Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Lars Mushom

Disentangled Representations in Variational Autoencoders

Master's thesis in Applied Physics and Mathematics
Supervisor: Gunnar Taraldsen
June 2020

NTNU
Norwegian University of
Science and Technology

Lars Mushom

Disentangled Representations in Variational Autoencoders

Master's thesis in Applied Physics and Mathematics
Supervisor: Gunnar Taraldsen
June 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Summary

Deep generative models encompass models that combine a probabilistic framework with flexible deep neural networks that scale to high-dimensional data. Learning generative models that capture an interpretable representation of the vast amount of data remains a major challenge in machine learning. In this thesis, we investigate *Variational Autoencoders* (VAE), and their application to *disentangled representation learning*. We approach this model from a probabilistic perspective, viewed in the context of a latent variable model and approximate Bayesian inference. The thesis contains a clear and self-contained exposition of Variational Autoencoders.

Further, we consider four specific methods that aim to impose an interpretable structure in the latent variable in VAE. Such models aim to recover and disentangle the generative factors of variation in the data. We limit our scope to *unsupervised* methods. The methods, β-VAE, FactorVAE, β-TCVAE, and DIP-VAE, are thoroughly presented, highlighting their relative strengths and weaknesses. Moreover, we discuss the challenges of evaluating disentanglement and consider three supervised metrics which use the ground truth.

A common belief in unsupervised disentangled representation learning is the methods that can generalize across domains, and the representation is useful for downstream inference tasks. Using a variety of simple experimental setups, we take a sober and critical view of commonly accepted assumptions. Our results indicate that the considered models cannot be used to reliably learn disentangled representations in an unsupervised manner as we observe a lack of robustness to hyperparameters, random seeds, and domain shift. We also observe an insufficient ability to use unsupervised evaluation metrics, which we argue is a necessary condition for such models on real-world applications. Finally, we summarize the thesis and discuss directions for future research. We independently implement all methods, evaluation metrics, and analytics, and all the code is on GitHub. [1]

[1]https://github.com/larsmus/master-code

Sammendrag

Dype generative modeller omfatter modeller som kombinerer ideer fra sannsynlighetsteori med fleksible dype nevrale nettverk som skalerer til høy-dimensjonale data. Å lære generative modeller som fanger en representasjon store datamengder som vi kan tolke, er fortsatt en stor utfordring i maskinlæring. I denne oppgaven presenterer vi *Variational Autoencoders* (VAE) og dens anvendelse til å lære en *oppdelt representasjon*. Vi tar utgangspunkt i et probabilistisk perspektiv, sett i sammenheng med en latent variabel modell og tilnærmet Bayesisk inferens. Oppgaven inneholder en tydelig innføring i Variational Autoencoders.

Videre vurderer vi fire spesifikke metoder som å ønsker en tolkbar struktur i den latente variabelen i VAE. Slike modeller å finne de underliggende faktorene som genererer dataene. Vi begrenser omfanget vårt til ikke-veiledet metoder. Metodene, β-VAE, FactorVAE, β-TCVAE, og DIP-VAE, presenteres grundig med vekt på deres relative styrker og svakheter. Videre diskuterer vi utfordringene med å evaluere oppdeling og vurderer tre veiledet metrikker som bruker den underliggende sannheten.

En vanlig oppfatning er at ikke-veiledet læring av oppdelte representassjoner kan generalisere på tvers av domener, og representasjonen er nyttig for videre inferans. Ved å bruke en rekke enkle eksperimenter, stiller vi kristike spørsmål til allment aksepterte antagelser. Resultatene våre indikerer at modellene vi ser på ikke er tilstrekkelig pålitelig til å lære oppdelte representasjoner på en ikke-veiledet måte. Vi observerer en mangelfull robusthet overfor hyperparametre, tilfeldige seeds og domeneskifte. Vi observerer også en utilstrekkelig evne til å bruke ikke-veiledet metrikker, noe vi hevder er en nødvendig betingelse for å anvende slike modeller i praktiske applikasjoner. Avslutningsvis oppsummerer vi oppgaven og peker på mulige retninger for fremtidig forskning. Vi implementerer alle metoder, metrikker og analyser, og koden er tilgjengelig på GitHub. [2]

[2]https://github.com/larsmus/master-code

Preface

This thesis concludes my five years at the Norwegian University of Science and Technology (NTNU) and is the final work of my Master of Science degree in *Industrial Mathematics*. I will remember my time here with nothing but fondness. Furthermore, I want to express my gratitude for getting the chance to study abroad at the University of California, Berkeley, during my time at NTNU.

First, I would like to thank my supervisor, Professor Gunnar Taraldsen. Thank you for all our meetings and for allowing me to pursue a topic that I find truly exciting.

Most importantly, I want to thank my family for their unconditional and unwavering support, now, then and always. I want to thank all my friends and fellow students at NTNU for making my time in Trondheim truly special. I can't wait to see where life will bring us next.

Trondheim, 2020
Lars Mushom

List of Tables

List of Figures

Contents

Chapter 1

Introduction

In this thesis, we investigate a model called *Variational Autoencoders*, and their application to representation learning. In the last decade, machine learning has made tremendous progress in technologies such as image recognition, natural language processing, speech synthesis, and self-driving cars, making an increasing impact on our everyday life. While such methods gain an increased societal impact, it calls for *interpretable* models and algorithms. Learning an interpretable representation of the data is a step towards underling the inner workings and decisions from otherwise black-box models. Generative models provide a natural framework by inferring the data generating latent space, which implies capturing, to some extent, the salient characteristics of such data. Variational Autoencoder is a generative model that combines approximate Bayesian inference and deep neural networks.

The era of deep learning

Most of the state-of-the-art machine learning models today use *neural networks*; highly parameterized models that can represent complex patterns in data. Neural network models consisting of a hierarchical structure between input and output data carry the term *deep learning*. The term "neural network" originates from attempts at formalizing how the brain processes information [McCulloch and Pitts, 1943, Rosenblatt, 1958]. Regardless of biological plausibility, neural networks have proven itself useful across domains in representing complex data. However, difficulties in training neural networks limited their success early on. Innovations such as backpropagation [Rumelhart et al., 1986] and later stochastic optimizations techniques [Kingma and Ba, 2015, Duchi et al., 2010] made it feasible to train such models. Combined with advances in computing technology and available high-dimensional data, interest in deep learning has flourished, leading to disruptive new technologies. We will review the inner workings of relevant concepts in deep learning in Chapter 2.

The early paradigms in deep learning were mostly computational. Most breakthroughs were in designing function approximations for a given underlying data distribution, which also could be efficiently trained. The structure of a neural network is commonly termed its architecture. In the 1990s, researchers made advancements in modeling sequences, and a big challenge was to preserve long-term dependencies [Bengio et al., 1994]. Hochreiter and Schmidhuber [1997] introduced the recurrent neural network architecture called the long short-term memory (LSTM) network to deal with this challenge. To this day, LSTM is still used in state-of-the-art applications such as machine translation and other natural language processing tasks. While this was a significant breakthrough, neural networks could not live up to the expectation and other machine learning approaches such as kernel machines [Schölkopf and Smola, 2002] and graphical models [Jordan, 1999]. However, later neural networks made its comeback by emphasizing that deeper network could improve generalization while still be efficiently trained [Hinton et al., 2006, Bengio et al., 2007]. These breakthroughs led to a renewed popularity of deep neural networks that have continued to flourish.

Deep learning models thrived in the image domain after the introduction of convolutional neural networks (CNN). A CNN architecture named *AlexNet* [Krizhevsky et al., 2012] famously outperformed the competitors by a large margin in the yearly *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) in 2012. The task was to classify the 1.2 million images in the *ImageNet* dataset to 1000 different classes. The success of AlexNet and similar models proved the potential of deep learning, leading to a meteoric rise in the popularity of deep learning research. New software infrastructures such as Tensorflow [GoogleResearch, 2015] and Pytorch [Steiner et al., 2019] have made it easy for everyone to train deep learning models efficiently on GPUs, while benchmarks datasets have become publicly available. Big technology companies such as Apple, Google, Facebook, and Microsoft are pressing deep learning technologies close to their hearts, making it an integral part of their intellectual property. Deep learning technologies continuities to advance new territories with reinforcement learning systems playing the game Go [Silver et al., 2017] and text-to-speech applications [Oord et al., 2016], both by Deepmind, being some recent big leaps.

While achieving impressive results across domains, the theoretical foundation for deep learning is still immature. Areas such as statistics, probability theory, and information theory have a long history and deep theoretical roots, which have, in recent years, have fertilized the theoretical understanding of deep learning. Moreover, as deep learning models become more popular and widely used in consumer products, concepts such as uncertainty, robustness, and the causal relationship become increasingly prevalent. This has lead to an increased interest in combining Bayesian approaches with deep learning. Early Bayesian approaches did not scale well to large data but have regained attention in the light of deep learning. That way, one could achieve a probabilistic representation while still scale to larger models and datasets. This area of research, often called Bayesian Deep Learning. In particular, an area of Bayesian research that recently has become deep is generative models.

Generative Models

The key objective of generative modeling is to approximate the underlying probability distribution that generates the data we observe. In other words, if we observe the data $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and assume that it is generated from a unknown distribution with density $p^*(\mathbf{x})$, we want to find a tractable density that approximates $p^*(\mathbf{x})$. We usually work with a parametric density denoted by $p_\theta(\mathbf{x})$, which in turn means that we seek the parameter θ such that $p_\theta(\mathbf{x}) \approx p^*(\mathbf{x})$ for any observation $\mathbf{x}$. In a classification setting, a discriminative model will approximate the conditional density of some label given the data, while a generative model will approximate the joint density of the entire dataset. While there is a diverse set of applications of generative models, we can identify three fundamental goals when making inferences.

1. The first is concerned with density estimation, meaning that for any new observed sample $\mathbf{x}^*$ we want $p_\theta(\mathbf{x}^*)$ to be a good approximation to the true density.
2. Moreover, we want to be able to generate new samples. The model should be able to generalize to create new samples that were unseen in the dataset.
3. Lastly, we want to learn a representation of the data that is interpretable and useful.

Recent advances in parameterizing generative models using deep neural networks have enabled scalable modeling of high-dimensional data, including text [Devlin et al., 2018], speech [Oord et al., 2016] and images [Goodfellow et al., 2014, Kingma and Welling, 2013]. This research area, termed deep generative models, has become one of the most fruitful and rapidly evolving fields of modern statistical machine learning. In this thesis, we will focus on learning representations in deep generative models. In particular, we concentrate on *disentangled representations* of images using Variational Autoencoders.

Disentangled Representation Learning

Often, when the data is high-dimensional, approximating a data generating density involves embedding the data in lower dimensions. Data is often noisy, and it hard to interpret in its raw form. Learning a disentangled representation, imply recovering the *underlying factors of variation*. To gain an intuition of what that means, we consider a brief example. Imagine a dataset consisting of images of faces. While the dimension of a data point is large, say $3 \times 256 \times 256$, we can describe an image using fewer factors. Such high-level factors may be lightning conditions, hair color, facial expression, etc. Discovering the underlying structure of the data is a key element in approximating the generative density, and the objective is to recover and disentangle the factors in an interpretable form.

Current machine learning models excel at optimizing well-defined discriminative tasks from large i.i.d. data. Furthermore, most models require to be retrained for every new task. Humans can generalize across domains using a small amount of data and transfer knowledge between different tasks. We can re-use and re-purpose previous knowledge with local abstractions and new experiences. To do that, we reason in the forward, generative, and causal direction. Discriminative machine learning models go the other way by mapping observations to probable causes. To truly model the surroundings, machine learning models will also have to adapt across domains for non-i.i.d. data. In a sense, there is a gap between the predictions- based paradigm of traditional machine learning models and how humans abstract and reason. Learning a disentangled representation that captures the generative factors and their causal relationships are likely to generalize to such settings [Schölkopf, 2019].

A central concern about deep learning models is the ability to interpret the model and explain inferences done by the model. A way to counteract this problem is by considering simpler explainable models trained on an interpretable representation of the data, which may be learned by a deep learning model. Suppose we can obtain an entirely disentangled representation, which implied that we could approximate the data generating distribution to arbitrary accuracy using interpretable factors. These factors can then be used as features in a supervised setting on downstream tasks. The desired property is then that a simple model, for instance, a linear model in a regression case, is sufficient.

We discuss disentangled representation learning in more depth in Section 2.5.

Thesis structure

State-of-the-art approaches for unsupervised disentanglement learning are largely based on Variational Autoencoders. We aim to give a clear exposition to Variational Autoencoders (Chapter 3), before considering their specific application to disentangled representation learning (Chapter 4). We develop and study techniques for addressing a generative model with an interpretable latent structure. Concretely, the chapters are organized as follows.

- Chapter 2 gives an overview of the fundamental theoretical background. This includes the probabilistic framework, divergences between probability distributions, and the neural network machinery we use throughout this thesis. Furthermore, we will paint a broad picture of deep generative models and emphasize an information-theoretic approach to representation learning.
- Chapter 3 provides an introduction on the Variational Autoencoder. [Kingma and Welling, 2013, Rezende et al., 2014]. It starts from a latent variable model and describes how variational inference provides an optimization-based approach to approximate Bayesian inference. The flexibility of neural networks allows Variational Autoencoders to amortize the computation to scale in order to large datasets. This chapter can serve as a self-contained tutorial to Variational Autoencoders.
- Chapter 4 considers the problem of disentangled representations. We present the *β-VAE* [Higgins et al., 2019], and we show that it is closely related to the *information bottleneck* [Tishby et al., 2000]. Furthermore, we consider three additional models that modify of VAE to impose a disentangled structure in the latent variable [Chen et al., 2018, Kumar et al., 2017, Kim and Mnih, 2018], in addition to three different evaluation metrics.
- Chapter 5 contains a variety of experiments that aim to compare and challenge unsupervised VAE models that aim to recover a disentangled representation. *(i)* We reproduce β-VAE and provide additional detailed analysis of the model. *(ii)* We take a critical view of four different disentanglement models based on VAE. Based on lacking robustness to hyperparameters and randomness, we challenge the difficulties of model selection needed for practical deployment. *(iii)* We empirically study three supervised evaluation metrics and compare them to available unsupervised metrics. The results lead us to conclude that truthful unsupervised evaluation of disentanglement, a condition necessary for practical use, is an unsolved problem. *(iv)* Lastly, we consider the usefulness of disentanglement models in downstream inference tasks and domain adaption. Our results show that future work should investigate the concrete benefits of enforcing disentanglement on the learned representations.
- Finally, chapter 6 concludes and discusses the future outlook.

Chapter 2

Preliminaries

This chapter presents the fundamental theory that we'll employ in the rest of the thesis. Generative models are fundamentally concerned with probability distributions. We'll formally define probability spaces and density function, and we provide an overview of different approaches to compare two probability distributions. Before indulging in deep generative models, we'll introduce the essentials of neural networks. This leads us to representation learning, which we will explore from an information-theoretic viewpoint.

2.1 Probability Theory

In this section, we will review some essential concepts in probability theory that will be helpful later when we discuss ways of comparing distributions. A random experiment has three associated objects: a sample space, events, and a probability. The *sample space* Ω includes all conceptually possible outcomes. Outcomes are elements of the sample space, $\omega \in \Omega$, and may also be referred to as realizations or sample points. An event, A, is a particular subset of the sample space $A \in \Omega$, and the family of events is the σ-algebra $\mathcal{F}$ on Ω. A σ-algebra is a collection of subsets in Ω that satisfies the following criteria

1. The empty set and Ω is contained in $\mathcal{F}$.
2. $\mathcal{F}$ is closed under complementation.
3. $\mathcal{F}$ is closed under countable intersection.

A special kind of σ-algebra is Borel sets, $\mathcal{B}(\mathbb{R})$, which are generated by half-open intervals $(a, b]$ with $a < b$ on $\mathbb{R}$. The pair $(\Omega, \mathcal{F})$ is called a *measurable space* on which we can define a function $P : \mathcal{F} \to [0, 1]$ known as the probability measure. The probability P satisfies

1. $P(A) \geq 0$ for all $A \in \mathcal{F}$.
2. $P(\Omega) = 1$.
3. Whenever $A_1, A_2, \ldots$ are pairwise disjoint sets in $\mathcal{F}$,

$$P\left(\sum_{n=1}^{\infty} A_n\right) = \sum_{i=1}^{\infty} P(A_n).$$

The triple $(\Omega, \mathcal{F}, P)$ is a probability space. For a probability space we can define a *random variable* as a function $X : \Omega \to \mathcal{X}$ such that the inverse image $X^{-1}(B) \in \mathcal{F}$ for every Borel set $B \in \mathcal{B}(\mathbb{R})$. Equivalently, such a function is a random variable if $\{X \leq t\}$ is an event for every $t \in \mathbb{R}$. The *distribution* of a random variable is the associated probability and the *distribution function* $F : \mathbb{R} \to [0, 1]$ is the special case $F(t) = P(\{X \leq t\})$. A probability measure is *absolutely continuous* if there exists a *density function* f such that

$$P(\{X \in (a, b]\}) = \int_a^b f(t)dt,$$

for every interval $(a, b]$ on $\mathbb{R}$. If we have two probability measures P and Q, we say that P is absolutely continuous with respect to Q, denoted by $P \ll Q$, if there is a function g such

$$P(A) = \int_A g d\mathrm{Q},$$

for every Borel subset A. The function g is called the Radon-Nikodym derivative. For further details on probability theory, we refer to Jacod and Protter [2012].

Notation

The subject matter in this thesis encompasses multiple fields of study, each with their conventions in notation. We have used the following notation throughout the thesis. Calligraphic letters, such as $\mathcal{X}$, will denote sets. Capital, italic letters such as X will be random variables, while lower case x will be the corresponding instance. We denote a probability distribution by a capital, roman letter P, and the density will be lower case $p(x)$. It is common, especially in the machine learning literature, to not make an explicit distinction between random variables as upper-case and the values as lower-case. We'll adopt this practice to in Chapter 3 and 4 to be in line with the existing literature. The distinction will always be clear from the context. We'll use boldface for vectors $\mathbf{x}$, and subscripts will either denote the indices in a set as $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ or an element in a vector x_i.

2.2 Divergences

A fundamental problem in information theory and statistics is the notion of distance between probability distributions. This problem is especially relevant when investigating if a proposed stochastic model is a satisfactory approximation to the real underlying model. A *divergence* D is a semimetric, meaning that it satisfies *Gibb's inequality* $D(\mathrm{P}, \mathrm{Q}) \geq 0$ with equality if and only if $\mathrm{P} = \mathrm{Q}$, where P and Q are probability distributions. Another commonly used name for divergences is *probability metrics*. Note that the divergence does not have to be symmetric or satisfy the triangle inequality as opposed to a regular metric. Classical applications of divergences include hypothesis testing, compression, limit theorems, and empirical process theory. High-dimensional data are often modeled in terms of a complex probability distribution rich enough to cover the observed diversity. However, it may not be possible to compute the characteristics of such a complex distribution explicitly. Due to computational constraints, we often consider a smaller stochastic model described by some parameters as sufficient approximation to a more complex model. Then a natural question is the magnitude of the error to tolerate when using the stochastic model. In statistical machine learning, we are interested in minimizing the error between the parametric stochastic model and the target distribution in an efficient and tractable manner. In the following sections, we'll review some of the well-known divergences and their key properties used in generative models.

f-divergences

A general class of functions that measures the difference between two distributions are *f-divergences*, introduced independently in [Ali and Silvey, 1966] and [Csiszár, 1972], and the class encompasses well-known divergences such as the Kullback-Leilbler divergence and the total variation distance. Given two probability distributions P, Q defined on $\mathcal{X}$ such that P is absolutely continuous with respect to Q, and a convex function f such that $f(1) = 0$, the f-divergence $D_f(\mathrm{P}\|\mathrm{Q})$ is defined by

$$D_f(\mathrm{P}\|\mathrm{Q}) = \int_{\mathcal{X}} f\Big(\frac{d\mathrm{P}}{d\mathrm{Q}}\Big) d\mathrm{Q} = \mathbb{E}_{\mathrm{Q}}\Big[f\Big(\frac{d\mathrm{P}}{d\mathrm{Q}}(X)\Big)\Big], \tag{2.1}$$

Here $\frac{d\mathrm{P}}{d\mathrm{Q}}$ is the Radon-Nikodym derivative. The Radon-Nikodym derivative is the likelihood ratio between the distributions. The expression can be rewritten in terms of the densities by $p(x) = \frac{d\mathrm{P}(x)}{d\mu}$ and $q(x) = \frac{d\mathrm{Q}(x)}{d\mu}$, where μ is the dominating measure, giving

$$D_f(\mathrm{P}\|\mathrm{Q}) = \int_{\mathcal{X}} f\Big(\frac{p(x)}{q(x)}\Big) q(x) d\mu(x).$$

Let Y be a function of the data, $Y = \phi(X)$. Then the f-divergence satisfy the data processing inequality [Liese and Vajda, 2006]

$$D_f(\mathrm{P}_X \| \mathrm{Q}_X) \geq D_f(\mathrm{P}_Y \| \mathrm{Q}_Y).$$

This tells us that processing the data through some physical operation cannot increase the information, and that information generally is lost. We can relate the f-divergence to entropy by

$$\begin{aligned} D_f(\mathrm{P}\|\mathrm{Q}) &= \mathbb{E}_{\mathrm{Q}}\Big[f\Big(\frac{d\mathrm{P}}{d\mathrm{Q}}(X)\Big)\Big] \\ &= \mathbb{E}_{\mathrm{Q}}\Big[f\Big(\frac{d\mathrm{P}}{d\mathrm{Q}}(X)\Big)\Big] - f\Big(\mathbb{E}_{\mathrm{Q}}\Big[\frac{d\mathrm{P}}{d\mathrm{Q}}(X)\Big]\Big) \\ &= \mathbb{H}_f\Big(\frac{d\mathrm{P}}{d\mathrm{Q}}(X)\Big), \end{aligned}$$

which is the ϕ-entropy where $f = \phi$ [Wainwright, 2019, p. 58]. If we choose $\phi(x) = x^2$, the ϕ-entropy is the variance, and generally other choices of ϕ reflects a notion of spread. Loosely speaking, the f-divergence can therefore be interpreted as variance-like quantity of the likelihood ratio.

