, L., and Gretton, A. Kernel Bayes' rule: Bayesian inference with positive definite kernels. *J. Machine Learning Research*, 2013. (pp. 74.)

Gauthier, J. *Conditional generative adversarial nets for convolutional face generation*, 2015. (pp. 60.)

Gelman, A., Carlin, J., Stern, H., and Rubin, D. *Bayesian Data Analysis*. Chapman and Hall, 1995. (pp. 34.)

Gneiting, T. and Raftery, A. E. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102:359–378, 2007. (pp. 11, 12, 15, 16, 144, and 146.)

Goodfellow, I. J. NIPS 2016 tutorial: Generative adversarial networks. *arxiv:1701.00160*, 2016. (pp. 61.)

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *NIPS*, 2014. (pp. 28, 61, and 79.)

Gower, J. C. A general coefficient of similarity and some of its properties. *Biometrics*, 27(4):857–871, 1971. (pp. 91.)

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Alexander, S. A kernel two-sample test. *JMLR*, 2012. (pp. 16 and 29.)

Guo, Y., Zhang, L., Hu, Y., He, X., and Gao, J. MS-Celeb-1M: A dataset and benchmark for large scale face recognition. *ECCV*, 2016. (pp. 117 and 204.)

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-VAE: Learning basic visual concepts with a constrained variational framework. *ICLR*, 2017. (pp. 107.)

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. Stochastic variational inference. *J. Machine Learning Research*, 2013. (pp. 110, 203, 205, and 206.)

Horst, R., Pardalos, P. M., and V., N. *Introduction to Global Optimization*. Springer, 2000. (pp. 150.)

Huszár, F. *Scoring rules, divergences and information in Bayesian machine learning, PhD Thesis*, 2013. (pp. 12, 16, 17, and 130.)

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017. (pp. 61.)

Jaccard, P. Distribution de la flore alpine dans le Bassin des Dranses et dans quelques rgions voisines. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37:241–272, 1901. (pp. 82 and 91.)

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with Gumbel-Softmax. *ICLR*, 2017. (pp. 82.)

Joachims, T., Finley, T., and Yu, C.-N. J. Cutting-plane training of structural SVMs. *J. Machine Learning Research*, 2009. (pp. 151.)

Johnson, M. J., Duvenaud, D., Wiltschko, A. B., Datta, S. R., and Adams, R. P. Composing graphical models with neural networks for structured representations and fast inference. *NIPS*, 2016. (pp. 115 and 126.)

Johnson, N. L., Kotz, S., and Balakrishnan, N. *Discrete Multivariate Distributions*. John Wiley & Sons, Ltd, 1997. (pp. 191.)

Karaletsos, T., Belongie, S., and Rätsch, G. Bayesian representation learning with oracle constraints. *ICLR*, 2016. (pp. 108 and 109.)

Kemp, C., Tenenbaum, J. B., Griffiths, T. L., Yamada, T., and Ueda, N. Learning systems of concepts with an infinite relational model. *AAAI*, 2006. (pp. 132.)

Khoshnevisan, D. *Probability*. American Mathematical Society, 2007. (pp. 138.)

Kim, T., Cha, M., Kim, H., Lee, J. K., and Kim, J. Learning to discover cross-domain relations with generative adversarial networks. *ICML*, 2017. (pp. 108.)

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. *ICLR*, 2014. (pp. 29, 30, 203, and 207.)

Kingma, D. P., Rezende, D. J., Mohamed, S., and Welling, M. Semi-supervised learning with deep generative models. *NIPS*, 2014. (pp. 107.)

Kulkarni, T. D., Whitney, W., Kohli, P., and Tenenbaum, J. B. Deep convolutional inverse graphics network. *NIPS*, 2015. (pp. 108.)

Kumar, M. P., Turki, H., Preston, D., and Koller, D. Learning specific-class segmentation from diverse data. *ICCV*, 2011. (pp. 33 and 34.)

Kumar, M. P., Packer, B., and Koller, D. Modeling latent variable uncertainty for loss-based learning. *ICML*, 2012. (pp. 65.)

Kusner, M. J. and Hernández-Lobato, J. M. GANs for sequences of discrete elements with the Gumbel-softmax distribution. *NIPS Workshop on Adversarial Training*, 2016. (pp. 82.)

Lacoste-Julien, S., Huszár, F., and Ghahramani, Z. Approximate inference for the loss-calibrated Bayesian. *AISTATS*, 2011. (pp. 5.)