To evaluate the f-divergence, we need the densities of both P and Q. In generative models we want to approximate an underlying distribution P with Q. However, $D_f(\mathrm{P}\|\mathrm{Q})$ is infeasible since P is only accessible through samples. An important property of f-divergences in this context, is that they can be represented in a variational form making it possible to do convex empirical risk minimization [Nguyen et al., 2010]. To establish this result, we define the Fenchel conjugate dual function of a convex function f as,

$$f^*(v) = \sup_u \{uv - f(u)\}.$$

Then f^* is also convex and by definition $f(u) = \sup_v\{uv - f^*(v)\}$. Using the conjugate dual function, we write the f-divergence as

$$\begin{aligned} D_f(\mathrm{P}\|\mathrm{Q}) &= \int \sup_T \Big\{T\frac{d\mathrm{P}}{d\mathrm{Q}} - f^*(T)\Big\} d\mathrm{Q} \\ &\geq \sup_{T\in\mathcal{T}} \Big\{\int T d\mathrm{P} - f^*(T) d\mathrm{Q}\Big\} \\ &= \sup_{T\in\mathcal{T}} \{\mathbb{E}_{\mathrm{P}}[T(X)] - \mathbb{E}_{\mathrm{Q}}[f^*(T(X))]\}, \end{aligned} \tag{2.2}$$

where $\mathcal{T}$ is an arbitrary function class with $T : \mathcal{X} \to \mathbb{R}$. The inequality follows from Jensens inequality and the fact that we restrict the function to $\mathcal{T}$ from all measureable functions. By differentiating the with respect to T and setting to zero and using the property of the Fenchel conjugate $x = f'(f^{*'}(x))$, the variational representation of f-divergences is tight for

$$T^*(x) = f'\left(\frac{d\mathrm{P}}{d\mathrm{Q}}\right).$$

The f-divergence has an interesting interpretation in terms of a binary classifier. Consider the classification problem where X is a random variable on $\mathcal{X}$ and $Y \in \{-1, +1\}$ is a binary random variable. Let μ be the Borel probability measure on the product space. For a loss function $L : \{-1, +1\} \times \mathbb{R} \to \mathbb{R}$, the optimal risk is defined as

$$R^L_{\mathcal{F}} = \inf_{f\in\mathcal{F}} \int L(y, f(x)) d\mu(x, y).$$

Let P and Q be the distributions conditioned on $Y = 1$ and $Y = -1$. Then Nguyen et al. [2009] showed for each loss function L, there is one unique corresponding f-divergence with $R^L_{\mathcal{F}} = -D_f(P\|Q)$.

We will encounter several f-divergences in the context of deep generative models, and we'll review some of the most important ones next.

Kullback-Leibler divergence

Another name for the Kullback-Leibler (KL) divergence is relative entropy. When $f(t) = t \log t$, we have

$$D_{\mathrm{KL}}(P\|Q) = \int_{\mathcal{X}} \log\left(\frac{d\mathrm{P}}{d\mathrm{Q}}\right) d\mathrm{P}.$$

Note that the KL divergence is in general not symmetric, and therefore not a metric. The KL divergence displays an attractive decoupling feature in that if $\mathrm{P}^{1:n}$ and $\mathrm{Q}^{1:n}$ denotes the corresponding product distributions on the product space, we have

$$D_{\mathrm{KL}}(\mathrm{P}^{1:n}\|\mathrm{Q}^{1:n}) = \sum_{i=1}^{n} D_{KL}(\mathrm{P}_i\|\mathrm{Q}_i),$$

which follows directly from the definition. Minimizing the KL divergence between a density estimator and the underlying distribution is asymptotically equivalent to maximum likelihood estimation. Suppose we have data $\mathcal{D} = \{x_i\}_{i=1}^{N}$ with density $p(x)$, and we have a model density q_θ. The *likelihood function* is the defined as

$$L_N(\theta; \mathcal{D}) = \prod_{i=1}^{N} q_\theta(x_i).$$

Note that the likelihood function is a function of the parameter θ, and not the data $\mathcal{D}$. The *maximum likelihood estimator* (MLE), $\hat{\theta}_{MLE}$, of the underlying parameter θ is defined to be the value that maximizes the likelihood. Intuitively, this is the parameter that maximized the probability of observing the data we are given under the assumed parametric model. For computational convenience it is common to work with the average log-likelihood function $l_N(\theta; \mathcal{D}) = \frac{1}{N} \log L_N(\theta; \mathcal{D})$. The MLE benefits from several preferable properties such as consistency and asymptotic efficiency [Keener, 2009]. So if P is the true distribution with density p and Q is an approximate distribution with parametric density q_θ, then

$$\begin{aligned}
\arg\min_\theta D_{\mathrm{KL}}(\mathrm{P}\|\mathrm{Q}) &= \arg\min_\theta \mathbb{E}_{\mathrm{P}}[\log p(X) - \log q_\theta(X)] \\
&= \arg\max_\theta \mathbb{E}_{\mathrm{P}}[\log q_\theta(X)] \\
&= \lim_{N\to\infty} l_N(\theta; \mathcal{D}),
\end{aligned}$$

which is the maximum likelihood estimator in the asymptotic limit. Note that the last line follows from the law of large numbers.

Reverse Kullback-Leibler divergence

The KL divergence is not symmetric, and $D_{\mathrm{KL}}(Q\|P)$ have different properties and use-cases than $D_{\mathrm{KL}}(P\|Q)$. In in the reverse case, the generating function is $f(t) = -\log t$ and $Q \ll P$. The fact that the reverse KL-divergence is also a f-divergence follows from a general result that states that for any f-divergence $D_f(\mathrm{P}\|\mathrm{Q})$, there exists a function $\hat{f}$ such that $D_{\hat{f}}(\mathrm{P}\|\mathrm{Q})$ is also a f-divergence with $D_f(\mathrm{P}\|\mathrm{Q}) = D_{\hat{f}}(\mathrm{Q}\|\mathrm{P})$ given that $\mathrm{Q} \ll \mathrm{P}$ [Liese and Vajda, 2006]. We will empirically shows the different properties of the regular KL and the reverse KL divergence in Section 3.2.

Jensen-Shannon divergence

The Jensen-Shannon divergence (JSD) is expressed in terms of the KL-divergence,

$$D_{\mathrm{JSD}}(\mathrm{P}\|\mathrm{Q}) = \frac{1}{2} D_{KL}(\mathrm{P}\|\frac{\mathrm{P}+\mathrm{Q}}{2}) + \frac{1}{2} D_{KL}(\mathrm{Q}\|\frac{\mathrm{P}+\mathrm{Q}}{2}).$$

Note that the Jensen-Shannon divergence is an average of the regular KL and reverse KL, and it can interpolate the behavior of both. The Jensen Shannon divergence is symmetric, as opposed to the KL-divergence. The generating function is $f(t) = -(u+1)\log\frac{u+1}{2} + u\log u$. Later we will see that the Jensen-Shannon divergence can be efficiently estimated and has a key part in designing deep generative models.

Integral Probability Metrics

Intuitively, a natural way to measure the distance between two distributions is measuring the discrepancy between their expectations of a function. A *integral probability metric* (IPM) is measuring the maximum mean discrepancy for a function class $\mathcal{F}$ that contains functions $f : \mathcal{X} \to \mathbb{R}$ that are integrable with respect to P and Q,

$$D_{\mathcal{F}}(\mathrm{P}, \mathrm{Q}) = \sup_{f \in \mathcal{F}} \left\{ \int_{\mathcal{X}} f(d\mathrm{P} - d\mathrm{Q}) \right\} = \sup_{f \in \mathcal{F}} \left\{ \mathbb{E}_{\mathrm{P}}[f(X)] - \mathbb{E}_{\mathrm{Q}}[f(X)] \right\}. \tag{2.3}$$

Note that in some contexts IPMs are written with an absolute value, however, the expression stated here (2.2) will never be negative since we can choose f to always map to zero. IPMs were first introduced as a unifying framework in [Müller, 1997], extensively studied in [Rachev et al., 2013], and connections to f-divergences and binary classifications were explored in [Sriperumbudur et al., 2009]. Several well known probability measures can be obtained by restricting the function class $\mathcal{F}$.

Wasserstein Metric

A function is L-Lipschitz with respect to the metric ρ if

$$|f(x) - f(y)| \leq L\rho(x, y) \quad \text{for all } x, y \in \mathcal{X},$$

and let $\|f\|_L$ be the smallest L that satisfies the inequality. Then the Wasserstein metric is obtained when choosing $\mathcal{F} = \{f : \|f\|_L \leq 1\}$,

$$D_{\mathrm{W}}(\mathrm{P}, \mathrm{Q}) = \sup_{\|f\|_L \leq 1} \left\{ \int_{\mathcal{X}} f(d\mathrm{P} - d\mathrm{Q}) \right\}. \tag{2.4}$$

The Kantorovich-Rubinstein theorem states that the Wasserstein distance can be equivalently stated as a coupling-based distance [Dudley, 2002, Theorem 11.8.2]. Let the distribution M be a coupling of (P, Q) on the product space $\mathcal{X} \times \mathcal{X}$. This means that $\mathrm{M}(\mathrm{P}, \mathrm{Q})$ is the set of joint distributions that have the marginal distributions P and Q. Now for a 1-Lipschitz function f, we have

$$\int_{\mathcal{X}} f(d\mathrm{P} - d\mathrm{Q}) = \int_{\mathcal{X} \times \mathcal{X}} (f(x) - f(y)) d\mathrm{M}(x, y) \leq \int_{\mathcal{X} \times \mathcal{X}} \rho(x, y) d\mathrm{M}(x, y)$$

If we minimize over the possible couplings M, the theorem states an equivalent dual formulation of the Wasserstein metric

$$D_{\mathrm{W}}(\mathrm{P}, \mathrm{Q}) = \inf_{\mathrm{M}} \int_{\mathcal{X} \times \mathcal{X}} \rho(x, y) d\mathrm{M}(x, y) = \inf_{\mathrm{M}} \mathbb{E}_{\mathrm{M}}[\rho(X, Y)]. \tag{2.5}$$

The Wasserstein metric is closely linked to the theory of optimal transport [Peyré and Cuturi, 2019]. The following practical example can illustrate the fundamental problem in optimal transport. Suppose a mathematician is at the beach, and she wants to build a sandcastle. In front of her is a pile of sand, and the goal is to transform this pile into the shape of a castle, with the same volume as the pile, in a minimum amount of work. Now let $d\mathrm{M}(x, y) = m(x, y)dxdy$ where $m(x, y)$ is the joint density, and think of the pile and the castle as the two marginal probability distributions P and Q. For each grain of sand, she wants to move it a distance $\rho(x, y)$ from dx to dy. The joint distribution $m(x, y)$ is the transportation strategy. By integrating over the whole domain, we get the total work performed in building the castle. The mathematician wants to minimize the work with respect to the transportation strategy $m(x, y)$. The amount of work done by the mathematician with the optimal transportation strategy is the Wasserstein metric. Due to this interpretation, another name of the Wasserstein metric is the *earth mover distance*.

Total variation

If we choose the function class to be $\mathcal{F} = \{f : \|f\|_\infty \leq 1\}$, we get the the total variation metric. By the definition of the norm, we have that $|f(x) - f(y)| \leq 1$ for all $x, y \in \mathcal{X}$. Moreover, since the difference is invariant to constant shifts, the total variation metric is defined by $f(x) \in [0, 1]$ for all $x \in \mathcal{X}$,

$$D_{\mathrm{TV}}(\mathrm{P}, \mathrm{Q}) = \sup_{f : \mathcal{X} \to [0,1]} \int f(d\mathrm{P} - d\mathrm{Q}).$$

The total variation metric has an alternative formulation in terms of the maximum difference in probability for an event A [Levin and Peres, 2017, Proposition 4.2],

$$D_{\mathrm{TV}}(\mathrm{P}, \mathrm{Q}) = \sup_{A \in \mathcal{X}} |\mathrm{P}(A) - \mathrm{Q}(A)|,$$

where A is a measurable subset. Alternatively, we can express the total variation in the setting of optimal transport using the coupling M between the two distribution P and Q. The Wasserstein metric associated with the Hamming distance $\rho(x, y) = \mathbf{1}_{x \neq y}$ yields the total variation [Levin and Peres, 2017, Proposition 4.7]

$$D_{\mathrm{TV}}(\mathrm{P}, \mathrm{Q}) = \inf_{\mathrm{M}} \mathrm{E}_{\mathrm{M}}[\mathbf{1}_{X \neq Y}] = \inf \mathrm{M}(X \neq Y).$$

Interestingly, the total variation can also be formulated as a f-divergence with $f(t) = |t - 1|$. In [Sriperumbudur et al., 2009] they show that the total variation is the only nontrivial probability metric that is both a f-divergence and an IPM, establishing that the two families of distances are essentially different.

Maximum Mean Discrepancy

The maximum mean discrepancy (MMD) [Gretton et al., 2012] is defined over the function class $\mathcal{F} = \{f : \|f\|_{\mathcal{H}} \leq 1\}$, where $\mathcal{H}$ represents reproducing kernel Hilbert space (RKHS). Here we will first give a brief introduction to RKHS following [Wainwright, 2019, Chapter 12] and then present some key properties of the MMD. First we recall from functional analysis that a Hilbert space $\mathcal{H}$ is a inner product space where every Cauchy sequence converge to some element $f \in \mathcal{H}$. Let $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ denote the associated inner product. Riesz representation theorem tells us that for a continuous linear functional, $L(f) : \mathcal{H} \to \mathbb{R}$ belonging to the dual space of $\mathcal{H}$, there exists a unique feature mapping g such that $L(f) = \langle f, g \rangle_{\mathcal{H}}$. In a RKHS the feature map is a positive semidefinite kernel on the form $g(x) = k(x, \cdot)$, meaning that for data $\{x_i\}_{i=1}^{N}$, the $N \times N$ matrix $K_{ij} = k(x_i, x_j)$ is positive semidefinite. Now we extend the notion of the feature map to embeddings in a probability distributions. Let $\mathbb{E}_{\mathrm{P}}[f] = \langle f, \mu_{\mathrm{P}} \rangle$ denote the mean embedding for a distribution P. We can then write the MMD as

$$\begin{aligned} D_{\mathrm{MMD}}(\mathrm{P}, \mathrm{Q})^2 &= \left[\sup_{f \in \mathcal{H}} \{\mathbb{E}_{\mathrm{P}}[f(X)] - \mathbb{E}_{\mathrm{Q}}[f(X)]\} \right]^2 \\ &= \left[\sup_{f \in \mathcal{H}} \{\langle \mu_{\mathrm{P}} - \mu_{\mathrm{Q}}, f \rangle\} \right]^2 \\ &= \|\mu_{\mathrm{P}} - \mu_{\mathrm{Q}}\|_{\mathcal{H}}^2. \end{aligned}$$

From this expression, we see that the MMD is zero if $\mu_{\mathrm{P}} = \mu_{\mathrm{Q}}$ and Gretton et al. [2012] show that it is equivalent. Writing out the norm and using the definition of the mean embedding and the kernel, we get a form of the MMD that is convenient for Monte Carlo estimation,

$$D_{\mathrm{MMD}}(\mathrm{P}, \mathrm{Q})^2 = \mathbb{E}\left[k(X, X') - 2k(X, Z) + k(Z, Z')\right], \tag{2.6}$$

where $X, X' \sim \mathrm{P}$ and $Z, Z' \sim \mathrm{Q}$. Note that these are independent copies from their respective distributions. The empirical estimates follow by using the corresponding empirical mean and two-sample U-statistics yielding an unbiased estimator [Vaart, 1998, Chapter 12]. Gretton et al. [2012] goes on investigating the MMD as a test statistic in the hypothesis testing problem where the null hypothesis is $\mathrm{P} = \mathrm{Q}$ and derives the computational cost and concentration bounds. The same authors show that the MMD displays an attractive moment matching property. This way, we can do density estimation by projecting the data into the RKHS with the mean feature map. Choosing the kernel will allow the matching of higher-order moments.

2.3 Deep learning

The essentials of deep learning

A neural network aims to provide a flexible function that can represent highly complex and non-linear relationships in the data, while still being able to fit such a function with reasonable resources. While neural networks may be perceived with some sense of mystery due to its initial biological inspiration and the surge of impressive applications, a neural network can, in its simplest form, a non-linear generalization of a linear model. Given a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, a linear model seek an approximation as an affine transformation of the input data, $\hat{y}_i = \mathbf{w}^T\mathbf{x}_i + b$. Here we refer $\mathbf{w}$ as the *weights* and b as the *bias*. To capture non-linear relationships in the data we could transform the input data by some basis functions $\phi_1, \ldots, \phi_d$ such that the model is on the form

$$\hat{y}_i(\mathbf{x}_i, \mathbf{w}, b) = g\left(\sum_{j=1}^{d} w_j \phi_j(\mathbf{x}_i) + b\right). \tag{2.7}$$

Here $g(\cdot)$ is called the *activation function* that operates element-wise on the vector. When we fix the basis functions, the model (2.7) is known as non-linear regression, and if the activation function is the identity, we have a linear model in the basis functions. However, in the case of a neural network, we want the basis functions to depend on parameters that can be optimized jointly with the weight and bias parameters. This leads to the key element in a neural network, which is that the basis functions itself are in the form of (2.7). The basis functions are a non-linear function of an affine combination of the inputs. The weights and bias in the affine combination are also parameters that can be optimized. The model can then be written as a repeated composition between non-linear functions and affine transformations, allowing for the model to capture highly non-linear relationships in both data and parameters.

The operation in (2.7) is what is known as a *neuron* in a neural network. We can easily generalize the model described in the previous section to a structure of neurons called the feed-forward neural network. A group of neurons operating at the same depth in the network is called a *layer*. In a fully connected feed-forward neural network, each neuron in a layer gets its input from every neuron in the previous layer, and the output connects to the next layer. Mathematically speaking, each layer has an associated weight matrix $\mathbf{W} \in \mathbb{R}^{n^{[l]} \times n^{[l-1]}}$, where $n^{[l]}$ is the number of neurons in the l-th layer. The output from a layer, or its activation is then given by $\mathbf{a}^{[l]} = g^{[l]}(\mathbf{W}^{[l]}\mathbf{a}^{[l-1]} + \mathbf{b}^{[l]})$. If the neural network consists of N layers, we say that the depth of the network is N, and the layers indexed by $2, \ldots, N-1$ are called *hidden layers*.

As we go through the layers, we repeatedly apply non-linear transformations to the data by choosing the activation function. A neural network that solely uses the identity function as the activation functions is essentially a linear model. Hence we need to find non-linear activation functions. Furthermore, as we will see later, neural networks use gradient information to train, so the activation function need to be differentiable almost everywhere. Typically, the activation function is chosen to be a function that is applied element-wise. Popular choices of activation functions for hidden layers are the rectified linear unit (ReLU) $g(x) = \max\{0, x\}$ and the sigmoid function $g(x) = \sigma(x) = \frac{1}{1+e^x}$. Another popular activation function that we will use later is Leaky ReLU, $g(x) = \max\{cx, x\}$, where c is a small scalar, typically 0.02.

The network we have described is a feed-forward network since the information flows from the input and sequentially through each layer to the output with no closed directed cycles. It is fully connected since each neuron connects to every neuron in the previous layer in the subsequent layer. We display such a network as a directed graphical model in Figure 2.1. However, it is possible to design a completely different structure and topology. The structure of a neural network is commonly known as its *architecture*. There exist several results on their ability of a feed-forward network to approximate arbitrary functions. Hornik [1991] showed that a feed-forward architecture with a single hidden layer could uniformly approximate on a compact input domain to an arbitrary accuracy for a wide range of activation functions and a linear output layer. However, the downside is that the network needs to be exponentially wide. This result is known as a universal approximation theorem. In practice, a deeper network typically achieves better results on a wide variety of tasks. More recently, Lu et al. [2017] proved a universal approximation theorem for networks with bounded width and ReLU as the activation function. While it is reassuring that a universal approximation theorem exists, there is no guarantee that we can estimate any function. As we will see next, the optimization problem and overfitting pose as obstacles in finding the optimal function.

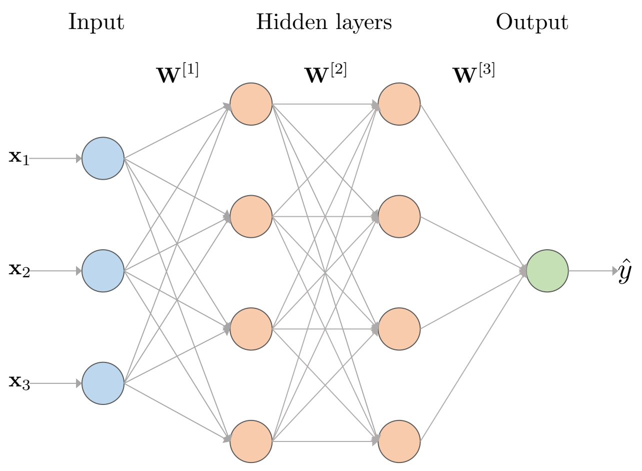


Figure 2.1: Graphical representation of a neural network with two hidden layers.

Training

Now that we have established neural networks as a general parametric nonlinear class of functions, a natural question is how to estimate the parameters in the network. In a neural network with L layers, the parameters are $\theta = (\mathbf{W}^{[l]}, b^{[l]})_{l=1}^{L}$. For a given set a parameters we can evaluate the neural network by passing the feature vector $\mathbf{x}$ through the network to get an estimate $\hat{y}$. The process of evaluating the neural network is termed *forward propagation*. Using the framework of empirical risk minimization, we specify a loss function $L_\theta(y, \hat{y})$ measuring the loss when using the estimate instead of the true value. The risk is then the expected loss with respect to the data distribution $R(\theta) = \mathbb{E}\left[L_\theta(y, \hat{y})\right]$. However, since the distribution of the data is unknown we use the empirical estimate $\hat{R}(\theta) = \frac{1}{N}\sum_{i=1}^{N} L_\theta(y_i, \hat{y}_i)$ from the dataset $\mathcal{D}$. A common choice for the loss function is the L_2 distance $L_\theta(y, \hat{y}) = (y - \hat{y})^2$.

The question of estimating the parameters in a neural network is widely studied and is still an active research area today. However, most algorithms are based on some form of gradient descent. Gradient descent is an iterative scheme that updates the parameters in the direction of the negative gradient on the following simple form,

$$\theta_{i+1} = \theta_i - \alpha \nabla \hat{R}(\theta_i).$$

Here α is the step size of the updates, commonly called the *learning rate*. Note that since a neural network is non-convex, we are not guaranteed to find a global minimum, and a local minimum usually has to suffice. Too small learning rates might imply that we don't make any progress, while we risk overstepping with a large learning rate. We say that the gradient vanishes when the gradient approaches zero, and we're not able to make any progress. Typically only first-order optimization techniques are used since higher-order techniques are often too costly due to the high number of parameters and the size of datasets. The *backpropagation* algorithm [Rumelhart et al., 1986] is used to compute the value of the gradient. The algorithm exploits the nested structure of a neural network by using the chain rule and dynamical programming to compute the gradient.

Instead of using the entire dataset to estimate the gradient of the risk, it is possible to use a subset of samples, which leads us to the idea of stochastic optimization. In stochastic gradient descent (SGD), the gradient estimate uses only a single sample drawn uniformly from the dataset. Using a simple sample leads to an estimate with higher variance, but that is substantially more efficient to compute. Despite its simplicity, SGD is still a popular method in state-of-the-art applications. We can reduce the variance in the gradient by averaging over multiple samples, in what is called *mini-batching*. The size of a mini-batch is called the *batch size*, and it is typically of the order $10 - 10^3$. *Batch normalization*, a popular technique that significantly improves the optimization of neural networks, applied the standard score $\frac{X-\mu}{\sigma}$ to any layer across the mini-batch. Several other stochastic optimization algorithms exist as well, most notably *Adam* [Kingma and Ba, 2015], which uses an exponential moving average of the first and second moment of the gradient in the estimation.

We assume that the data that we observe follows an unknown distribution and that all observations contain some noise. A model estimated from a set of observations coming from this distribution should behave similarly as when faced with new and unseen data from the same distribution but absent from the dataset. This property is called the *generalization* of the model. When the model is highly flexible with a large number of parameters, it can fit the noise of the data and not the underlying structure of the distribution. This is known as *overfitting*. There is a range of regularization techniques that can be applied to prevent overfitting. A classical way of doing this is to add a L^2 or L^1 penalty to the cost function, called ridge and lasso regularization, respectively, in the statistical learning literature. Another popular regularization technique in neural networks is *dropout*. At each layer in each iteration in training, we set some neurons to zero randomly with a certain probability. This prevents weight parameters from being too large and ensures that the model spreads information across the nodes. Empirically, the network is then more robust to variations in the data and reduces overfitting.

Convolutional neural networks

Convolutional neural networks (CNN) is a special kind of neural network where the convolution operator replaces the general matrix multiplication in at least one layer. In general terms, the convolution between the functions f and g is defined as

$$(f * g)(x) = \int f(y)g(x-y)dy.$$

If g is a probability distribution, we can interpret this as the weighted average of f with respect to g, where the weights are larger closer to the argument x. In the context of CNNs, f is typically called *input*, g is called the *kernel* and the output is referred to as the *feature map*. Of course, for integer-valued functions, the integral is a sum.

In a CNN, it is common to make the kernel have a smaller range than the input such that the weights will be sparse. We illustrate this in Figure 2.2, where the kernel where the highlighted nodes denote the range of the upmost node. Note that in a deeper network with repeated convolutional layers, a node deep in the network can indirectly be connected to most of the input data. Using convolutional layers also reduces the number of parameters needed as opposed to a fully connected network. Matrix multiplication requires $m \times n$ with m inputs and n inputs, while a convolutional operation only requires $k \times n$ parameters where k is the size of the kernel.

CNN's are well suited for grid-like structures such as images. The supremacy of CNN in the image domain manifested itself in 2012 in the ImageNet classification contest. The winning model named AlexNet [Krizhevsky et al., 2017] was based on CNN and it outperformed its competitors by a large margin. Since then, CNN has become a standard building block for any model in the image domain.

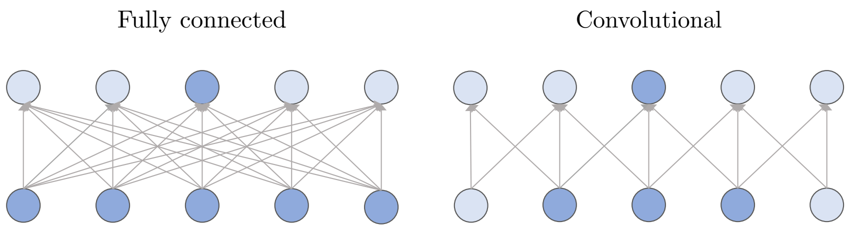


Figure 2.2: Comparison between a fully connected layer and a convolutional layer.

2.4 Deep generative models

The idea of generative models has existed for a long time in the statistical literature. Still, as with several other research areas, it has resurged as deep learning has gained traction and popularity. Deep generative models aim to combine probabilistic models with the scalability and flexibility of deep learning models. Over the last six years, several different approaches to deep generative models have surfaced, and we can loosely classify them in four different types of models.

- Generative Adversarial Networks [Goodfellow et al., 2014].
- Variational Autoencoders [Kingma and Welling, 2013].
- Normalizing Flows [Rezende and Mohamed, 2015a].
- Autoregressive Models [Larochelle and Murray, 2011].

Each class of models has advantages and drawbacks. Generative Adversarial Networks (GAN) have achieved great success in image generations, produces realistic-looking images not seen in the dataset. Since its introduction in 2014, GAN models have rapidly improved by developing the model's architecture, making it possible to generate more realistic images with higher resolution. In Figure 2.3 we see samples from a current state-of-the-art model in image generation called *StyleGAN* [Karras et al., 2018]. The model, created by NVIDIA, displays samples that are nearly impossible to distinguish from real images.

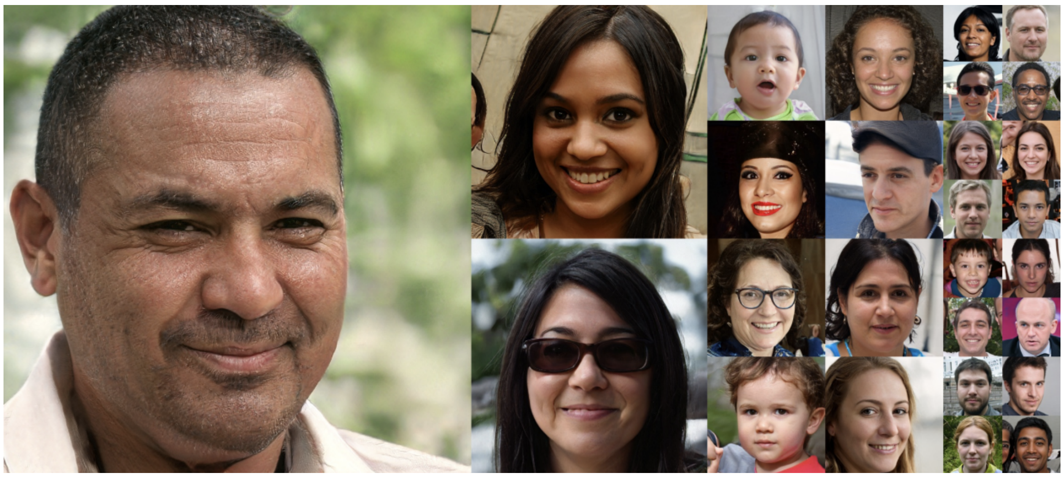


Figure 2.3: Images produced by the generator of StyleGAN trained on the Flickr-Faces-HQ dataset. The figure is taken directly from the original paper [Karras et al., 2018].

Autoregressive models have proven to be state-of-the-art in the processing of high-dimensional sequential data such as speech. Oord et al. [2016] uses this kind of model in *WaveNet*, a model that generates realistic speech from text and is the core basis of Google Assistant. Variational Autoencoders has its strength in explicitly learning a lower-dimensional representation of the data, providing a compression of the data that can be interpreted as higher-order features. Lastly, normalizing flows show their core capabilities in density estimation, where they are a specified density for all possible observations.

The classic approach to generative modeling in statistics is to utilize the likelihood function. Such models approximate some true density $p^*(\mathbf{x})$ by a parametric density $p_\theta(\mathbf{x})$ by maximizing the log-likelihood $l_n(\theta; \mathbf{x}) = \log p_\theta(\mathbf{x})$ function. Maximum likelihood estimation is widely studied, has several appealing properties such as asymptotic consistency, normality, and efficiency [Keener, 2009, Chapter 8], and most models in statistics and machine learning uses the likelihood function. In such setting settings, the parametric density is specified explicitly for the full support of the data, and this is defined as a *prescribed statistical model* by Diggle and Gratton [1984]. In contrast, an *implicit statistical model* is defined by a stochastic mechanism that generates data from an approximate distribution. Note that an implicit model does not necessarily have a defined density over the full support of the data. A generative model in high dimension will typically construct the distribution using a latent variable $\mathbf{z}$ following a prior distribution $p(\mathbf{z})$, which are then transformed by a deterministic function g_θ. The marginal parametric density of $\mathbf{x}$ is then $p_\theta(\mathbf{x}) = \int p(\mathbf{z}) p_\theta(\mathbf{x}|\mathbf{z})$. However, in a high-dimensional setting where g_θ is typically a complex non-linear function such as a neural net, the integral is intractable, and the likelihood function is unavailable. Intractability is also a problem for implicit models, but this motivates the techniques of *likelihood-free* estimation.

Generative Adversarial Networks

Generative Adversarial Networks (GAN) were first introduced in [Goodfellow et al., 2014]. A GAN consists of two neural networks, one generative model and one discriminative model. The generator tries to replicate samples from the data generating distribution. In contrast, the discriminator tries to tell the difference between real samples that are observed and fake samples from the generator. In Figure 2.4, we see a graphical representation of a GAN model. A GAN model estimates the distribution by samples, and it cannot explicitly evaluate the density function. It is, therefore, an implicit generative model.

More formally, let Z denote some latent random variable with prior distribution P_z and let X be a random variable underlying, unknown distribution P^* that generates the data. Let P^* and P_θ have the corresponding densities $p^*(\mathbf{x})$ and $p_\theta(\mathbf{x})$. The generator defines a differentiable mapping from the latent space into the data space $G : \mathcal{Z} \to \mathcal{X}$ with parameter θ, which in turn induces a distribution P_θ. Moreover, the discriminator defines a differentiable mapping $D : \mathcal{X} \to [0, 1]$, which describes the probability of an observed sample $\mathbf{x}$ coming from P^* rather than P_θ. The learning can be formulated as a minimax game with value function $V(G, D)$

$$\begin{aligned} V(G,D) &= \mathbb{E}_{p^*(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \\ &= \mathbb{E}_{p^*(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{p_\theta(\mathbf{x})}[\log(1 - D(\mathbf{x}))], \end{aligned} \tag{2.8}$$

which is jointly optimized in a two step procedure switching between the parameters in the generator and the discriminator. The first term in the value function reflects the discriminator loss, which aims to maximize the log-probability of assigning correct label to samples from P^*. Simultaneously we train the generator to minimize the log-probability of assigning the wrong label to samples from P_θ, i.e. classifying a generated sample as a data sample. Note that for a fixed generator, the discriminator minimizes the binary cross-entropy between the real and the generated data distribution.

Consider the case when the the generator G is fixed. We can then deduce an optimal discriminator. Then the optimal discriminator can be found to be

$$D^*(\mathbf{x}) = \frac{p^*(\mathbf{x})}{p^*(\mathbf{x}) + p_\theta(\mathbf{x})},$$

which is easily found by differentiating the function. Inserting this into the value function, we get

$$\begin{aligned} V(G,D^*) &= \mathbb{E}_{p^*(\mathbf{x})}\left[\log \frac{p^*(\mathbf{x})}{p^*(\mathbf{x}) + p_\theta(\mathbf{x})}\right] + \mathbb{E}_{p_\theta(\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x})}{p^*(\mathbf{x}) + p_\theta(\mathbf{x})}\right] \\ &= -\log(4) + D_{\mathrm{KL}}\left(\mathrm{P}^* \,\middle\|\, \frac{\mathrm{P}^* + \mathrm{P}_\theta}{2}\right) + D_{\mathrm{KL}}\left(\mathrm{P}_\theta \,\middle\|\, \frac{\mathrm{P}^* + \mathrm{P}_\theta}{2}\right) \\ &= -\log(4) + 2D_{\mathrm{JSD}}\left(\mathrm{P}^*, \mathrm{P}_\theta\right), \end{aligned}$$

where we in the last line recognize the Jensen-Shannon divergence, which is a smoothed and symmetric version of the Kullback-Leibler divergence. Since the Jensen-Shannon divergence is non-negative, $-\log(4)$ is a global minimum with respect to the parameters in the discriminator, and this is only obtained when $\mathrm{P}^* = \mathrm{P}_\theta$ and $D^*(\mathbf{x}) = \frac{1}{2}$. It is important to note that these theoretical results are based on updates in the function space of D and G. In practice, we parameterize the functions by deep neural nets and the updates are made in the parameter space. We are therefore limited to the properties of the functions, and as deep neural nets are nonconvex, the theoretical guarantees do not hold.

2.5 Representation learning

Much of the advances in deep learning have followed two main lines of research, optimization and representation learning. Optimization is concerned with tuning the neural network parameters to enhance performance and generalize to new data. This must also be done efficiently to scale to large and high-dimensional data. Moreover, as neural networks typically are over-parameterized, an optimizer has to make sure that each parameter provides enough information to make progress, while at the same time, it does not overfit. Examples along this line of research are, but not limited to, various stochastic optimization schemes, regularization techniques, and theoretical studies of the

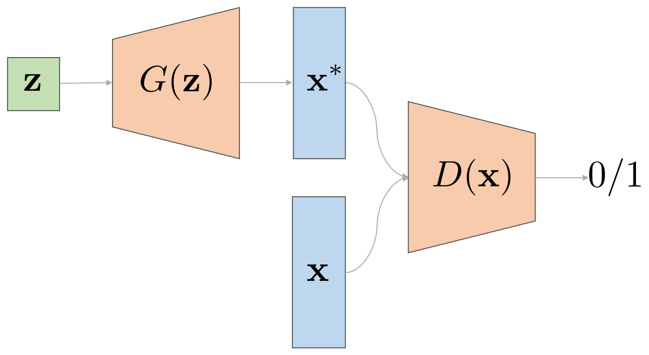


Figure 2.4: Graphical representation of a GAN. A latent variable $\mathbf{z}$ sampled from a prior distribution transformed by a deterministic mapping $G : \mathcal{Z} \rightarrow \mathcal{X}$. The discriminator D inputs either a generated sample $\mathbf{x}^*$ or a realization from the dataset $\mathbf{x}$ and outputs the probability of it being a real observation.

bias-variance trade-off. Common for all of these is that they treat the deep neural network itself as a black-box family of functions and rely on their universal approximation properties. This approach is in contrast to the other line of research, representation learning, that will be our primary concern throughout this thesis.

Representation learning focuses on the fundamental question of how a neural network represents the data, making it easy to extract useful information. At the same time, it remains agnostic to the optimization procedure. Traditionally, prior knowledge of a human affected the choice of representation, or the features, of a dataset. This process, typically termed feature engineering, plays a significant part in the model's performance on a given task. While feature engineering utilizes the human understanding of the world, the process is tiresome and typically have to be repeated if the domain of the data changes. The aim of representation learning at large is, therefore, making models that *understands the world* by capturing underlying explanatory factors and reduce the need for human input.

We'll illustrate representation learning by mentioning some of the advances within the field that use neural networks in their models. In Natural Language Processing (NLP) models utilize neural networks to represent each word in what is called an *word embedding*, which is a representation in a low dimensional vector space [Devlin et al., 2018, Mikolov et al., 2013]. These embeddings carry semantic meaning and can be used as features in inference tasks such as text generation, question answering, and summarizing. Furthermore, we can fine-tune a large language model on a smaller domain-dependent dataset using fewer resources. This process is called *transfer learning*, which itself is a general objective of representation learning. By learning a simpler representation of complex data, this representation can be used in other inference tasks and adapted to new domains. In the image domain, CNNs have been the dominant force in recent innovations. While models on image data previously used specific algorithms to detect features such as edges and textures, one has observed that deep CNN architectures learn such features layer-wise by training the model end-to-end.

We need to make a distinction between supervised and unsupervised representation learning. In the supervised case, the model changes its representation in the parameters based on some ground-truth or label. An example of this is when we estimate a CNN in an object classification task. Based on a prediction loss computed from the true label, the model changes its weights, and implicitly the model learns to represent edges and textures in the hidden layers. Quite naturally, unsupervised learning is when the model only has access to the data and not some ground truth. The word embeddings described in the previous paragraph is an example of when a model can learn a useful representation of the data while only having access to the text itself. As most of the world is unlabeled, unsupervised representation learning poses a more challenging problem with more significant potential. The rest of this thesis will be in the unsupervised setting.

Unsupervised representation learning is no recent endeavor. Principal Component Analysis (PCA) [Jolliffe, 1986] composes the data in a linear transformation where the basis consists of orthogonal vectors that capture the highest variance. More recently, several methods have been proposed to capture non-linear representations. Independent Component Analysis (ICA) [Hyvärinen and Oja, 2000] generalizes to non-linear models where the aim is to identify a non-Gaussian basis that is statistically independent of each other, and not only uncorrelated as in PCA. Neural networks have been widely employed in unsupervised representation learning, notably in a class of models termed Autoencoders [Hinton and Zemel, 1994]. An Autoencoder consists of two networks; the *encoder* maps from the data $\mathbf{x}$ to some feature space $\mathbf{z}$ of typically lower dimension than the data, and the *decoder* maps back to a point $\mathbf{x}'$ of the same dimension as the data. Both the encoder and decoder are trained simultaneously on how well the model can reconstruct the data, $\mathcal{L}(\mathbf{x}, \mathbf{x}')$. The point in the feature space $\mathbf{z}$ will be a compressed representation of the observed data.

We need to define what constitutes a good representation of the data before we look at specific models and methods to evaluate them. Several papers have discussed this fundamental problem [Bengio et al., 2012, Ridgeway, 2016, Lake et al., 2016], and here we will summarize some of the essential qualities a desired representation should have. First of all, the representing should be *faithful* to the data, meaning that the representation should preserve the information. Yet, the representation should encode the data in a *compact* manner, meaning that the representation should be an efficient compression of the data. Still, the features in the compressed representation should be *interpretable* by humans. For instance, even though an image consists of a high-dimensional feature space defined by the pixels, there may be a small set of features that a human needs to describe visually. Such features can be the position, rotation, shape, and color of an object. Therefore, it is desirable to have a representation that captures a causal relationship between the features in the representation and what a human observes in the data, and are invariant to irrelevant transformations.

Disentanglement

Both Bengio et al. [2012], Ridgeway [2016] advocate for a factorial representation where each factor captures attributes that makes up the data. A common assumption is that the complex data we observe originates from interactions from a set of explanatory factors. The set of relevant factors and irrelevant factors might differ depending on the task at hand, but generally, we want a representation that generalizes to multiple tasks. The interaction between these factors might be arbitrarily complex, and a fundamental goal is to separate, or *disentangle*, the underlying factors. While there is no formal definition of disentanglement that is widely accepted, beyond the human intuition, recent methods usually align on the goal given by Bengio et al. [2012].

> *A representation should disentangle as many factors of variations as possible, discarding as little information about the data as is practical.*

Such a representation will make it easier to extract useful information when performing prediction or other inference tasks.

In a generative setting, we assume the data is generated in a two-step process. First the unknown latent variables $\mathbf{z}$ are sampled from a prior distribution $p(\mathbf{z})$. Then the observed data $\mathbf{x}$ is sampled from a conditional distribution $p(\mathbf{x}|\mathbf{z})$. Intuitively, the latent variables correspond to semantically meaningful factors that transform into the form of the data we observe. We then want to estimate an approximation to the true underlying latent variables, termed $\hat{\mathbf{z}} = f(\mathbf{x})$, for some transformation f. When a single underlying factor of variation changes in the true and unknown latent variable z_i, a disentangled representation should lead to a change in a single coordinate of the estimated representation $\hat{\mathbf{z}}$.