Larsen, A. B. L. and Sønderby, S. K. *Generating Faces with Torch*, 2015. URL `http://torch.ch/blog/2015/11/13/gan.html`. (pp. 79.)

Leadbetter, R., Cambanis, S., and Pipiras, V. *A Basic Course in Measure and Probability*. Cambridge University Press, 2014. (pp. 71, 138, and 141.)

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *IEEE*, 86(11):2278–2324, 1998. (pp. 26, 117, and 203.)

Lehrmann, A. M., Gehler, P. V., and Nowozin, S. Efficient nonlinear Markov models for human motion. *CVPR*, 2014. (pp. 52 and 53.)

Li, C.-L., Chang, W.-C., Cheng, Y., Yang, Y., and Póczos, B. MMD GAN: Towards deeper understanding of moment matching network. *arxiv:1705.08584*, 2017. (pp. 100 and 131.)

Li, Y., Swersky, K., and Zemel, R. Generative moment matching networks. *ICML*, 2015. (pp. 29, 61, 100, and 130.)

Linsker, R. Self-organization in a perceptual network. *Computer*, 21(3):105–117, Mar 1988. (pp. 103.)

Liu, M., Breuel, T., and Kautz, J. Unsupervised image-to-image translation networks. *arxiv:1703.00848*, 2017. (pp. 108.)

London, B. and Schwing, A. G. Generative adversarial structured networks. *NIPS Workshop on Adversarial Training*, 2016. (pp. 82.)

Lopez-Paz, D., Radford, A., and Bottou, L. *NIPS Workshop on adversarial training*, 2016. URL `https://sites.google.com/site/nips2016adversarial/`. (pp. 61.)

Louizos, C., Swersky, K., Li, Y., Welling, M., and Zemel, R. The variational fair autoencoder. *ICLR*, 2016. (pp. 107.)

MacKay, D. J. C. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, 2002. (pp. 21.)

Maddison, C. J., Mnih, A., and Whye Teh, Y. The concrete distribution: A continuous relaxation of discrete random variables. *ICLR*, 2017. (pp. 82.)

Maji, S., Bourdev, L. D., and Malik, J. Action recognition from a distributed representation of pose and appearance. *CVPR*, 2011. (pp. 50.)

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I. J., and Frey, B. Adversarial autoencoders. *ICLR Workshop*, 2015. (pp. 31 and 107.)

Mathai, A. M. and Rathie, P. N. *Basic Concepts in Information Theory and Statistics: Axiomatic Foundations and Applications*. John Wiley & Sons, Ltd, 1975. (pp. 39.)

Mathieu, M. F., Zhao, J. J., Zhao, J., Ramesh, A., Sprechmann, P., and LeCun, Y. Disentangling factors of variation in deep representation using adversarial training. *NIPS*, 2016. (pp. 107 and 117.)

McAllester, D. A., Hazan, T., and Keshet, J. Direct loss minimization for structured prediction. *NIPS*, 2010. (pp. 83, 85, 95, 166, 171, 173, 174, 182, and 183.)

Mescheder, L. M., Nowozin, S., and Geiger, A. Adversarial variational Bayes: Unifying variational autoencoders and generative adversarial networks. *ICML*, 2017. (pp. 31.)

Miller, K., Kumar, M. P., Packer, B., Goodman, D., and Koller, D. Max-margin min-entropy models. *AISTATS*, 2012. (pp. 36 and 56.)

Mirza, M. and Osindero, S. Conditional generative adversarial nets. *NIPS Deep Learning Workshop*, 2014. (pp. 60 and 79.)

Muandet, K., Fukumizu, K., Sriperumbudur, B. K., and Schölkopf, B. Kernel mean embedding of distributions: A review and beyonds. *Foundations and Trends in Machine Learning*, 10(1–2):1–141, 2016. (pp. 16.)

Murali, V., Chaudhuri, S., and Jermaine, C. Bayesian sketch learning for program synthesis. *arxiv:1703.05698v2*, 2017. (pp. 108.)

Murphy, K. P. *Conjugate Bayesian Analysis of the Gaussian Distribution*, 2007. (pp. 115.)

Musio, M. and Dawid, A. P. Rényi's scoring rules. *48th Scientific Meeting of the Italian Statistical Society*, 2016. (pp. 130.)

Neal, R. and Hinton, G. E. *A View Of The EM Algorithm That Justifies Incremental, Sparse, And Other Variants*. Kluwer Academic Publishers, 1998. (pp. 34.)