The Information Bottleneck

Representation learning can be stated using the principle of the information bottleneck, first introduced by Tishby et al. [2000]. Before stating the principle, we need to review some of the core concepts in information theory [Song et al., 2008]. Assume that we have two continuous random variables X and Y with probability density functions $p(x)$ and $p(y)$, with values x and y. Note that we in this section denotes the densities without subscript, and differentiate them through the argument. The *differential entropy* of X is defined as

$$H(X) = -\int p(x)\log(p(x))dx = -\mathbb{E}_p[\log(p(X))].$$

When X is a discrete random variable and p is the probability mass function, the integral is a sum over the domain, and the quantity is simply known as the *entropy*. For discrete data $x_1, \ldots, x_N$ with probability mass function p, the entropy is

$$H(X) = -\sum_i p(x_i)\log(p_i(x)).$$

In this case, entropy is interpreted as the average level of uncertainty in the outcomes of a random variable. It describes the amount of average information required to describe the random variable [Shannon, 2001]. Note that the differential entropy does not have the same interpretation in general, as we can construct a continuous density such that the entropy is negative. To see this consider the uniform distribution on a interval $[0, a]$ with density $p(x) = 1/a$ for a scalar a. Now we compute the differential entropy to be $H(X) = \log(a)$, which is negative for $a \in (0, 1)$.

Letting $p(x, y)$ be the joint probability density of X, Y, the conditional differential entropy is defined by

$$H(X|Y) = -\int p(x,y)\log(p(x|y))\,dx\,dy. \tag{2.9}$$

Using the chain rule, $p(x, y) = p(x|y)p(y)$, and inserting into Equation (2.9), the joint differential entropy becomes $H(X, Y) = H(X|Y) + H(Y)$.

The *mutual information* is defined as

$$I(X,Y) = \int p(x,y)\log\left(\frac{p(x,y)}{p(x)p(y)}\right)\,dx\,dy.$$

Recalling to Section 2.2,we see that the mutual information is the KL divergence, or *relative entropy*, between the joint and the product of the marginals. As opposed to the differential entropy, the mutual information retains its properties in the continuous case. We then see that the mutual information is zero if and only if $p(x, y) = p(x)p(y)$, i.e., the random variables are independent. As the KL-divergence is non-negative, the mutual information is a measure of dependence between X and Y. In information theoretic terms, the KL-divergence $D_{KL}(p||q)$ is the inefficiency when assuming the distribution is q when the true distribution is p. We can see this connection when rewriting the mutual information using the chain rule once again and marginalizing over y,

$$\begin{aligned} I(X,Y) &= -\int p(x,y)\log\left(p(x)\right)\,dx\,dy + \int p(x,y)\log\left(p(x|y)\right)\,dx\,dy \\ &= H(X) - H(X|Y). \end{aligned}$$

Observe that the mutual information is the reduction in entropy when Y is known. The mutual information is a symmetric quantity as seen from the definition, and can this be equivalently be written as $I(X, Y) = H(Y) - H(Y|X)$. Furthermore, as the KL-divergence is non-negative we have that $H(X) \geq H(X|Y)$. In the discrete case the entropy $H(X)$ is non-negative as the probability mass function takes values on the interval $[0, 1]$, but this is not necessarily the case in a continuous setting.

Now we can state the information bottleneck principle. Suppose we extract a representation $\hat{Z}$ from the data X, which we then use for a task Y. For instance, the task could be image classification, and $\hat{Z}$ is some compressed representation of the high-dimensional image. Ideally, $\hat{Z}$ should capture only the relevant information for the task. The flow of information follows $X \to \hat{Z}$ and $\hat{Z} \to Y$. The amount of information contained in that data for the task is given by the mutual information $I(X, Y)$. Note that the data processing inequality [Cover and Thomas, 2006, Theorem 2.8.1], also previously discussed for f-divergences, states that $\hat{Z}$ can't contain more information about Y than X, written out as $I(X, Y) \geq I(\hat{Z}, Y)$. The representation should ideally contain all of this information such that the data processing inequality is equality, $I(\hat{Z}, Y) = I(X, Y)$, in which case we say that the representation $\hat{Z}$ is *sufficient statistic* for Y. However, the identity mapping would obtain a sufficient representation, which is not useful. Therefore we would like to maximize the mutual information under some constraint on the complexity to ensure that the representation is indeed a compression. Let such a constraint be given by $I(X, \hat{Z}) \leq I_c$ for some constant information constraint I_c. Now we can state the problem equivalently in the Lagrangian form, where we want to maximize the function

$$\mathcal{L}(p(z|x)) = I(\hat{Z}, Y) - \beta I(\hat{Z}, X). \tag{2.10}$$

This formulation is known as the Information Bottleneck. By varying β, the function $\mathcal{L}$ trades the encoding $\hat{Z}$ ability to be expressive about Y and impressive about X.

The question of finding all the relevant information of a random variable lies close to the heart of classical statistics, which implicitly aligns with the essence of representation learning. This concept of relevance is formulated through a *sufficient statistics* for a parametric family of distributions. Concretely, suppose X has distribution from a family $\mathcal{P} = \{P_\theta : \theta \in \Omega\}$ for parameters θ. Then $T = T(X)$ is a sufficient statistic for X if for ever θ and t, $P_\theta(X[T = t)$ does not depend on θ. Moreover, T is *minimal* if for every other sufficient statistic $\hat{T}$ there is a deterministic function f such that $T = f(\hat{T})$ [Keener, 2011, Chapter 3]. The notion of a minimal sufficient statistic can also be formalized as the statistic having the smallest mutual information with X and the largest mutual information with θ [Kullback and Leibler, 1951]. The information bottleneck framework provides a trade-off between sufficiency and minimality by the parameter β.

Achille and Soatto [2017] further points out two other properties of an optimal representation besides minimality and sufficiency. First, the representation should be independent of random variables that are not informative to the task. Such as a random variable, N, is termed a nuisance, and $\hat{Z}$ should be *invariant* to N such that $I(\hat{Z}, N) = 0$. For instance, if the task is to classify images of dogs and cats, the lighting should not be of concern and is, therefore, a nuisance variable. Secondly, an optimal representation should be *maximally disentangled*, which points back to our previous discussion. Each factor in $\hat{Z}$ should capture the underlying factor of variation. Achille and Soatto [2017] states this in terms of statistical independence of the components, and proposes to measure the disentanglement by the *total correlation*,

$$\mathrm{TC}(\hat{Z}) = D_{\mathrm{KL}}\left(p(z) \| \prod_i p(z_i)\right).$$

The mutual information is generally hard to compute, so the information bottleneck was originally stated using discrete data [Tishby et al., 2000], and was later extended to Gaussian variables [Chechik et al., 2005]. More recently, the information bottleneck poses as a framework for the theoretical study of deep learning [Tishby and Zaslavsky, 2015]. Furthermore, a variational approximation to the information bottleneck was derived such that arbitrary distributions, e.g., parameterized by neural networks, can be estimated [Alemi et al., 2016]. Later in Chapter 4, we will see how the information bottleneck principle is used in learning disentangles representations in deep generative models.

Chapter 3

Variational Autoencoders

Variational Autoencoders (VAE) [Kingma and Welling, 2013, Rezende et al., 2014] have become a well-established framework in deep generative models that combines approximate Bayesian inference and deep neural networks. In this chapter, we give an independent and self-contained introduction to VAE from Bayesian principles. We start by introducing the latent variable model and discuss how we can use classical variational inference to estimate an otherwise intractable posterior density. VAE scales classical variational inference by utilizing the flexibility of neural networks, while still remaining tractable. Lastly, we consider VAE in the greater context of deep generative models and point out significant extensions and related work.

3.1 Latent variable models

A common assumption in Bayesian inference is that the generative process depends on some *latent variable*. Latent variables $\mathbf{z}$ are part of the model, but they are not observed. The generative process is specified by the samples we observe $\mathbf{x}$, as well as the latent variables, denoted by $\mathbf{z}$, and the joint distribution $p(\mathbf{x}, \mathbf{z})$. The data is assumed to be generated by first sampling in the latent space $\mathcal{Z}$ then mapped to the space $\mathcal{X}$ where we observe it as data. We let p^* denote the true, but unknown, data generating distribution, and let p_z be a specified prior distribution on the latent variable. We model the data by specifying a mapping from the latent space to the data with the transformation denoted as $G_{\boldsymbol{\theta}} : \mathcal{Z} \to \mathcal{X}$, where $\boldsymbol{\theta}$ are the parameters. The transformation can possibly be random, but we will assume it is deterministic. The transformation specifies a conditional distribution $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ of the data conditioned on a latent variable. The central object in generative models is to approximate the true underlying distribution of the data $p^*(\mathbf{x})$ with the distribution $p_{\boldsymbol{\theta}}(\mathbf{x})$. This is typically done using *maximum likelihood*. Assume that we have a dataset $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N$ that are $i.i.d.$ and fully-observed. We then want to maximise the the probability of observing the data with respect to the parameters $\boldsymbol{\theta}$. For convenience, we can equivalently maximize the log-likelihood which takes the form,

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}_1, \dots, \mathbf{x}_N) = \sum_{i=1}^N \log p_{\boldsymbol{\theta}}(\mathbf{x}_i).$$

Note that this is a function of the $\boldsymbol{\theta}$ and not the data. Using the latent model setup, the density is given by

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \int p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) p_z(\mathbf{z}) d\mathbf{z}.$$

If the prior distribution over $\mathbf{z}$ is discrete, we see from this expression that the distribution of $\mathbf{x}$ is a mixture of the conditional distributions. In the continuous case, we can interpret the marginal as an infinite mixture over the infinitesimals $d\mathbf{z}$. Using this setup, we can specify a flexible marginal density implicitly using the prior and conditional, which, in turn, allows us to model complex dependencies between variables without looking at the dependencies between them directly. As we will see later, in a Variational Autoencoder both $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ and $p_z(\mathbf{z})$ are Gaussian distributions. Any smooth density can be approximated arbitrarily precise by a Gaussian mixture model with enough components [Sorenson and Alspach, 1971].

We would often like to compute the posterior distribution of the latent variables. Using Bayes' theorem, we have

$$p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})} = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_z(\mathbf{z})}{\int p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})d\mathbf{z}}.$$

The joint density is easily accessible using the specified prior distribution and the conditional density using the transformation from the latent space to data. However, we need the solve the integral in the denominator to evaluate the marginal density for $\mathbf{x}$. Similarly, we need $p_{\boldsymbol{\theta}}(\mathbf{x})$ to evaluate the log-likelihood such that we can estimate $\boldsymbol{\theta}$. However, if the mapping $G_{\boldsymbol{\theta}}$ is complex the integral may not have a analytical solution. This is the case when $G_{\boldsymbol{\theta}}$ is a deep neural network. Furthermore, we typically want to scale to applications with large datasets. To compute the exact posterior requires to integrate over all the latent variables, which typically are intractable when the data are large and of high-dimension. This leads us to approximate inference techniques.

The different approaches in approximate inference fit into two main categories, either based on sampling or optimization. Methods based on Monte Carlo samples provides a nonparametric representation of a probability distribution through the set of samples. Arguably the most successful approach is Markov Chain Monte Carlo (MCMC), which produces samples through a constructed random walk [Robert and Casella, 2005]. MCMC has the advantage of often being unbiased, and it processes exact samples in the asymptotic limit. However, MCMC is typically computationally expensive, which poses as a practical constraint for considering MCMC when dealing with large datasets. Optimization-based methods, such as variational inference, are often faster and scales better to large datasets, but generally not asymptotic exact. Ultimately, the preferred approach is highly dependent on the available data and the intended application. Variational Autoencoders are stated using the framework of variational inference.

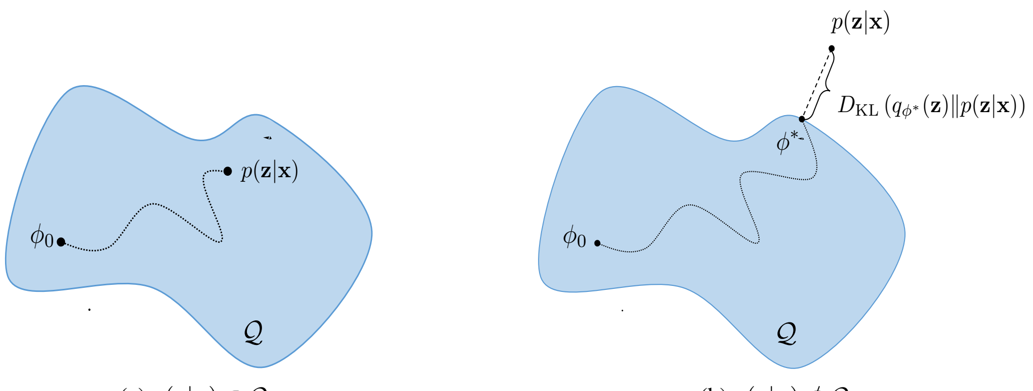
3.2 Variational inference

Variational inference [Blei et al., 2016, Wainwright and Jordan, 2008, Zhang et al., 2017] seek to find an approximate distribution to the posterior $p(\mathbf{z}|\mathbf{x})$ by optimizing an objective function. Recall from Section 2.2 that there exists a range of divergences which measures the similarity of two distributions. While variational inference can be extended to a range of divergences, the most commonly used is the reverse KL divergence. Thus we seek an *variational distribution* that minimize the reverse KL divergence with respect to the unknown posterior distribution, which in turn can be used as a proxy for the posterior distribution. However, as optimizing over all possible distribution is infeasible, the we restrict the problem to a subset, or a *variational family* of distribution, termed $\mathcal{Q}$ that is parameterized by *variational parameters* $\boldsymbol{\lambda}$. For instance, if the variational family is the set of univariate Gaussian distributions the variational parameters is $\boldsymbol{\lambda} = \{\boldsymbol{\mu}, \boldsymbol{\sigma}^2\}$. For notational simplicity, we will omit the variational parameters using the convention $q(\mathbf{z}) = q(\mathbf{z}; \boldsymbol{\lambda})$. The problem at hand is therefore,

$$q_{\boldsymbol{\phi}^*} = \arg\min_{q \in \mathcal{Q}} D_{\mathrm{KL}}(q(\mathbf{z}) \| p(\mathbf{z}|\mathbf{x})). \tag{3.1}$$

Note that we can obtain an exact representation of the posterior only if $p(\mathbf{z}|\mathbf{x}) \in \mathcal{Q}$, otherwise, the approximation will be inherently biased. We can reduce the bias by selecting $\mathcal{Q}$ to be flexible enough to provide a sufficient approximation measured by the KL-divergence, as illustrated in Figure 3.1.

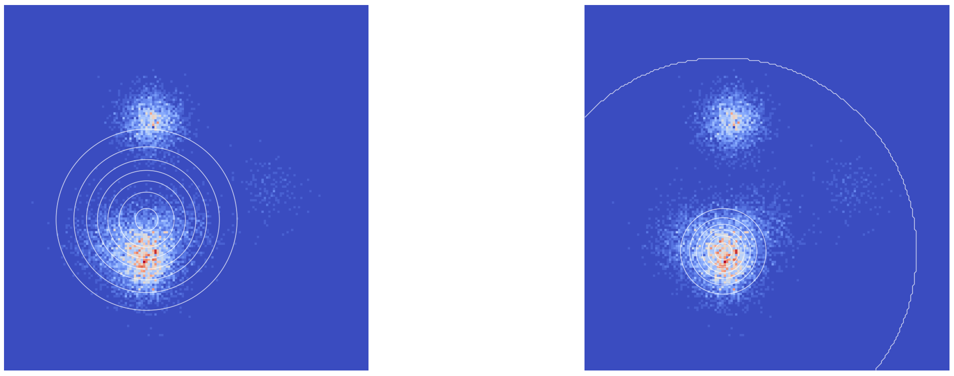
As the KL divergence is not symmetric, there is a considerable difference when considering $D_{\mathrm{KL}}(p\|q)$ and $D_{\mathrm{KL}}(q\|p)$ as the objective. While variational inference utilizes the former, the latter is the basis for another approximate inference method called *expectation propagation* [Minka, 2013]. The difference is notably apparent when we use an underspecified model, and we will provide a short example to illustrate the different properties. Consider the case as shown in Figure 3.2 where the data is generated from a mixture of Gaussians $p(\mathbf{x})$ and the approximate model is a unimodal Gaussian with isotropic covariance matrix $q(\mathbf{x})$. The model is underspecified since we approximate a multimodal distribution with simpler distribution with only one mode. The underlying data is generated as a mixture of isotropic Gaussians $p(\mathbf{z}) = \sum_i w_i N(\boldsymbol{\mu}_i, 1/\tau_i I)$ where $\boldsymbol{\mu}_i$ and τ_i is sampled from a normal and uniform distribution respectively, while the mixture weights w_i are sampled from the Dirichlet distribution. Then an isotropic Gaussian distribution $q(\mathbf{z}) = N(\boldsymbol{\mu}, \sigma^2 \mathrm{I})$ is used to approximate the data according to the forward KL in 3.2a and the backward KL in 3.2b.


$p(\mathbf{z}|\mathbf{x})$
ϕ_0
$\mathcal{Q}$
$p(\mathbf{z}|\mathbf{x})$
$D_{\mathrm{KL}}\left(q_{\phi^*}(\mathbf{z})\|p(\mathbf{z}|\mathbf{x})\right)$
ϕ^*
ϕ_0
$\mathcal{Q}$

(a) $p(\mathbf{z}|\mathbf{x}) \in \mathcal{Q}$ (b) $p(\mathbf{z}|\mathbf{x}) \notin \mathcal{Q}$

Figure 3.1: If the exact posterior belongs to the variational family, the optimization yields an optimal solution given a convex parameterization (Figure 3.1a). Otherwise, the variational distribution will have a bias measured by the KL divergence up to a constant (Figure 3.1b).

Consider first the case in Figure 3.2a with objective $D_{\mathrm{KL}}(p\|q)$ where the expectation is taken with respect to $p(\mathbf{x})$. In areas with low mass, i.e. $p(\mathbf{x}) \approx 0$, the KL will approach zero as well and $q(\mathbf{x})$ can be arbitrarily large without paying additional cost in the optimization problem. Conversely, the KL can become arbitrarily large if $p(\mathbf{x}) > 0$ and $q(\mathbf{x}) \approx 0$. This results in that $q(\mathbf{x})$ will try to cover most of the mass, even though if that means that it cover areas where $p(\mathbf{x})$ has no mass. We clearly see this in Figure 3.2a, where the approximation seems to average over the modes. The opposite abstractions is valid when we consider $D_{\mathrm{KL}}(q\|p)$ as in Figure 3.2b. Here $q(\mathbf{x})$ will place most of the mass around the largest mode of $p(\mathbf{x})$, at the cost of ignoring smaller modes. In short, we say that $D_{\mathrm{KL}}(q\|p)$ is *mode-seeking*, while $D_{\mathrm{KL}}(p\|q)$ is *mode-averaging*. Note that both are consistent strategies, but they differ in underspecified settings.



(a) $D_{\mathrm{KL}}(p(\mathbf{x})\|q(\mathbf{x}))$ (b) $D_{\mathrm{KL}}(q(\mathbf{x})\|p(\mathbf{x}))$

Figure 3.2: Difference between $D_{\mathrm{KL}}(p \parallel q)$ and $D_{\mathrm{KL}}(q \parallel p)$ when using a underspecified model. The heat-map shows the underlying data, while the contours represent the fitted model. We used code from Theis et al. [2015] to produce these results.

Evidence Lower Bound

Observe that the KL divergence in Equation (3.1) is itself dependent on the unknown posterior, and thus, it can't be optimized directly. We circumvent this by rewriting the KL divergence in a quantity that is equivalent up to a constant, known as the *Evidence Lower Bound* (ELBO). As the same suggests, we aim to lower bound the evidence, which is the term commonly used for the marginal likelihood of the data. Using Jensen's inequality we get,

$$\begin{aligned}\log p(\mathbf{x}) &= \log \int p(\mathbf{x},\mathbf{z})\,d\mathbf{z} = \log \int \frac{p(\mathbf{x},\mathbf{z})}{q(\mathbf{z})} q(\mathbf{z})\,d\mathbf{z} \\ &= \log \left(\mathbb{E}_q \left[\frac{p(\mathbf{x},\mathbf{z})}{q(\mathbf{z})} \right] \right) \geq \mathbb{E}_q \left[\log \left(\frac{p(\mathbf{x},\mathbf{z})}{q(\mathbf{z})} \right) \right] := \mathcal{L}(q).\end{aligned}$$

We want to maximize $\mathcal{L}$ as it is a lower bound. The ELBO can be interpreted in several ways, and by splitting the logarithm we end up at our first,

$$\mathcal{L}(q) = \mathbb{E}_q \left[\log\left(p(\mathbf{x},\mathbf{z})\right] + H(q),$$

where we recognize the differential entropy. Maximizing this expression, the first term encourages $q(\mathbf{z})$ to place where mass where $p(\mathbf{x},\mathbf{z})$ have high probability, while the entropy ensures that $q(\mathbf{z})$ doesn't collapse to one point.

As the variational objective in Equation (3.1) is to minimize the KL divergence, we need to establish that maximization of the ELBO is an equivalent objective. By the product rule,

$$\begin{aligned}D_{\mathrm{KL}}(q(\mathbf{z})\|p(\mathbf{z}|\mathbf{x})) &= \mathbb{E}_q\left[q(\mathbf{z})\right] - \mathbb{E}_q\left[p(\mathbf{z}|\mathbf{x})\right] \\ &= \mathbb{E}_q\left[q(\mathbf{z})\right] - \mathbb{E}_q\left[p(\mathbf{x},\mathbf{z})\right] + \log p(\mathbf{x}) \\ &= -\mathcal{L}(q) + \log p(\mathbf{x}).\end{aligned}$$

Observe that the log-evidence, $\log p(\mathbf{x})$, does not depend on $q(\mathbf{z})$ and can thus be regarded as a constant. Varying $q(\mathbf{z})$ such that the ELBO increases will therefore imply to the KL-divergence decreases by the same amount, and consequently, minimization of the KL-divergence is equivalent to maximizing the ELBO.

Since the KL-divergence is non-negative, the ELBO is a lower bound on the evidence, and the KL-divergence describes the tightness of the bound. However, if the posterior is contained in the variational family, as illustrated in Figure 3.1, the KL-divergence approaches zero for a convex problem, and we can accurately estimate the evidence using the ELBO. If we didn't restrict on the variational distribution $q \in \mathcal{Q}$, the optimal choice would be $q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x})$.

Finally, we provide a interpretation of the ELBO when dealing with a latent variable model where the joint density is on the form $p(\mathbf{x},\mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$. Then we can rewrite the ELBO as

$$\begin{aligned}\mathcal{L}(q) &= \mathbb{E}_q\left[\log\left(\frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q(\mathbf{z})}\right)\right] \\ &= \mathbb{E}_q\left[\log\left(p(\mathbf{x}|\mathbf{z})\right)\right] - D_{\mathrm{KL}}(q(\mathbf{z})\|p(\mathbf{z})).\end{aligned}$$

Here the first term is the expected log-likelihood of the data with respect to the variational distribution, while the second term acts as a regularizer ensuring that the variational distribution does not deviate too much from the prior. The classic approach in variational inference, is to analytically evaluate the expectation before optimizing. Both terms is an expectation is with respect to $q(\mathbf{z})$. It is common, as we will see in the next section, to choose a form of variational family that makes it simple to evaluate the ELBO.

Classical variational inference

A popular choice of the variational family is the so-called *mean field approximation*. The assumption is that the $q(\mathbf{z})$ factorizes into a product of lower-dimensional terms on the form,

$$q(\mathbf{z};\boldsymbol{\lambda}) = \prod_{i=1}^{N} q_i(z_i;\lambda_i).$$

Computing an expectation with respect to a distribution of this form amounts to solving a collection of one-dimensional integrals, which is easier than solving a potentially high-dimensional integral. Furthermore, the KL-divergence factorizes since the components of $q(\mathbf{z})$ is assumed to be independent. The factorized form of $q(\mathbf{z})$ is suited for a coordinate ascend optimization scheme where each coordinated is updated while we fix the others. It is possible to derive closed-form updates for each coordinate. The derivations often simplify when dealing with distributions in the exponential family as models with conjugate priors gives the form of the variational distribution. We refer to [Bishop, 2006, Chapter 10] for an example on how to explicitly carry out such derivations. Note that the ELBO is intractable for many models, and even if it has an analytical solution, the derivations can be cumbersome. It is possible to generalize the approximation above so that we factorize the distributions in disjoint groups instead of each coordinate.

However, this practicality of the mean-field approximation comes a cost. First of all, we restrict the model not to include dependencies between the latent variable. The variational distribution is thereby less flexible and might give a poor approximation in some cases. Furthermore, there is an increasing need for making variational inference applicable to complex models and big datasets. In the approach described above, it is necessary to optimize the variational parameters for each data point, as illustrated in Figure 3.3. Stochastic optimization has been deployed to combat this issue in models termed *Stochastic Variational Inference* (SVI) [Hoffman et al., 2013]. We can then more easily scale to larger datasets, but SVI still works with local variational parameters. There have also been several recent efforts into making variational inference more generic and go beyond conjugate distribution in the exponential family. As we will see in the next section, Variational Autoencoders deals with both the scalability and generality, which often poses as bottlenecks for classical variational inference methods.

3.3 Variational Autoencoders

Variational Autoencoders (VAE) is a latent variable model that efficiently scales to large datasets and complex distributions using neural networks. It accomplishes this mainly by combining two concepts. First, *amortized inference* bypasses the need to optimize the variational parameters for each data point, making it more efficient. Second, it allows for computing the gradient of the ELBO with respect to the variational parameters using *reparametrization* gradient, eliminating the need for model-specific calculations as the practice is for conjugate exponential family.

Amortized inference

As seen in the graphical model Figure 3.3, classical variational inference is typically set up such that each data point. Each data point $\mathbf{x}_i$ corresponds to a latent variable $\mathbf{z}_i$, which in turn is specified by the variational parameters λ_i. When the dataset becomes large, this update loop becomes computationally expensive, making it impractical to scale in many cases. Amortized inference bypasses this bottleneck by introducing a function that can predict the optimal latent variable given a data point. This function is parameterized and estimated using the data, allowing for using past computations to support future computations. More specifically, we introduce a deterministic predictor, $\mathbf{z}_i = f_{\boldsymbol{\phi}}(\mathbf{x}_i)$ with parameters $\boldsymbol{\phi}$ and assume that it can give a sufficient approximation of the latent variables. The model shares the parameters $\boldsymbol{\phi}$ all data points, thereby coining the term amortized inference. We illustrate the difference between SVI and amortized inference in Figure 3.3.

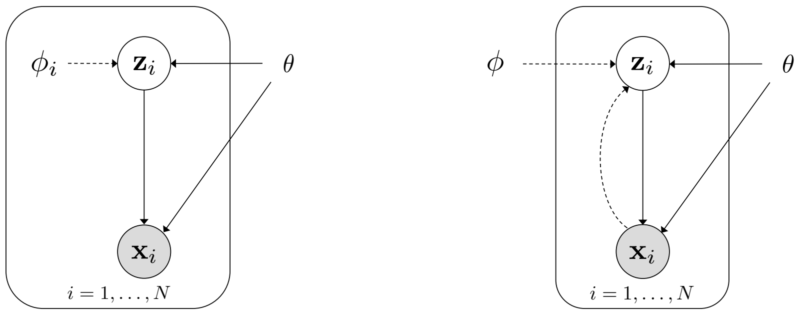
Even though amortized inference involves a deterministic mapping as a predictor, we still have to specify the variational distribution. This is done be letting the variational parameters be parameterized by the mapping, as we will illustrate in the Gaussian case. In this case, the variational parameters are $\lambda = \{\mu, \sigma^2\}$, where we assume that the Gaussian is isotropic. The amortized assumption is then that both the mean and variance can be predicted from the data, which in turn induces the variational distribution. We deploy the notation $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ to underline that the latent variables are a function of the data specified by $\boldsymbol{\phi}$. We can then specify the variational distribution as

$$q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i) = \mathrm{N}(\boldsymbol{\mu}(\mathbf{x}_i), \mathrm{diag}(\boldsymbol{\sigma}^2(\mathbf{x}_i))).$$

Furthermore, it is common to impose a mean-field approximation on the variational distribution, disregarding correlations between data points,

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

Note that left hand side in this equation denotes the set of observations, while the right hand side denotes the product of the vector elements in the set.



(a) SVI (b) Amortized inference (VAE)

Figure 3.3: Graphical models that illustrate the difference between Stochastic Variational Inference (SVI) and amortized inference in VAE. For a dataset indexed bu $i = 1, \ldots, N$, we observe the data $\mathbf{x}_i$ that we assumes depends on a latent variable $\mathbf{z}$. We model the dependency with a parameter $\boldsymbol{\theta}$ that specifies a mapping from the latent variable to the data. Classical VI techniques and SVI assumes that each $\mathbf{z}_i$ depends on some variational parameter λ_i which is optimized per datapoint (Figure 3.3a). On the other hand, in amortized inference, $\boldsymbol{\phi}$ specifies the parameter in an inverse mapping from a $\mathbf{x}_i$ to $\mathbf{z}_i$. (Figure 3.4b).

By using amortized inference, we rely on that the function mapping is flexible enough to generalize across the dataset. In VAE, this mapping is a deep neural network, and the parameters $\boldsymbol{\phi}$ denotes the corresponding weights and biases. When the network is estimated, a new data point passes through to predict the latent variable. In the sense, this neural network are often called *inference networks* since this facilitates approximate inference of the unknown posterior $p(\mathbf{z}|\mathbf{x})$. In addition do the inference network, VAE also employs an additional neural network to model the generative process, that is the conditional $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$. Here $\boldsymbol{\theta}$ denotes the parameters in the neural network. The generative process works by first constructing a sample from the prior on the latent variables $p(\mathbf{z})$ before feeding the sample through the network that induces $p(\mathbf{x}|\mathbf{z})$. In training a VAE the flow is governed by first sampling from the approximate posterior $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ before feeding it through $p(\mathbf{x}|\mathbf{z})$. The above sampling procedure is the reason the neural network is referred to as the *generative network*. Due to the latter procedure, we also refer to is as the *recognition network*

We now have that the ELBO depends on both $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$, written out as

$$\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\theta}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log\left(p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right)\right] - D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})). \tag{3.2}$$

Instead of relying on coordinate ascent and simplifying mean field approximation, the parameters are learning using an approach called *black-box variational inference* first coined by Ranganath et al. [2013]. The idea is to use gradient based optimization that are applicable to a large class of variational distributions. Hence, the only assumption is that $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is differentiable with respect to $\boldsymbol{\phi}$. Furthermore, instead of only optimizing the inference network, VAE simultaneously estimates both $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$. This leaves us with the task of computing the gradient of the ELBO with respect to both $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$. For the latter, the gradient can easily be computed as

$$\nabla_{\boldsymbol{\theta}}\mathcal{L} = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\nabla_{\boldsymbol{\theta}} \log\left(p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right)\right],$$

since the KL-divergence term doesn't depend on $\boldsymbol{\theta}$. Note that the gradient can be moved inside since the expectation is with respect to $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ that is independent of $\boldsymbol{\theta}$. The gradient can then be estimated using Monte Carlo. However, taking gradient with respect to $\boldsymbol{\phi}$ is not that easy. Both terms in the ELBO is an expectation with respect to $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, and therefore the order of the expectation and the gradient can not be interchanged straightforwardly. This leads us to the second contribution of VAE that makes it feasible to efficiently compute unbiased gradient estimates using the reparametrization trick.

Gradient estimation

Estimating the gradient of an expectation of a function with respect to the parameters defining the expectations itself is a problem that shows up in several machine learning applications, and we refer to Mohamed et al. [2019] for a detailed survey. Before introducing the reparametrization gradient, we'll review another approach used in black box variational inference where the *score function* is utilized. The score function is the gradient of log-likelihood with respect to the distributional parameters, and using the chain rule it can be written as

$$\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \frac{\nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}. \tag{3.3}$$

The score functions is known from the context of maximum likelihood estimation, where we have the properties that the expectation is zero and the variance is the quantity known as the Fisher information. Using this, we derive the score function estimator of the ELBO as

$$\begin{aligned}
\nabla_{\boldsymbol{\phi}} \mathcal{L} &= \nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})] \\
&\overset{(i)}{=} \int (\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - (\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) + 1)) \nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \, d\mathbf{z} \\
&\overset{(ii)}{=} \int (\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})) \nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \, d\mathbf{z} \\
&\overset{(iii)}{=} \int (\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \, d\mathbf{z} \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[(\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})],
\end{aligned}$$

where the last term can be estimated using Monte Carlo. In (i) we differentiate under the integral, then we use in (ii) that $\int \nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \nabla_{\boldsymbol{\phi}} \int q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \nabla_{\boldsymbol{\phi}} 1 = 0$, before inserting the score function (3.3) in (iii). This is an unbiased and consistent estimator, and it is derived using only the assumption of $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is differentiable. This makes it applicable to a large range of distributions, even discrete latent variables. However, the estimator have been found to have large variance, making it hard to apply in many models. Ranganath et al. [2013] apply control variates and Rao-Blackwellization as means of reducing the variance.

The second approach to estimating the gradient, which is used in VAE, is to write the $\mathbf{z}$ as a deterministic transformation of a simpler and auxiliary random quantity $\boldsymbol{\epsilon}$. More specifically, we write $\mathbf{z} \overset{d}{=} g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon})$, where $\boldsymbol{\epsilon}$ follows some base distribution $p(\boldsymbol{\epsilon})$. This is known as as the *reparametrization trick* in Kingma and Welling [2013], but have previously been used in other applications as the pathwise derivative Glasserman [2013] or process derivative Pflug [1989]. Then using the Law of the Unconscious Statistician we have that the expectation of any function f can be written as

$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[f(\mathbf{z})] = \mathbb{E}_{p(\boldsymbol{\epsilon})}[f(g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}))]. \tag{3.4}$$

Equipped with this parameterization in Equation (3.4), we can move the gradient inside the expectation since the base distribution does not depend on $\boldsymbol{\phi}$. Specifically, the gradient of the ELBO becomes

$$\begin{aligned}
\nabla_{\boldsymbol{\phi}} \mathcal{L} &= \nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})] \\
&= \mathbb{E}_{p(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\phi}} \log p_{\boldsymbol{\theta}}(\mathbf{x}, g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon})) - \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon})|\mathbf{x})],
\end{aligned}$$

which is easily estimated using Monte Carlo drawing samples from $p(\boldsymbol{\epsilon})$. The reparamatrization gradient have empirically been shown to have lower variance than the score function estimator. However, reparamatrization gradient does not trivially extend to a large range of distributions since we rely on having a transformation from a simple base distribution, and in particular it does not extend to discrete distributions. A common example for when the reparametrization trick is applicable, is the multivariate Gaussian. Then if $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \mathrm{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $p(\boldsymbol{\epsilon}) = \mathrm{N}(\mathbf{0}, \mathbf{I})$, we can write $\mathbf{z} = \boldsymbol{\mu} + \mathbf{L}\boldsymbol{\epsilon}$ with $\mathbf{L}\mathbf{L}^T = \boldsymbol{\Sigma}$. Such simple transformations exists for common distributions such as Dirichlet, Gamma and Exponential.

VAE typically inference network that parameterize a isotropic Gaussian distribution on the form

$$q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i) = \mathrm{N}(\boldsymbol{\mu}(\mathbf{x}_i), \mathrm{diag}(\boldsymbol{\sigma}^2(\mathbf{x}_i))),$$

and where prior on the latent variables is set to $p(\mathbf{z}) = \mathrm{N}(\mathbf{0}, I)$. Note the use of amortized inference where the mean and variance is predicted using a neural network with parameters $\boldsymbol{\phi}$. We'll close this section off by deriving the reparameterized estimate of the ELBO that is used in optimization, using the form in Equation (3.2). For simplicity, we'll write out the ELBO for a single data point $\mathbf{x}_i$. For i.i.d. data, the ELBO for the entire dataset is the sum the ELBO for each individual data point. First we note that the KL-divergence term can be solved analytically for two Gaussian distributions. If we let J be the dimension of $\mathbf{z}$, the KL is on the form

$$D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i)\|p(\mathbf{z})) = \frac{1}{2}\sum_{j=1}^{J}\left(1+\log\sigma_j^2-\mu_j^2-\sigma_j^2\right). \tag{3.5}$$

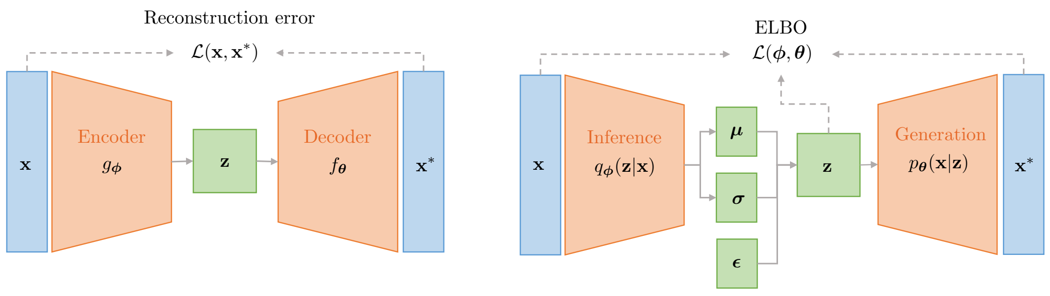
To estimate the second term, we'll draw L samples from the base distribution $\boldsymbol{\epsilon}_{(i,l)} \sim p(\boldsymbol{\epsilon})$, which are then transformed by $\mathbf{z}_{(i,l)} = \boldsymbol{\mu}(\mathbf{x}_i) + \boldsymbol{\sigma}(\mathbf{x}_i) \odot \boldsymbol{\epsilon}_{(i,l)}$. Here $\odot$ denotes the element-wise product. The full estimate of the ELBO is

$$\hat{\mathcal{L}}(\boldsymbol{\theta},\boldsymbol{\phi};\mathbf{x}_i) = \frac{1}{2}\sum_{j=1}^{J}\left(1+\log\boldsymbol{\sigma}_j^2(\mathbf{x}_i)-\boldsymbol{\mu}_j^2(\mathbf{x}_j)-\boldsymbol{\sigma}_j^2(\mathbf{x_i})\right)+\frac{1}{L}\sum_{l=1}^{L}\log p_{\boldsymbol{\theta}}(\mathbf{x}_i|\boldsymbol{\mu}(\mathbf{x}_i)+\boldsymbol{\sigma}(\mathbf{x}_i)\odot\boldsymbol{\epsilon}_{(i,l)}).$$

This expression is then optimized jointly over $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ using stochastic optimization where an average is used over a mini-batch of data.

Connection to the Autoencoder

Autoencoders are a class of unsupervised, deterministic models where two functions, typically neural networks, are trained jointly in order to reconstruct the input. The first network $g_{\boldsymbol{\phi}}$, called the *encoder*, maps from the input $\mathbf{x}$ to some lower-dimensional variable $\mathbf{z}$. Then another network $f_{\boldsymbol{\theta}}$, the *decoder*, maps back to the dimension of the input, into a new variable $\mathbf{x}^*$. The full model is then trained using a loss function $\mathcal{L}(\mathbf{x},\mathbf{x}^*) = \mathcal{L}(\mathbf{x}, g_{\boldsymbol{\phi}}(f_{\boldsymbol{\theta}}(\mathbf{x})))$ which measures the closeness between the input data and the reconstructed data. We enforce a bottleneck by passing through the low-dimensional variable $\mathbf{z}$, which then becomes a compression of the data. If the dimension of $\mathbf{z}$ and $\mathbf{x}$ were the same, the network would learn the trivial identity mapping. However, by using the bottleneck structure depicted in Figure 3.4a, we force the network to learn a compact and useful representation. Since the encoder and decoder typically are neural networks, an autoencoder is a way to perform a non-linear dimensional reduction. We can use the encoder to obtain a lower-dimension representation of a new data point. This property is opposed to other dimensional reductions methods, such as PCA, where we have to recompute the representation has to when faced with new data.



(a) Autoencoder

(b) VAE

Figure 3.4: Schematic representation of (a) Autoencoders and (b) VAE

As the name suggests, Variational Autoencoders is similar in structure to a regular autoencoder. Both train two neural networks simultaneously using a bottleneck structure, as illustrated in Figure 3.4, although VAE is a probabilistic model derived using principles from variational inference. The inference network $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is often referred as an encoder, and the generative network the decode r$p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$. However, it is an important distinction that the encoder and decoder in a regular autoencoder are deterministic mappings, while in a VAE, they specify distributions. The distributions are specified using a deterministic function of randomly drawn samples. The fact that noise is injected in the intermediate layer has a regularizing effect. Furthermore, the ELBO has a term that regularizes the approximate posterior towards the prior $p(\mathbf{z})$ leading to a distribution that generalizes better to unseen data. Notice that if we set $\boldsymbol{\sigma}$ to be constantly zero and omit KL divergence, the VAE becomes a regular autoencoder with cross-entropy loss.

3.4 Improvements and related work

Since the inception of VAE in 2013 by Kingma and Welling [2013], it has become a fruitful area of research with several extensions and other related works. The original formulation was concerned with continuous representations and has been widely employed in image modeling, but other domains have since been explored as well. This includes discrete latent representation [Oord et al., 2017], sequential data such as text [Bowman et al., 2015] and temporal data Gregor et al. [2018]. VAE has been successful in both generative modeling and learning useful representations of data. Furthermore, it is among state-of-the-art in semi-supervised learning where the goal is to use a generative model to improve classification [Kingma et al., 2014]. For instance, VAE has been used in a model that achieves good results on classification on handwritten digits (MNIST) while only have been shown ten labeled images for each digit [Maaløe et al., 2016]. In the following, we will briefly review some of the extensions to VAE that we deem relevant to this thesis, while we refer to [Kingma and Welling, 2019, Chapter 4] for a general survey of follow-up work.

Tighter lower bound

One line of research is modifying the ELBO or use other objective functions. Furthermore, VAE places strong assumptions of the model where it often assumes that the posterior is approximately factorial, which may come at a cost of how expressive the model is and how diverse the samples are. Burda et al. [2016] counteracts these challenges by tightening the ELBO using importance sampling. The model, called the importance-weighted autoencoder (IWAE), uses the following lower bound on the marginal log-likelihood

$$\log p(\mathbf{x}) \geq \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\frac{1}{L}\sum_{l=1}^{L}\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}^{(l)}))}{q_{\boldsymbol{\phi}}(\mathbf{z}^{(l)}|\mathbf{x})}\right] = \mathcal{L}_{\mathrm{IW}}^{L}(\boldsymbol{\phi},\boldsymbol{\theta}),$$

where $\mathbf{z}^{(1)},\ldots,\mathbf{z}^{(L)}$ L are the samples drawn from $q_{\boldsymbol{\phi}}(\mathbf{z}^{(l)}|\mathbf{x})$. When $L = 1$ we recover the original ELBO. Burda et al. [2016] furthermore showed that the lower bound becomes increasingly tight for larger L and that it the log-likelihood asymptotically. The downside of this approach is that it is more computationally intensive and does not scale as well.