Nowozin, S. and Lampert, C. H. Structured learning and prediction in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 2011. (pp. 33 and 81.)

Nowozin, S. and Shotton, J. *Action Points: A Representation for Low-latency Online Human Action Recognition*, July 2012. (pp. 52 and 53.)

Oberweger, M. *DeepPrior - Accurate and Fast 3D Hand Pose Estimation*, 2015. URL `https://github.com/moberweger/deep-prior`. (pp. 183.)

Oberweger, M., Wohlhart, P., and Lepetit, V. Training a feedback loop for hand pose estimation. *ICCV*, 2015a. (pp. 74, 76, and 183.)

Oberweger, M., Wohlhart, P., and Lepetit, V. Hands deep in deep learning for hand pose estimation. *Computer Vision Winter Workshop*, 2015b. (pp. 3, 5, 62, 74, 76, 78, 80, and 183.)

Oquab, M., Bottou, L., Laptev, I., and Sivic, J. Is object localization for free? weakly-supervised learning with convolutional neural networks. *CVPR*, 2015. (pp. 57.)

Osokin, A., Bach, F., and Lacoste-Julien, S. On structured prediction theory with calibrated convex surrogate losses. *arxiv:1703.02403*, 2017. (pp. 7.)

Paisley, J. W., Blei, D. M., and Jordan, M. I. Variational Bayesian inference with stochastic search. *ICML*, 2012. (pp. 82.)

Parmigiani, G. and Inoue, L. *Decision Theory: Principles and Approaches*. John Wiley & Sons, Ltd, 2009. (pp. 13 and 14.)

Pathak, D., Krähenbühl, P., and Darrell, T. Constrained convolutional neural networks for weakly supervised segmentation. *ICCV*, 2015. (pp. 57.)

Ping, W., Liu, Q., and Ihler, A. Marginal structured SVM with hidden variables. *ICML*, 2014. (pp. 35 and 36.)

Pinson, P. and Tastu, J. *Discrimination ability of the Energy score*. Technical report, Technical University of Denmark, 2013. (pp. 190 and 191.)

Pitman, J. *Combinatorial stochastic processes*. Technical report, Department of Statistics, University of California at Berkeley, 2002. (pp. 132.)

Polyak, B. T. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4:1–17, 1964. (pp. 185.)

Premachandran, V., Tarlow, D., and Batra, D. Empirical minimum Bayes risk prediction: How to extract an extra few% performance from vision models with just three more parameters. *CVPR*, 2014. (pp. 63.)

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *ICLR*, 2015. (pp. 28, 61, and 79.)

Rahman, A. and Wang, Y. Optimizing Intersection-Over-Union in deep neural networks for image segmentation. *International Symposium on Visual Computing*, 2016. (pp. 82.)

Ralaivola, L., Swamidass, S. J., Saigo, H., and Baldi, P. Graph kernels for chemical informatics. *Neural Networks*, 18:1093–1110, Oct 2005. (pp. 91.)

Rao, C. R. Diversity and dissimilarity coefficients: A unified approach. *Theoretical Population Biology*, 1982. (pp. 60 and 65.)

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., and Lee, H. Generative adversarial text to image synthesis. *ICML*, 2016. (pp. 60.)

Ren, Y., Li, J., Luo, Y., and Zhu, J. Conditional generative moment-matching networks. *NIPS*, 2016. (pp. 61 and 130.)

Rezende, D. J., Mohamed, S., and Daan, W. Stochastic backpropagation and variational inference in deep generative models. *arxiv:1401.4082v3*, 2014. (pp. 29 and 30.)

Rolf, S. Maximum likelihood theory for incomplete data from an exponential family. *Scandinavian Journal of Statistics*, 1(2):49–58, 1974. (pp. 20 and 34.)

Rolfe, J. T. Discrete variational autoencoders. *ICLR*, 2017. (pp. 82.)

Ross, S. M. *Applied Probability Models with Optimization Applications*. Dover Publications, 1970. (pp. 26.)

Salojärvi, J., Puolamäki, K., and Kaski, S. Expectation maximization algorithms for conditional likelihoods. *ICML*, 2005. (pp. 34.)

Schwing, A. G., Hazan, T., Pollefeys, M., and Urtasun, R. Efficient structured prediction with latent variables for general graphical models. *ICML*, 2012. (pp. 35.)

Serfling, R. *Approximation theorems of mathematical statistics*. John Wiley & Sons, Ltd, 1980. (pp. 67.)

Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W., and Webb, R. Learning from simulated and unsupervised images through adversarial training. *CVPR*, 2017. (pp. 108.)