Note that we can use the importance sampling methodology to estimate the log-likelihood for a datapoint $\mathbf{x}$. As VAE only uses a lower bound to estimate the model, there is no explicit way of evaluate the model using the log-likelihood. It is then possible to estimate it using

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

Flexible posteriors

In addition to modifying the ELBO, there have been several works on how to explicitly modify the approximate posterior to increase flexibility and go beyond the Gaussian assumption. One way of doing this is by using *normalizing flows*, as introduced in the context of gradient based variational inference by Rezende and Mohamed [2015b]. Using normalizing flows, the posterior is specified by a series of invertible transformations from a simple base density. Assume that $\boldsymbol{\epsilon}$ follows a base distribution $p_{\boldsymbol{\epsilon}}(\boldsymbol{\epsilon})$ and that $\mathbf{z}$ is a expressed as a deterministic transformation from a sample from the base distribution, $\mathbf{x} = f(\boldsymbol{\epsilon})$. We require that $\mathbf{z}$ and $\boldsymbol{\epsilon}$ have the same dimension and that the transformation f is invertible, and both f and f^{-1} differentiable, such that the distribution of $\mathbf{x}$ can be expressed through a change of variable,

$$p_{\mathbf{x}}(\mathbf{z}) = p_{\boldsymbol{\epsilon}}(\boldsymbol{\epsilon})|\det J_f(\boldsymbol{\epsilon})| \quad \text{where} \quad \boldsymbol{\epsilon} = f^{-1}(\mathbf{z}),$$

and $J_f(\boldsymbol{\epsilon})$ denotes the Jacobian. Invertible and differentiable transformations are composable, meaning that we can stack several transformations giving increasing flexibility without losing the abilities to compute the $p_{\mathbf{z}}(\mathbf{z})$. The chain of transformation is then for $t = 1,\ldots,T$ described as $\mathbf{z}_t = f_t(\mathbf{z}_{t-1})$ where $\mathbf{z}_0 = \boldsymbol{\epsilon}$ and $\mathbf{z} = \mathbf{z}_T$. However, the computation time cost of the Jacobian is in general cubic in the dimension of the data, which becomes intractable for most cases. Therefore the transformations need to be designed in such a way that yields tractable computations in at most linear time. The can for instance be done using *autoregressive flows* where we impose a ordering of the coordinates of the data and condition on previous values. Specifically, the coordinate-wise transformations are on the form

$$z'_i = \tau(z_i, \boldsymbol{\lambda}_i) \quad \text{where} \quad \boldsymbol{\lambda}_i = c_i(\mathbf{z}_{<i}). \tag{3.6}$$

Here τ is called the *transformer*, and this needs to be invertible as we described above. The function c_i is the *i-conditioner* and is a function of the previous coordinates of $\mathbf{z}$, denoted by $\mathbf{z}_{<i}$. The conditioner does not need to be invertible, but the output $\boldsymbol{\lambda}_i$ acts as input to the transformer. This way, the transformation can be flexible, but since the structure is autoregressive, the Jacobian is the product of the diagonal. Hence, the computational time cost will be at most linear.

We will illustrate a normalizing flow in the context of a latent variable model and flexible posteriors in VAE using *Inverse Autoregressive Flows* (IAF) proposed by Kingma et al. [2016]. Recall that the aim is to model the approximate posterior $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ and go beyond the Gaussian case where we used the reparametrization trick to estimate the gradient. IAF is similar to the Gaussian reparametrization, where it in addition combines a autoregressive flow. First, it encodes the data by specifying the posterior $\mathbf{z}_0 \sim q(\mathbf{z}_0|\mathbf{x})$. As done in amortized inference, the variational parameters are predicted using a deterministic function, here a neural network, as $\boldsymbol{\mu}_0 = \boldsymbol{\mu}_0(\mathbf{x})$ and $\boldsymbol{\sigma}_0 = \boldsymbol{\sigma}_0(\mathbf{x})$. Then we draw a sample from the base distribution $\boldsymbol{\epsilon} \sim \mathrm{N}(\mathbf{0}, \mathbf{I})$ and use the familiar transformation $\mathbf{z}_0 = \boldsymbol{\mu}_0 + \boldsymbol{\sigma}_0 \odot \boldsymbol{\epsilon}_0$ to initialize the flow. Furthermore, a representation of the $\mathbf{h} = \mathbf{h}(\mathbf{x})$ is used in each subsequent step in the flow. The flow is is then specified by

$$\mathbf{z}_t = \boldsymbol{\mu}_t + \boldsymbol{\sigma}_t \odot \mathbf{z}_{t-1} \quad \text{where} \quad \boldsymbol{\mu}_t, \boldsymbol{\sigma}_t = c(\mathbf{z}_{t-1}, \mathbf{h}).$$

Note the connection here to Equation (3.6), where the affine transformation to the left in the transformer, and the variational parameters are given by the conditioner. This transformation is done T times where the final iterate $\mathbf{z} = \mathbf{z}_T$ is sample from a flexible posterior $q(\mathbf{z}|\mathbf{x})$ that is used in the final estimate of the ELBO. Since we increase the flexibility of the posterior the ELBO becomes a tighter estimate of the log-likelihood. Note that since all transformation are invertible and differentiable, we can still use stochastic gradient-based optimization.

VAE/GAN

Both VAE and GAN, as we described in Section 2.4, specify generative models using deep neural networks, thereby the name deep generative models. Furthermore, normalizing flows can also be used in a purely generative model to map from a base distribution to a marginal density. Using the previous terminology, GANs are implicit models that use an adversarial process to estimate the generative process. VAE and normalizing flows are prescribed models that specify a density over the full support of the data. Both approaches have their advantages. GANs typically give better samples, while VAE is better for representation learning since it uses latent variables with an encoder, and it often gives better likelihood estimates. Furthermore, it is typically easier to train a VAE. Training GAN requires a careful balance between the generator and the discriminator, and much of the literature on GAN is concerned with improving convergence and stability. There have been several works on unifying the two directions to get the best of both worlds Makhzani et al. [2015], Larsen et al. [2015], Mescheder et al. [2017], one of which we will describe here called *Adversarial Variational Bayes* (AVB) [Mescheder et al., 2017].

AVB introduce a representation of a given $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ using a discriminator that will distinguish samples from the latent prior $p(\mathbf{z})$ and the approximate posterior $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$. Specifically, let $T(\mathbf{x}, \mathbf{z})$ be the discriminator parameterized by neural network, and the objective function for a single data point $\mathbf{x}$ is

$$\max_T \left\{ \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \left[\log \sigma(T(\mathbf{x}, \mathbf{z}) \right] + \mathbb{E}_{p(\mathbf{z})} \left[\log(1 - \sigma(T(\mathbf{x}, \mathbf{z}))) \right] \right\}.$$

Here $\sigma(t) = (1 - e^{-t})^{-1}$ is the sigmoid function that ensure an output in the range $[0, 1]$. Note the similarity to the objective function for a GAN as seen in Equation (2.8) as the cross-entropy loss. The same interpretation applies where the discriminator tries to distinguish pairs $(\mathbf{x}, \mathbf{z})$ sampled independently from the dataset using either $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ or $p(\mathbf{z})$. For a fixed $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ we can differentiate the expression to find the optimal discriminator similar to what we did for a regular GAN. It turns out the optimal discriminator is on the form,

$$\sigma(T^*(\mathbf{x}, \mathbf{z})) = \frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) + p(\mathbf{z})},$$

or equivalently,

$$T^*(\mathbf{x},\mathbf{z}) = \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) - \log p(\mathbf{z}).$$

Now we recognize this term in the ELBO we use the following relation, $D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[T^*(\mathbf{x},\mathbf{z})]$, which allows us to rewrite the ELBO as

$$\mathcal{L}(\boldsymbol{\phi},\boldsymbol{\theta}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log\left(p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right) - T^*(\mathbf{x},\mathbf{z})\right].$$

Practically, the model is estimated in a two-step procedure where the optimal discriminator T^* is estimated before using it in the ELBO to estimate $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$. AVB allows for estimating the KL term in the ELBO using an arbitrarily complex function instead of relying on distributions with closed form expressions, thus using adversarial training to make the approximate posterior more expressive.

Chapter 4

Disentangled Representations in VAE

In Section 2.5 we discussed representation learning and the concept of *disentanglement*. A commonly accepted assumption is that a set of unknown, underlying factors generate the observed data. Furthermore, these factors should be interpreted in terms of how we semantically describe the data. For instance, in the image domain, such factors can be the position of an object, color, lighting, shape, and size. An observer may use different words to describe the data in terms of factors of variation, and a disentangled representation should reflect these properties. Furthermore, it should discard as little information as possible. This property is especially important in an unsupervised setting, where the representation may be used for a variety of tasks. Th we may not know beforehand what information is irrelevant and can be considered nuisance variables. In this section, we will motivate disentangled representation learning in VAE, before considering four models that modify the objective function to impose the desired structure. Lastly, we consider the problem of evaluating disentanglement, and we describe three different metrics.

4.1 Representation learning with a latent variable model

In terms of a latent variable model, we want the underlying generative factors to correspond to the coordinates in the latent variable $\mathbf{z}$. Recalling the latent variable setup, we have an unknown density $p^*(\mathbf{x}, \mathbf{z})$ over the observed data $\mathbf{x}$ and unobserved latent variables $\mathbf{z}$. We model the generative process by a parametric transformation $G_{\boldsymbol{\theta}} : \mathcal{Z} \to \mathcal{X}$ with the corresponding density $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ and some specified prior of the latent variables. Assume that the datapoint $\mathbf{x}$ have dimension K, which typically is much larger than the dimension of the generative factors, $J \ll K$. In representation learning we want to invert the generative process to recover the latent representation $\hat{\mathbf{z}} = f(\mathbf{x})$. Here $\hat{\mathbf{z}}$ would typically correspond to a sample from an estimated posterior distribution over $\mathbf{z}$. A common practice to enforce a disentangled behaviour in the representation, is to assume that the latent components are independent and hence that the prior factorizes. The joint density can then be factorized as

$$p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) = p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) \prod_{j=1}^{J} p(z_j).$$

Although independence is a common model assumption, it is important to note that it does not necessarily align with the of a disentangled behavior. While the literature agrees on desired properties and intuition of disentanglement, a precise definition is still debatable. Many popular models use the latent variables' independence as an operational definition for a disentangled and interpretable representation [Higgins et al., 2019, Kim and Mnih, 2018]. Such models will also typically benchmark on synthetically built datasets where the underlying factors are independent by constructions, and all samples are i.i.d., in which case independence is a reasonable claim. Moreover, several evaluation metrics [Higgins et al., 2019, Kim and Mnih, 2018] require the generative factors to be known, limiting its applicability to synthetic data. A typical dataset is *2D Shapes* which we consider later in Chapter 5. Each sample is generated from 5 underlying factors that the model should be able to recover. Since the ground truth is known, we can construct reasonable evaluation metrics. The real world, however, is, of course, more complex. We cannot, in general, assume i.i.d data and generative factors may be correlated or organized in structural dependencies. There have been several works that refine the definition of disentanglement. Higgins et al. [2018] pursue independent groups in a topological sense and Schölkopf [2019] disentanglement in terms of causal factorization using structural dependencies.

Conceptually, the idea of recovering generative factors is not new in statistics and machine learning [Jolliffe, 1986, Hyvärinen and Oja, 2000]. However, the recent endeavor of deep generative models provides a framework that scales to large datasets of high dimension. Of particular interest are models that explore disentanglement based on VAE. VAE naturally provides an inference model of the latent variable, and current state-of-the-art models in disentanglement follow this path with additional structure or modifications. Within this framework, there is a distinction between fully unsupervised methods and methods that use some sort of supervision. Semi-supervised methods that have access to the true generative factors for a subset of the data have also been used to learn disentangled representations [Bouchacourt et al., 2017, Kingma et al., 2014, Siddharth et al., 2017]. Our focus in this thesis will be unsupervised methods.

4.2 β-VAE

Higgins et al. [2019] extended the VAE model by emphasizing disentangled representation in latent variable, which have since spurred other works following the same line. The model, termed β-VAE introduces an extra hyperparameter in the ELBO objective on the form,

$$\mathcal{L}^{\beta}(\boldsymbol{\phi}, \boldsymbol{\theta}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log(p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right] - \beta D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})).$$

The ELBO is estimated across samples from the true and unknown data generating distribution $p*(\mathbf{x})$, meaning that we estimate the term $\mathbb{E}_{p*(\mathbf{x})}[\mathcal{L}^{\beta}(\boldsymbol{\phi}, \boldsymbol{\theta})]$. As we have discussed in Section 3.2, the first term in the ELBO encourages reconstruction of the input, while the second terms can be interpreted as a regularizer of the approximate posterior with respect to the prior. When $\beta = 1$ we recover the original ELBO, and the authors argue that $\beta > 1$ to encourage disentanglement. The objective can also be interpreted as a constrained optimization problem, where the aim is to maximize the reconstruction term while keeping $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ close to $p(\mathbf{z})$. The typical case is that the prior is unit Gaussian while the posterior is a Gaussian with diagonal covariance. β-VAE enforces a trade-off between the two terms, aiming to limit the capacity of the latent bottleneck while still being able to reconstruct the data. When proposed, β-VAE achieved impressive empirical results, but was mainly heuristically motivated. There have since been interest in examining why VAE, and especially β-VAE, disentangles, some of which we will present here.

Recall from Section 2.5 that the Information Bottleneck describes an objective that aims to maximize the mutual information between a representation $\hat{Z}$ and a task Y while constraining information with regards to the input data X. This framework appears in a supervised setting, and Y is typically a classification task. However, we are interested in the unsupervised setting, where the task is the reconstruction of the input data. In this case, Y is a reconstruction X'. Conceptually, the information bottleneck and the ELBO are similar in that both consist of a reconstruction term and a regularization term. Burgess et al. [2018] motivate β-VAE by the Information Bottleneck and consider the case where both the prior and variational distribution are factorized Gaussians. Then the closed form of the KL divergence is

$$D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i) \| p(\mathbf{z})) = \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log \boldsymbol{\sigma}_j^2 - \boldsymbol{\mu}_j^2 - \boldsymbol{\sigma}_j^2\right).$$

For each datapoint $\mathbf{x}_i$ there is a posterior distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ with mean $\boldsymbol{\mu}_i$ and variance $\boldsymbol{\sigma}_i$. The KL divergence is zero when the posterior matches the prior, i.e., both $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i$ are always zero in all coordinates. The KL term can be decreased by dispersing the posterior means or increasing the posterior variances. By increasing β, there would be a greater overlap of the poster distribution. In the case of too much overlap, the latent variable conveys little information about the observed datapoint, making it hard to differentiate the reconstructions and for the latent variables to have a clear meaning. In the other end of the spectrum with $\beta = 0$, the model would yield good reconstruction, but the latent variables would collapse to point estimates reducing their meaning to a look-up table. Hence varying β allows to control for an appropriate level of overlap of the posterior distribution where points close in the input space is also close in the latent space. Mathieu et al. [2018] echoes this point, while extending the discussion to a general prior $p(\mathbf{z})$, possibly multimodal. They also claim that appropriate overlap of the variational posteriors $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is not sufficient for disentanglement and that is necessary to enforce structure on the variational distribution

$$q_{\boldsymbol{\phi}}(\mathbf{z}) = \mathbb{E}_{p^*(\mathbf{x})}[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})]. \tag{4.1}$$

This is also called the *aggregated posterior* [Tomczak and Welling, 2017]. We illustrate how β controls the overlap between the posterior distribution in Figure 4.1. There is a range of methods that explicitly enforce a structure on the aggregated posterior. We will continue the discussion about the role of the aggregated posterior in Section 4.3.

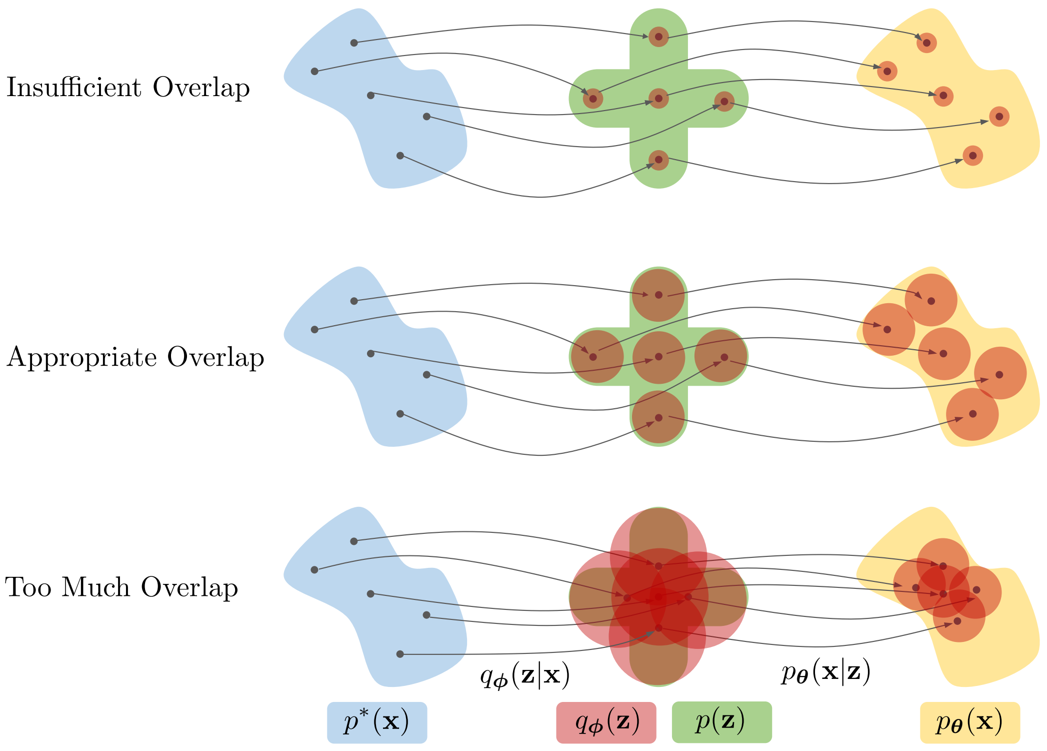

Insufficient Overlap
Appropriate Overlap
Too Much Overlap
$q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$
$p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$
$p^*(\mathbf{x})$
$q_{\boldsymbol{\phi}}(\mathbf{z})$
$p(\mathbf{z})$
$p_{\boldsymbol{\theta}}(\mathbf{x})$

Figure 4.1: Illustration of the effect β in the latent space of β-VAE. The prior $p(\mathbf{z})$ desired target structure in the latent space. With insufficient overlap (top), the model becomes deterministic, and the latent space becomes a look-up table. With too much overlap (bottom), we cannot differentiate between samples, and the latent space contains little information. In this case, the generated samples have little diversity. The illustration is adapted from [Mathieu et al., 2018].

Although Burgess et al. [2018] draws inspiration from the Information Bottleneck to explain β-VAE to some extent, they does not show the mathematical relationship. Here we will fill out the details and show that the regularized ELBO, $\mathcal{L}^{\beta}$, is in fact an upper bound. A similar result is shown by Alemi et al. [2016], however, they consider the supervised case and a variational approximation to the decoder $p(\mathbf{x}|\mathbf{z})$. In this case, we are dealing with the variational posterior $q_{\boldsymbol{\phi}}(\mathbf{z_i}|\mathbf{x}_i)$ and the data distribution $p^*(\mathbf{x})$. This induces the the aggregated posterior in Equation (4.1) and the joint distribution $q_{\boldsymbol{\phi}}(\mathbf{z},\mathbf{x}) = p^*(\mathbf{x})q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$. We also have the stochastic decoder $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$. The Information Bottleneck can then be written as.

$$\mathcal{L}_{\text{IB}}(\theta,\phi) = I_{\boldsymbol{\phi}}(\mathbf{x},\mathbf{z}) - \beta I_{\boldsymbol{\phi}}(\mathbf{x}',\mathbf{z}).$$

Here the notation $I_{\boldsymbol{\phi}}$ emphasizes the dependence on the parameters $\boldsymbol{\phi}$ in our inference model. We'll assume to work a infinite model family, meaning that that our parametric distribution corresponds to the underlying distributions, in which case these parametric terms are exactly the mutual information $I(\mathbf{x},\mathbf{z})$.

We first consider the term $I_{\boldsymbol{\phi}}(\mathbf{x},\mathbf{z})$. Using the relation $q_{\boldsymbol{\phi}}(\mathbf{z},\mathbf{x}) = q_{\boldsymbol{\phi}}(\mathbf{x}|\mathbf{z})q_{\boldsymbol{\phi}}(\mathbf{z})$, we can rewrite $I_{\boldsymbol{\phi}}$ as

$$\begin{aligned}
I_{\boldsymbol{\phi}}(\mathbf{x},\mathbf{z}) &= \int\int q_{\boldsymbol{\phi}}(\mathbf{z},\mathbf{x}) \log\left(\frac{q_{\boldsymbol{\phi}}(\mathbf{z},\mathbf{x})}{p^*(\mathbf{x})q_{\boldsymbol{\phi}}(\mathbf{z})}\right) d\mathbf{x}d\mathbf{z} \\
&= \int \left(\int q_{\boldsymbol{\phi}}(\mathbf{x}|\mathbf{z}) \log\left(\frac{q_{\boldsymbol{\phi}}(\mathbf{x}|\mathbf{z})}{p^*(\mathbf{x})}\right) d\mathbf{x}\right) q_{\boldsymbol{\phi}}(\mathbf{z})d\mathbf{z} \\
&= \int \left(\int q_{\boldsymbol{\phi}}(\mathbf{x}|\mathbf{z}) \log q_{\boldsymbol{\phi}}(\mathbf{x}|\mathbf{z})d\mathbf{x} - \int q_{\boldsymbol{\phi}}(\mathbf{x}|\mathbf{z}) \log p^*(\mathbf{x})d\mathbf{x}\right) q_{\boldsymbol{\phi}}(\mathbf{z})d\mathbf{z}.
\end{aligned}$$

To establish a lower bound on the mutual information, we use the fact that the KL divergence is non-negative. Specifically, we compare the induced distribution $q_{\boldsymbol{\phi}}(\mathbf{x}|\mathbf{z})$ and the stochastic decoder $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$. Rewriting the inequality $D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{x}|\mathbf{z})\|p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})) \geq 0$ gives

$$\int q_{\boldsymbol{\phi}}(\mathbf{x}|\mathbf{z}) \log q_{\boldsymbol{\phi}}(\mathbf{x}|\mathbf{z})d\mathbf{x} \geq \int q_{\boldsymbol{\phi}}(\mathbf{x}|\mathbf{z}) \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})d\mathbf{x},$$

which we can insert in the expression for $I_{\boldsymbol{\phi}}$,

$$\begin{aligned}
I_{\boldsymbol{\phi}}(\mathbf{x},\mathbf{z}) &\geq \int \left(\int q_{\boldsymbol{\phi}}(\mathbf{x}|\mathbf{z}) \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})d\mathbf{x} - \int q_{\boldsymbol{\phi}}(\mathbf{x}|\mathbf{z}) \log p^*(\mathbf{x})d\mathbf{x}\right) q_{\boldsymbol{\phi}}(\mathbf{z})d\mathbf{z} \\
&= \int \left(\int q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})d\mathbf{z} - \int q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \log p^*(\mathbf{x})d\mathbf{z}\right) p^*(\mathbf{x})d\mathbf{x}.
\end{aligned}$$

In this line we have simply reformulated the joint $q_{\boldsymbol{\phi}}(\mathbf{x}|\mathbf{z})q_{\boldsymbol{\phi}}(\mathbf{z}) = q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})p^*(\mathbf{x})$ and switched the order of integration. Finally we can marginalize over $\mathbf{z}$ over the second term, and by doing that we recover the data entropy $H(\mathbf{x})$. Finally, we write the lower bound as

$$I_{\boldsymbol{\phi}}(\mathbf{x},\mathbf{z}) \geq \mathbb{E}_{p^*(\mathbf{x})}\left[\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right]\right] + H(\mathbf{x}).$$

Now we turn our attention to the second term that measures the mutual information between a latent variable and the reconstruction $I_{\boldsymbol{\phi}}(\mathbf{x}',\mathbf{z})$. Note that we are looking for an upper bound since the regularization term ELBO is negated.

$$\begin{aligned}
I_{\boldsymbol{\phi}}(\mathbf{x},\mathbf{z}) &= \int\int p^*(\mathbf{z})q_{\boldsymbol{\phi}}(\mathbf{z},\mathbf{x}) \log\left(\frac{q_{\boldsymbol{\phi}}(\mathbf{z},\mathbf{x})}{q_{\boldsymbol{\phi}}(\mathbf{z})}\right) d\mathbf{x}d\mathbf{z} \\
&\leq \int\int p^*(\mathbf{z})q_{\boldsymbol{\phi}}(\mathbf{z},\mathbf{x}) \log\left(\frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}{p(\mathbf{z})}\right) d\mathbf{x}d\mathbf{z} \\
&= \mathbb{E}_{p^*(\mathbf{x})}\left[D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))\right],
\end{aligned}$$

where we have used the fixed prior on $\mathbf{z}$. Putting the two inequalities together and comparing the Information Bottleneck with the regularized ELBO, we see that

$$\mathcal{L}_{\mathrm{IB}} \geq \mathbb{E}_{p^*(\mathbf{x})}[\mathcal{L}^{\beta}_{\mathrm{ELBO}}(\mathbf{x})] + H(\mathbf{x}) \tag{4.2}$$

We approximate this expression using samples from the empirical distribution of the data. By maximizing the objective for β-VAE, we approximate the information bottleneck from below. However, the inequality can never be tight as the entropy $H(\mathbf{x})$ is a property of the data and out of our control. It is important to note that these bounds are derived using strong assumptions on our functions' modeling capacity, which will lead to a larger gap in the inequalities. Moreover, many of the distributions used during the derivations are unavailable or impractical to evaluate. For instance, the mutual information is, in general, hard to evaluate since we do not have access to the true data density $p^*(\mathbf{x})$. In that sense, the objective for β-VAE is easy to evaluate and can give a crude approximation.

Alemi et al. [2017] analyzes these bounds on the mutual information further by deconstructing the trade-off between reconstruction and regularization in a phase diagram. Define the following terms that shows up in the previously derived inequalities,

$$\begin{aligned}
H &= -\mathbb{E}_{p^*(\mathbf{x})}[-\log p^*(\mathbf{x})] \\
D &= -\mathbb{E}_{p^*(\mathbf{x})}\left[\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[-\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right]\right] \\
R &= \mathbb{E}_{p^*(\mathbf{x})}\left[D_{\mathrm{KL}}\left(q_{\boldsymbol{\theta}}(z|x)\|p(z)\right)\right].
\end{aligned}$$

H is the entropy, D is negative reconstruction log-likelihood here termed the *distortion*, and R is the *rate*, namely the average KL divergence between the variational posterior and the specified prior. The objective for β-VAE over the data distribution can now be written as $\mathcal{L}^{\beta}_{\text{ELBO}} = -(D + \beta R)$ with the optimal β given by $\beta = \frac{\partial D}{\partial R}$. The two inequalities can be summarized as

$$H - D \leq I_{\phi}(\mathbf{x}, \mathbf{z}) \leq R.$$

Alternatively, instead of thinking of β as fixed, we can consider the case where the rate is fixed and then writing the optimal distortion as a function $D(R)$. Such a phase diagram is displayed in Figure 4.2. The bottom horizontal line corresponds to $\beta = 0$ in which case we can perfectly encode and decode the data. The lowest possible rate is then then entropy of the data, $R = H$. The left vertical line is when $\beta = \infty$ and all the weight is on the regularization. Then this forces the approximate posterior to match the prior, thus the latent representation is not encoding any useful information. Note that the ELBO is constant on the line $D = H - R$, thus highlighting that there is no incentive to choose a point with high rate over high distortion when $\beta = 1$. Furthermore, for any β it suggests that better disentangling behavior comes at the cost of the ability to reconstruct. These lines are based on the assumption of an infinite function family, and practically the bounds will not be tight. However, the inequalities must still hold suggesting that a optimal frontier $D(R)$ exists.

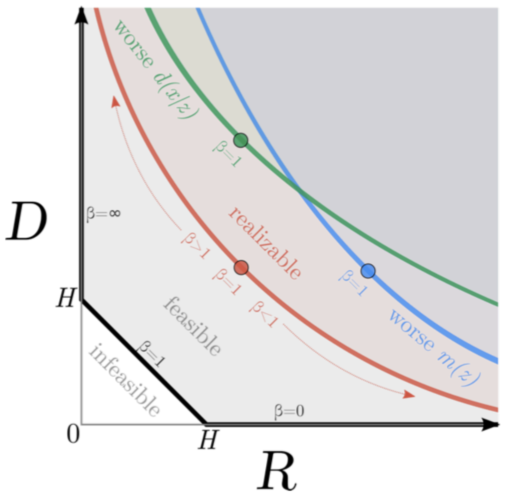


Figure 4.2: Phase diagram for the rate and distortion. The distortion (D) measures the reconstruction error of points in the data set. The rate (R) is the KL divergence between the variational distribution and prior. The curves represent the function $D(R)$ for a fixed β. We have adapted the figure from [Alemi et al., 2017]

Schematic representation of the phase diagram in the RD-plane. The distortion (D) axis measures the reconstruction error of the samples in the training set. The rate (R) axis measures the relative KL divergence between the encoder and our marginal approximation. The thick black lines denote the feasible boundary in the infinite model capacity limit.

As the previous discussion suggests, β provides a way to control the overlap of latent variables and the trade-off between the rate and distortion. However, this is not a sufficient explanation of why β-VAE is observed to learn disentangled representation. More specifically, it does not explain why the latent variables' axis aligns with generative factors of variation. The ELBO is constructed to provide a feasible objective for the log-likelihood and not specifically for representation learning. For instance, the objective is invariant to rotational changes in latent representation [Mathieu et al., 2018]. Hence the learned representation is not any meaningful than any arbitrary rotation of the factors. Rolinek et al. [2018] takes a step towards understanding the inner workings of VAE by connecting it to PCA. They give theoretical arguments of a factorial variational posterior, and the ELBO allows for the representation of VAE to align with the local principal components directions, which is favorable for enforcing disentangled representation.

4.3 Regularizing the aggregated posterior

Since the proposed β-VAE model, several other works have been developed to improve the observed disentangling behavior. A drawback of β-VAE is that it sacrifices the ability to reconstruct the input as β increases, and the disentangled behavior emerges. In the following, we will compare methods that augment the regularization to include the aggregated posterior.

Recall the aggregated posterior in Equation (4.1) denotes as $q_\phi(\mathbf{z})$. Where $q_\phi(\mathbf{z}|\mathbf{x})$ is the variational distribution of the representation $\mathbf{z}$ for a given datapoint $\mathbf{x}$, can the aggregated posterior be interpreted as marginal distribution of the representation for the entire dataset. A previous assumption, an a common operational definition for disentanglement, is that the prior $p(\mathbf{z})$ is factorized. In the standard β-VAE setup we ensure that the variational posterior is close to the prior, but there is no restrictions on the aggregated posterior. Since we assume that the factors vary independently, is is natural consider that the aggregated posterior ideally factorizes as well. Denote this ideal factorized distribution as $\overline{q}_\phi(\mathbf{z}) = \prod_{i=1}^N q_\phi(z_i)$. Note that aggregated posterior is stated using the empirical distribution over the data, meaning that we can estimate using the observed samples. However, we'll make no distinction in notation and by equality we let

$$q_\phi(\mathbf{z}) = \frac{1}{N}\sum_{i=1}^N q_\phi(\mathbf{z}_i|\mathbf{x}_i).$$

To inspect the effect of the aggregated posterior, we van rewrite the KL divergence. First, using the joint distribution $q_\phi(\mathbf{z},\mathbf{x}) = q_\phi(\mathbf{z}|\mathbf{x})p^*(\mathbf{x})$ and then expanding the logarithm, we get

$$\begin{aligned}
\mathbb{E}_{p^*(\mathbf{x})}\left[D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))\right] &= \mathbb{E}_{p^*(\mathbf{x})}\left[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p(\mathbf{z})}\right]\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z},\mathbf{x})}\left[\log\frac{q_\phi(\mathbf{z},\mathbf{x})}{p^*(\mathbf{x})p(\mathbf{z})}\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z},\mathbf{x})}\left[\log\frac{q_\phi(\mathbf{z},\mathbf{x})}{p^*(\mathbf{x})q_\phi(\mathbf{z})}\right] + \mathbb{E}_{q_\phi(\mathbf{z},\mathbf{x})}\left[\log\frac{q_\phi(\mathbf{z})}{p(\mathbf{z})}\right] \\
&= I_\phi(\mathbf{x},\mathbf{z}) + D_{\mathrm{KL}}(q_\phi(\mathbf{z})\|p(\mathbf{z})). \qquad (4.3)
\end{aligned}$$

Note that we in the last line recognise the mutual information defined using the joint distribution $q_\phi(\mathbf{z},\mathbf{x})$, and in the ladder term we can marginalize $\mathbf{x}$ to recover the KL divergence. Penalizing the first term, $I_\phi(\mathbf{x},\mathbf{z})$, restricts the information $\mathbf{z}$ can carry about $\mathbf{x}$, which may lead to poorer reconstruction. Penalizing $D_{\mathrm{KL}}(q_\phi(\mathbf{z})\|p(\mathbf{z}))$ pushes the aggregated posterior towards the factorial prior. β-VAE penalized both terms equally, and a large β can force the bottleneck to discard information about the data. We can dissect the expression further by introducing the factorized aggregated posterior, $\overline{q}_\phi(\mathbf{z})$,

$$\begin{aligned}
D_{\mathrm{KL}}(q_\phi(\mathbf{z})\|p(\mathbf{z})) &= \mathbb{E}_{q_\phi(\mathbf{z})}\left[\log\frac{q_\phi(\mathbf{z})}{\overline{q}_\phi(\mathbf{z})}\frac{\overline{q}_\phi(\mathbf{z})}{p(\mathbf{z})}\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z})}\left[\log\frac{q_\phi(\mathbf{z})}{\overline{q}_\phi(\mathbf{z})}\right] + \mathbb{E}_{q_\phi(\mathbf{z})}\left[\sum_{i=1}^N\log\frac{q_\phi(z_i)}{p(z_i)}\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z})}\left[\log\frac{q_\phi(\mathbf{z})}{\overline{q}_\phi(\mathbf{z})}\right] + \sum_{i=1}^N\mathbb{E}_{q_\phi(z_i)}\left[\log\frac{q_\phi(z_i)}{p(z_i)}\right] \\
&= D_{\mathrm{KL}}(q_\phi(\mathbf{z})\|\overline{q}_\phi(\mathbf{z})) + \sum_{i=1}^N D_{\mathrm{KL}}(q_\phi(z_i)\|p(z_i)).
\end{aligned}$$

Putting this all together, the KL divergence in the ELBO can be decomposed in three terms,

$$\mathbb{E}_{p^*(\mathbf{x})}\left[D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))\right] = \underbrace{I_\phi(\mathbf{x},\mathbf{z})}_{\text{(1) Mutual Information}} + \underbrace{D_{\mathrm{KL}}(q_\phi(\mathbf{z})\|\overline{q}_\phi(\mathbf{z}))}_{\text{(2) Total Correlation}} + \underbrace{\sum_{i=1}^N D_{\mathrm{KL}}(q_\phi(z_i)\|p(z_i))}_{\text{(3) Dimension KL}}. \qquad (4.4)$$

Here the total correlation encourages the aggregated posterior to match its factorial counterpart, thereby aiming towards independence in the coordinates. β-VAE weights all three terms equally, while regularizing on the mutual information may not be desirable for disentanglement in which it can lead to poor reconstructions. The decomposing in Equation (4.4) inspires several methods to regularize on the structure of the aggregated posterior, either in the form of total correlation or some terms with regards to the prior.

FactorVAE

FactorVAE [Kim and Mnih, 2018] uses the total correlation in the following objective,

$$\mathcal{L}_{\text{FactorVAE}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathcal{L}_{\text{VAE}}(\boldsymbol{\theta}, \boldsymbol{\phi}) + \lambda D_{\text{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}) \| \overline{q}_{\boldsymbol{\phi}}(\mathbf{z})),$$

with λ being a hyperparameter controlling the total correlation. It should be noted total correlation is not a sufficient condition for disentanglement, albeit it is a necessary condition. For instance, if the variational posterior is standard normal, the distribution is factorial but carries not useful information. Therefore total correlation is only meaningful when working in tandem with the reconstruction term.

The aggregate posterior and the factorized version is stated as a mixture over the empirical distribution, meaning that we need all the data points to evaluate them. This is often computationally demanding, and we want an objective that is catered to batching. FactorVAE circumvents this by using samples and a classifier. First, note that we can sample from $q_{\boldsymbol{\phi}}(\mathbf{z})$ by first sampling uniformly from the dataset then drawing samples from $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_i)$. Similarly, to draw samples from $\overline{q}_{\boldsymbol{\phi}}(\mathbf{z})$ by sampling from $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_i)$ and ignoring all but one coordinate. A deterministic classifier D is then trained to distinguish samples from the two distributions. By conditioning on a binary label $c = \{0, 1\}$ whether a sample is from one or the other distribution, the total correlation is estimated as

$$\begin{aligned} D_{\text{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}) \| \overline{q}_{\boldsymbol{\phi}}(\mathbf{z})) &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}\left[\log \frac{q_{\boldsymbol{\phi}}(\mathbf{z}|c=1)}{\overline{q}_{\boldsymbol{\phi}}(\mathbf{z}|c=0)}\right] = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}\left[\log \frac{q_{\boldsymbol{\phi}}(c=1|\mathbf{z})}{\overline{q}_{\boldsymbol{\phi}}(c=0|\mathbf{z})}\right] \\ &\approx \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}\left[\log \frac{D(\mathbf{z})}{1 - D(\mathbf{z})}\right]. \end{aligned}$$

Note that we in the second equality use Bayes' rule and that we assume the marginal class probabilities are equal, $p(c = 1) = p(c = 0) = \frac{1}{2}$. Practically, the classifier D is a neural networks and its parameters are estimated jointly with $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ in an inner loop. The technique of using a classifier is common in density ratio estimation. We recognize the procedure from adversarial training in GAN (Section 2.4) and VAE/GAN hybrids (Section 3.4).

β-TCVAE

β-TCVAE Chen et al. [2018] was proposed in parallel with FactorVAE and utilize the same objective function. However, it introduces an estimator for the total correlation which used instead of the classifier on the density ratio. Consider the term involving $\log q_{\boldsymbol{\phi}}(\mathbf{z})$. Provided with a minibatch $\{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$, the estimator is

$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[\log q_{\boldsymbol{\phi}}(\mathbf{z})] \approx \frac{1}{M} \sum_{i=1}^{M} \left[\log \frac{1}{NM} \sum_{i=1}^{M} q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i)\right].$$

The term involving $\log \overline{q}_{\boldsymbol{\phi}}(\mathbf{z})$ is simularly estimated using the sum of each coordinate. We refer the reader to [Chen et al., 2018, Appendix C] for a derivation of the estimator. The estimator is biased, but on the other hand it does not depend on other parameters that needs to be tuned or some inner optimization loop as in FactorVAE.

DIP-VAE

DIP-VAE [Kumar et al., 2017] follows the same trajectory of regularizing the aggregated posterior but concentrates on imposing a similarity towards the prior. Note that it assumes that the prior factorizes, but this is usually the case for most VAE models. In regularizing towards the prior, DIP-VAE aims to gain better control over the disentangling behavior. This focus is in line with the decomposition we arrived at in Equation (4.3). In general, the regularization term could be any suitable distance or divergence $D(q_{\boldsymbol{\phi}}(\mathbf{z}), p(\mathbf{z}))$. However, as before, we are limited in the ability to evaluate $\log q_{\boldsymbol{\phi}}(\mathbf{z})$ as it depends on the data distribution. The are several possibilities to remedy this. We can either generate samples from both distributions such that a classifier can estimate a KL divergence. This approach is in the spirit of FactorVAE, albeit using the KL divergence with respect to the prior. It is also possible to construct an estimator as in β-TCVAE.

DIP-VAE adopts a simpler and direct approach where it aims to match the second order moments of the two distributions. Denote the covariance matrix of $\mathbf{z}$ with respect to $q_{\boldsymbol{\phi}}(\mathbf{z})$ as

$$\mathbb{V}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[\mathbf{z}] := \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[(\mathbf{z} - \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[\mathbf{z}])(\mathbf{z} - \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[\mathbf{z}])^T].$$

As the prior $p(\mathbf{z})$ is set to be standard normal, the idea is to regularize the covariance matrix towards the identity matrix. The measure of similarity is simply chosen to be the entry-wise l_2 distance, equivelantly stated as the squared Frobenius norm, $\|\mathbb{V}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[\mathbf{z}] - I\|_F^2$. To ensure the the diagonal elements tends towards 1, term is split into diagonal and off-diagonal terms, each with a tuning parameter. This leads to the Disentangled Inferred Prior objective,

$$\mathcal{L}_{\mathrm{DIP-VAE}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathcal{L}_{VAE}(\boldsymbol{\theta}, \boldsymbol{\phi}) - \lambda \sum_{k \neq l} (\mathbb{V}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[\mathbf{z}])_{k,l}^2 - \lambda' \sum_{k} ((\mathbb{V}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[\mathbf{z}])_{k,k} - 1)^2.$$

As we want to maximize the objective, it encourages off-diagonal elements to be small and diagonal elements to be close to one. It would be possible to consider higher-order elements as well. Note that the approach of DIP-VAE restricts the modelling assumptions of the prior to be standard normal. This is the common practice in standard VAE models, but it would not easily convert to more structured domains. Dealing with other prior distribution would require to alter the objective itself.

Other divergences

Other divergences than the common KL divergence poses as another degree of freedom in choosing regularizer on the aggregated posterior. As in DIP-VAE, we consider choosing the divergence $D(q_{\boldsymbol{\phi}}(\mathbf{z}), p(\mathbf{z}))$. Since the aggregated posterior often is not tractable in an analytical form, we rely on divergences that can be estimated using samples.