Siddharth, N., Paige, B., , Van de Meent, J.-W., Desmaison, A., Wood, F., Goodman, N. D., Kohli, P., and Torr, P. H. Learning disentangled representations with semi-supervised deep generative models. *arXiv:1706.00400*, 2017. (pp. 107.)

Sohn, K., Lee, H., and Yan, X. Learning structured output representation using deep conditional generative models. *NIPS*, 2015. (pp. 61.)

Song, Y., Schwing, A. G., Zemel, R., and Urtasun, R. Training deep neural networks via direct loss minimization. *ICML*, 2016. (pp. 83, 85, 95, 163, 164, 166, 172, 180, and 182.)

Springenberg, J. T. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *ICLR*, 2016. (pp. 28.)

Taigman, Y., Polyak, A., and Wolf, L. Unsupervised cross-domain image generation. *ICLR*, 2017. (pp. 108.)

Tarlow, D. and Zemel, R. S. Structured output learning with high order loss functions. *AISTATS*, 2012. (pp. 91.)

Taskar, B., Guestrin, C., and Koller, D. Max-margin Markov networks. *NIPS*, 2003. (pp. 23 and 24.)

Taylor, J., Shotton, J., Sharp, T., and Fitzgibbon, A. The Vitruvian manifold: Inferring dense correspondences for oneshot human pose estimation. *CVPR*, 2012. (pp. 74.)

Tian, T., Chen, N., and Zhu, J. Learning attributes from the crowdsourced relative labels. *AAAI*, 2017. (pp. 109.)

Tokmakov, P., Alahari, K., and Schmid, C. Weakly-supervised semantic segmentation using motion cues. *ECCV*, 2016. (pp. 57.)

Tompson, J., Stein, M., Lecun, Y., and Perlin, K. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics*, 33, Aug 2014. (pp. 3, 5, 62, 74, 75, and 78.)

Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. Support vector machine learning for interdependent and structured output spaces. *ICML*, 2004. (pp. 23 and 24.)

Tulyakov, S., Liu, M.-Y., Yang, X., and Kautz, J. MoCoGAN: Decomposing motion and content for video generation. *CVPR*, 2017. (pp. 108.)

Vapnik, V. N. *The nature of statistical learning theory.* Springer, 1995. (pp. 23.)

Veit, A., Belongie, S., and Karaletsos, T. Conditional similarity networks. *CVPR*, 2017. (pp. 109.)

Wang, H., Gould, S., and Roller, D. Discriminative learning with latent variables for cluttered indoor scene understanding. *ECCV*, 2010. (pp. 34.)

Wang, X. and Gupta, A. Generative image modeling using style and structure adversarial networks. *ECCV*, 2016. (pp. 107.)

Wasserman, L. *All of Statistics: A Concise Course in Statistical Inference.* Springer, 2010. (pp. 140.)

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992. (pp. 82.)

Winkelmann, R. *Econometric Analysis of Count Data.* Springer, 2008. (pp. 193.)

Yan, X., Yang, K., Jimei Sohn, and Lee, H. Attribute2Image: Conditional image generation from visual attributes. *ECCV*, 2016. (pp. 61.)

Yi, Z., Zhang, H., Tan, P., and Gong, M. DualGAN: Unsupervised dual learning for image-to-image translation. *arxiv:1704.02510*, 2017. (pp. 108.)

Yu, C.-N. J. and Joachims, T. Learning structural SVMs with latent variables. *ICML*, 2009. (pp. 24, 32, and 34.)

Yu, L., Zhang, W., Wang, J., and Yu, Y. SeqGAN: Sequence generative adversarial nets with policy gradient. *AAAI*, 2017. (pp. 82.)

Yuille, A. L. and Rangarajan, A. The concave-convex procedure (CCCP). *NIPS*, 2002. (pp. 25, 35, 46, 47, and 151.)

Zhao, J., Kim, Y., Zhang, K., Rush, A. M., and LeCun, Y. Adversarially regularized autoencoders for generating discrete structures. *arxiv:1706.04223*, 2017a. (pp. 82.)

Zhao, S., Song, J., and Ermon, S. InfoVAE: Information maximizing variational autoencoders. *arxiv:1706.02262*, 2017b. (pp. 131.)

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. *ICCV*, 2017. (pp. 108.)

Zhu, L. L., Chen, Y., Yuille, A., and Freeman, W. Latent hierarchical structural learning for object detection. *CVPR*, 2010. (pp. 34.)