Similar in spirit to the density ratio trick, Adversarial Autoencoders (AAE) [Makhzani et al., 2015] uses a classifier to discriminate samples from two distribution. However, the objective is stated using the binary Bernoulli loss similar to a GAN. This results in the objective

$$\mathcal{L}_{\mathrm{AAE}}(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\psi}) = \mathcal{L}_{\mathrm{VAE}}(\boldsymbol{\theta}, \boldsymbol{\phi}) + \mathbb{E}_{p(\mathbf{z})}[\log D_{\boldsymbol{\psi}}(\mathbf{z})] + \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[\log(1 - D_{\boldsymbol{\psi}}(\mathbf{z}))].$$

Here the classifier outputs the probability of $\mathbf{z}$ being drawn from the prior, $D(\mathbf{z}) = p(c = 1|\mathbf{z})$. Note also that we introduce an additional set of parameters $\boldsymbol{\psi}$ in the classification model that needs to be tuned jointly with the $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$. Recall the objective function for a GAN in Section 2.4, where it was proven that the objective is equivalent to minimizing the Jensen-Shannon divergence $D_{\mathrm{JS}}(q_{\boldsymbol{\phi}}(\mathbf{z}), p(\mathbf{z}))$ for an optimal classifier $D^*(\mathbf{z})$. An AAE will therefore regularize the aggregated posterior towards the prior using the Jensen-Shannon divergence.

Another class of divergences that are readily available to choose from are f-divergences as described in Section 2.2, which include the KL divergence, Jensen-Shannon, and Total Variation. We rely on estimation techniques that use samples since the analytical form of the aggregated posterior is not available, and we, therefore, do not have a closed expression for $D_f(q_{\boldsymbol{\phi}}(\mathbf{z}), p(\mathbf{z}))$ that can be optimized. One way to alleviate this problem is to use the variational lower bound of f-divergences [Nguyen et al., 2010, Nowozin et al., 2016], which we described previously in Equation (2.2). However, this would include an inner optimization loop for an arbitrary function typically parameterized with another neural network. Another approach that does not entail additional optimization problems is to construct an estimator of f-divergence, which is similar to β-TCVAE. Rubenstein et al. [2019] proposes a general estimator of f-divergences in the framework of VAE and state conditions for which it is unbiased and consistent. Although it is not used for disentanglement in the paper, we could use it to estimate $D_f(q_{\boldsymbol{\phi}}(\mathbf{z}), p(\mathbf{z}))$, or even the total correlation.

Lastly, we consider the class of integral probability metrics (IPM) discussed in Section 2.2. This class includes the Wasserstein distance and the Maximum Mean Discrepancy (MMD). In particular, the MMD was been applied as a regularizer due to the alternative formulation in Equation (2.6) which

Model	$\mathbf{R}_1$	$\mathbf{R}_2$
β-VAE	$D_{\text{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}\vert\mathbf{x})\Vert p(\mathbf{z}))$	
FactorVAE / β-TCVAE		$D_{\text{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z})\Vert\overline{q}_{\boldsymbol{\phi}}(\mathbf{z}))$
DIP-VAE		$\Vert\mathbb{V}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[\mathbf{z}] - I\Vert_F^2$
AAE		$D_{\text{JS}}(q_{\boldsymbol{\phi}}(\mathbf{z}), p(\mathbf{z}))$
InfoVAE	$D_{\text{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}\vert\mathbf{x})\Vert p(\mathbf{z}))$	$D_{\text{MMD}}(q_{\boldsymbol{\phi}}(\mathbf{z}), p(\mathbf{z}))$
WAE		$D_{\text{MMD}}(q_{\boldsymbol{\phi}}(\mathbf{z}), p(\mathbf{z}))$

Table 4.1: Comparison of the models considered.

is favourable for Monte Carlo estimation. Practically, during training this can be estimated using batches. Both Zhao et al. [2017a] (InfoVAE) and Mathieu et al. [2018] propose an objective that encompasses both a regularizer similar to β-VAE and a term involving the aggregated posterior in a MMD,

$$\mathcal{L}_{\text{InfoVAE}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathcal{L}_{\text{VAE}}(\boldsymbol{\theta}, \boldsymbol{\phi}) + \lambda\mathbb{E}_{p^*(\mathbf{x})}[D_{\text{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\Vert p(\mathbf{z}))] + \lambda' D_{\text{MMD}}(q_{\boldsymbol{\phi}}(\mathbf{z}), p(\mathbf{z})).$$

Mathieu et al. [2018] argues that that the first regularization terms determines the appropriate level of overlap in the latent space, as we discussed for β-VAE and illustrated in Figure 4.1. The second term reflects the aim of the aggregated posterior matching the prior where the desired dependency structure is encoded. This holds especially true when the prior is something else than a simple unimodal distribution, which also is illustrated in Figure 4.1 as the target distribution.

Another model that relies on IPM the structural setup of a VAE, are Wasserstein Autoencoders (WAE) [Tolstikhin et al., 2018]. Here the goal is to approximate the Wasserstein distance between true data distribution $p^*(\mathbf{x})$ and the parametric approximation following the latent variable setup, $p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p(\mathbf{z})$. Using Kantorovich-Rubinstein theorem as displayed in Equation (2.5), they arrive at an equivalent formulation of the Wasserstein distance

$$D_{\text{W}}(p^*(\mathbf{x}), p_{\boldsymbol{\theta}}(\mathbf{x})) = \inf_{q_{\boldsymbol{\phi}}(\mathbf{z})=p(\mathbf{z})} \mathbb{E}_{p^*(\mathbf{x})}\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\rho(\mathbf{x}, G_{\boldsymbol{\theta}}(\mathbf{z})].$$

Here ρ is any cost function and $G_{\boldsymbol{\theta}}$ is the deterministic mapping which combined with the prior $p(\mathbf{z})$ induces the distribution $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$. Note that the formulation is stated over an equality constraint, however, this can be relaxed by only requiring that $q_{\boldsymbol{\phi}}(\mathbf{z})$ and $p(\mathbf{z})$ is sufficiently close by some distance D. This results in the WAE objective,

$$\mathcal{L}_{\text{WAE}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{p^*(\mathbf{x})}\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\rho(\mathbf{x}, G_{\boldsymbol{\theta}}(\mathbf{z})] + \lambda D(q_{\boldsymbol{\phi}}(\mathbf{z}), p(\mathbf{z})),$$

where the MMD is used as the distance. We can draw connections from this objective to the ELBO used in regular VAE. If the generative model in a VAE, $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ is Gaussian, the reconstruction terms amounts to simply the l_2 cost function. This can also be achieved in a WAE where $\rho = \Vert\mathbf{x} - \mathbf{z}\Vert_2^2$. Similarly can be done if the generative model is a Bernoulli distribution, i.e., the output is binary. Hence the reconstruction term in VAE and WAE is equivalent by choosing the cost function ρ in accordance to the generative model. The regularizer in VAE can be decomposed into the mutual information $I_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{z})$ and $D_{\text{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z})\Vert p(\mathbf{z}))$ as in (4.3). WAE drops the former term in addition to offer the increased flexibility of choosing another distance. Arjovsky et al. [2017] consider both a GAN based distance and the MMD.

As we have seen, there are several ways of combining different kinds of regularization in the VAE objective to encourage a disentangled behavior. The regularization terms are added to enforce a desired property of either the approximate posterior $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ or the aggregate posterior $q_{\boldsymbol{\phi}}(\mathbf{z})$. To summarize, the models considered in the previous sections be be subsumed into an objective on the form

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathcal{L}_{\text{VAE}}(\boldsymbol{\theta}, \boldsymbol{\phi}) + \lambda\mathbb{E}_{p^*(\mathbf{x})}[R_1(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}))] + \lambda' R_2(q_{\boldsymbol{\phi}}(\mathbf{z})),$$

for some functions R_1 and R_2. Note that β-VAE fits this equation with $\lambda = \beta - 1$ and λ'. All the models discussed are is formulated using this objective, as illustrated in Table 5.2.

4.4 Evaluating disentanglement

The fact that a proper definition of disentanglement is generally not commonly agreed upon beyond an intuition is mirrored in the construction of evaluation metrics. Hence there exists a range of different metrics that are constructed to reflect the notion of disentanglement. Most prior works perform some qualitative analysis by visualizing the generated samples when traversing the latent representation. Furthermore, it is common practice to benchmark models on synthetic datasets where the generative factors are known. These factors can then be used to create supervised metrics, as opposed to applying models on arbitrary datasets where the ground truth is unknown. In these cases, we typically have to rely on visual inspections. One could also use the objective functions itself, the reconstruction error, and total correlation as unsupervised evaluation. However, this does not generalize across datasets and architectures for model comparison. A proper metric is necessary to compare different models robustly across model design, hyperparameters, and random seeds. In the following, we will describe three of the most commonly used metrics. Note that all of these metrics assume that the generative factors are known. Later, in Chapter 5, we will empirically compare the evaluation metrics and inspect their ability to generalize across models and randomness. Furthermore, we will compare how well the unsupervised evaluation methods compare with metrics that use the ground truth.

β-VAE metric

Along with β-VAE, Higgins et al. [2019] also proposed a metric. This metric is based on the assumption of a change in one latent dimension should correspond to a change in one generative factor. The metric is the error rate of a simple linear classifier, and the procedure is as following.

1. Choose a generative factor k and generate a batch pairs with factor k as fixed within each pair while the other coordinates is randomly chosen. In this step, the underlying factors are being used. Suppose the batch is of size L and denote the pair by $\mathbf{x}^{(l)}$ and $\mathbf{x}'^{(l)}$.
2. Obtain the representation of the generated data using the variational posterior, $q_{\phi}(\mathbf{z}|\mathbf{x})$, yielding $\mathbf{z}^{(l)}$ and $\mathbf{x}'^{(z)}$ across the batch.
3. For each pair of points, compute the mean absolute difference, $\mathbf{z}_{\text{diff}} = \frac{1}{L}\sum_{l=1}^{L}|\mathbf{z}^{(l)} - \mathbf{z}'^{(l)}|$.
4. Each $(\mathbf{z}_{\text{diff}}, k)$ will then be a training input for a linear classifier that predicts which generative factor k that was fixed. The accuracy is then reported as the disentanglement metric.

If the representation were disentangled, $\mathbf{z}_{\text{diff},k}$ would be zero, and the linear classifier would map the zero value to k.

Factor VAE metric

FactorVAE [Kim and Mnih, 2018] points out some weaknesses of the β-VAE metric, and proposes its own. When proposed, the linear classifier in the β-VAE metric was justified by being simple, and it cannot disentangle itself. However, it is dependent on hyperparameters and optimization choices and is therefore not necessarily robust across different settings. The metric in FactorVAE is similar but relies on classification using a majority vote.

1. Generate a batch of data of size L using a fixed factor k while the others are chosen uniformly at random.
2. Obtain the representation $\mathbf{z}^{(l)}$ across the batch of the generated data using the variational posterior, $q_{\phi}(\mathbf{z}|\mathbf{x})$, yielding $\mathbf{z}^{(l)}$. Normalize the representation using the empirical standard deviation across the dataset such that the metric becomes invariant to scaling.
3. Take the empirical variance of the normalized latent representation in each coordinate. Compute the index with the lowest variance across the batch, $d^* = \text{argmin}_d \mathbb{V}[\mathbf{z}_d^{(l)}]$.
4. Each (d^*, k) will then be a training input for a majority vote classifier that predicts which generative factor k that was fixed. The accuracy is then reported as the disentanglement metric.

Mutual Information Gap

Chen et al. [2018] proposed an evaluation metric that computes the mutual information between ground truth factors and the learned representation. High mutual information will imply that the latent variable contains much information about the underlying generative factor. One single underlying factor can have high mutual information with several coordinates in the latent variable. The proposed metric, Mutual Information Gap (MIG), then considers the two coordinates with the highest mutual information. It is defined as the average and normalized difference in the mutual information for those two coordinates. Let v_k denote a ground truth factor. The MIG is then computed as

$$\mathrm{MIG} = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{H(z_k)}\left(I\left(v_{j_k}, z_k\right) - \max_{j\neq j_k} I\left(v_j, z_k\right)\right),$$

where $j_k = \mathrm{argmax}_j I\left(v_j, z_k\right)$. Since $I\left(v_j, z_k\right) \leq H(v_k)$, the metric is bounded within 0 and 1.

To compute $I\left(v_{j_k}, z_k\right)$, we rely on the joint distribution $q_{\boldsymbol{\phi}}(z_k, v_j) = \sum_{j=1}^{J} p(v_j)p(\mathbf{x}|v_j)q_{\boldsymbol{\phi}}(z_k|\mathbf{x})$. Here $q_{\boldsymbol{\phi}}(z_k|\mathbf{x})$ is the variational posterior, $p(\mathbf{x}|v_j)$ is the known generative process, and $p(v_j)$ is a distribution over the underlying factors. We use the empirical distribution for $p(v_j)$ and $p(\mathbf{x}|v_j)$ which are known. The empirical mutual information is then,

$$I\left(z_j, v_k\right) = \mathbb{E}_{q_{\boldsymbol{\phi}}(z_j, v_k)}\left[\log \sum_{\mathbf{x}\in\mathcal{X}_{v_k}} q\left(z_j|\mathbf{x}\right)p\left(\mathbf{x}|v_k\right)\right] + H\left(z_j\right),$$

where $\mathcal{X}_{v_k}$ is the support of $p\left(\mathbf{x}|v_k\right)$ and $H\left(z_j\right)$ is the entropy of the j-th component of the latent variable $\mathbf{z}$.

Chapter 5

Experiments

In the previous sections, we have explored Variational Autoencoders and its applications in disentangled representation learning in a theoretical manner. Now, we will shift focus towards the implementation of models and experimental results. This section aims to identify and showcase the capabilities and limitations of the models we have considered. We study β-VAE, FactorVAE, β-TCAE and DIP-VAE empirically, and provide a thorough comparison of the models. We emphasize the importance of evaluation metrics and provide an analysis of the β-VAE metric, the FactorVAE metric, and MIG. Lastly, using simple experimental setups, we challenge some common assumptions about the usefulness of disentangled representations. All the code needed to reproduce the following results can be found using the link, `https://github.com/larsmus/master-code`. Throughout all of the experiments, we will use a dataset consisting of images of 2D shapes called *dSprites* [Matthey et al., 2017].

5.1 Introduction

We start the experiments by reproducing results of VAE to showcase its abilities to generate and reconstruct images (Section 5.2). These experiments are included to built intuition of the following. Then, we consider β-VAE and inspect its ability to infer a disentangled latent space (Section 5.3). Continuing on the same path, we provide a comprehensive comparison of β-VAE, FactorVAE, β-TCVAE, and DIP-VAE (Section 5.4). After that, we devote the rest of the experiments to investigate three common assumption about disentanglement models.

First, in Section 5.5, we consider the problem of evaluating disentanglement. Model selection is crucial in order to deploy models in a practical setting. As we have stated in Section 4.4, evaluation of disentanglement is still an open-ended question alongside a proper definition. To test the applicability of the evaluation metrics most commonly used, we will test the agreement between the metrics and the robustness to random seeds.

A property of a disentangled representation that we often alluded to, but seldom showcased, is the *usefulness* of the representation to various downstream inference tasks. Intuitively, a representation that contains most of the relevant information of the data and isolates the factors of variation seems like a desired property with high potential and a range of applications. We will highlight two axes of applications where a disentangled representation could be useful, namely downstream tasks and generalization across domains (Section 5.6).

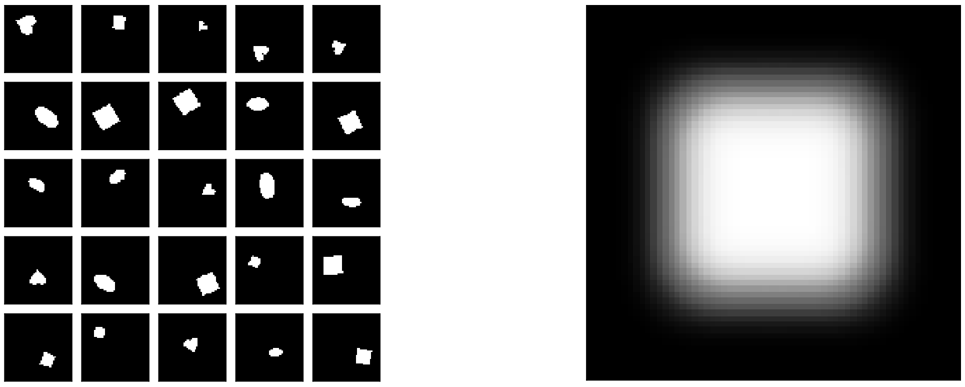
The *dSprites* dataset

The dataset is synthetically generated for disentangled representation learning, introduced alongside β-VAE. As the purpose is to learn a disentangled representation, the dataset originates from a set of known and independent factors of variation. The aim of the models is then to recover these factors. Since the ground truth is known, these can be used in evaluation metrics designed for this purpose, as described in Section 4.4.

Each point in the dataset is an image of 64×64 resolution, which is generated from six independent generative factors.

- **Color.** The 2D shape is white on a black background. In the image, each pixel have either a value of 1 or 0.
- **Shape.** This is a categorical value taking on one of three values: square, heart or ellipse.
- **Scale.** The scale, or size, of the shape can take 6 linearly spaced values.
- **Orientation.** 40 linerarly spaced values on the interval $[0, 2\pi]$.
- **X-position.** 32 linearly spaced values on the interval $[0, 1]$.
- **Y-position.** 32 linearly spaced values on the interval $[0, 1]$.

Each configuration of generative factors is present in the dataset exactly once, yielding $3 \times 6 \times 40 \times 32 \times 32 = 737280$ images. Figure 5.1a shows a subset of 25 uniformly sampled points from the dataset. Notice that the shapes do not go all the way out to the edges, also seen in a density plot for the entire dataset, in Figure 5.1b.



(a) Data samples (b) Data density

Figure 5.1: The dSprites dataset

Note that the dataset encompasses all possible configurations, meaning that the dataset fully describes the data generating distribution. The aim is then to optimize an objective on the known data distribution. During training, we only observe samples $\mathbf{x}$ from the dataset. As these are i.i.d. samples from the ground truth, we do not need a separate test set. This way, we can remove complexity by not having to deal with overfitting. During the evaluation, we can use the data generating process to compare with the learned representation.

Computational resources

We performed the majority of the computations on the NTNU IDUN computing cluster Själander et al. [2019]. The cluster has more than 70 nodes and 90 GPGPUs (General Purpose Graphical Processing Unit). Each node contains two Intel Xeon cores, at least 128 GB of main memory, and is connected to an Infiniband network. Half of the nodes are equipped with two or more Nvidia Tesla P100 or V100 GPGPUs. Idun's storage is provided by two storage arrays and a Lustre parallel distributed file system.

All the training of the models is done on GPU to leverage a data-parallel architecture for efficient training. The access to GPU has been a necessary condition for being able to perform experiments that otherwise would be impossible due to a shortage of memory, and the training time would simply be too long. The models were implemented in PyTorch [Steiner et al., 2019]. Reproducing all the models used in the following experiments requires approximately 24.6 days of GPU usage. Note, however, that the IDUN setup allows for training multiple models in parallel.

Model Setup

As we want to compare several models with different objective functions, other sources of variability are kept constant. This includes hyperparameters, optimizer, the architecture of the neural network models, and modeling distributions. These choices pose an additional inductive bias, and by keeping them constant, we isolate the effect we want to measure, which is the objective. We choose the model setup to be consistent with prior work on disentangled representation learning using VAE [Higgins et al., 2019, Kim and Mnih, 2018, Chen et al., 2018, Kumar et al., 2017] to ensure a fair comparison. Where the choices differ from the original works, we have tried to find a middle ground across models.

We first consider the hyperparameters used throughout, shown in Table 5.1. The batch size, meaning the number of data points used in a single gradient step, is 64. With 737 280 data points, this amounts to 742 iterations per epoch, with 30 epochs being 22 260 training iterations. The dimension of the latent variable is 10. We use the Adam optimizer [Kingma and Ba, 2015]. Adam is an adaptive gradient method, where it adapts the learning rate according to an exponential moving average of the first and second moments of the gradient. The parameters β_1 and β_2 controls the decay of each moment. Most hyperparameters are set to be common choices, and we have not performed any further tuning in the context of these experiments.

Parameter	**Value**
Batch size	64
Training epochs	30
Latent space dimension	10
Optimizer	Adam
Adam, β_1	0.9
Adam, β_1	0.999
Adam, learning rate	0.0001

Table 5.1: Common hyperparameter for all the considered models.

Both the variational distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ and the generative model $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ are parameterized by convolutional neural networks, where $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ denotes the parameters in the network. Both networks, called the encoder and decoder, are described in Table 5.2. The encoder inputs an image and runs it through four convolutional layers and three linear fully connected layers. Each step used ReLU as the activation function. We use an isotropic Gaussian encoder, meaning that given a data point $\mathbf{x}$ the distribution of the latent variable $\mathbf{z}$ is $\mathrm{N}(\boldsymbol{\mu}(\mathbf{x}), \boldsymbol{\sigma}^2(\mathbf{x})I)$. The dimension of the output of the encoder is twice the latent dimension as the first half is taken to be $(\boldsymbol{\mu}(\mathbf{x})$ and the second is $\boldsymbol{\sigma}^2(\mathbf{x})$. Using reparametrization, the latent variable is then $\mathbf{z} = \boldsymbol{\mu}(\mathbf{x}) + \boldsymbol{\epsilon}\boldsymbol{\sigma}(\mathbf{x})$.

Encoder	**Decoder**
Input: $\mathbb{R}^{64\times64}$	Input: $\mathbb{R}^{10}$
4×4 convolutional kernel, stride 2, 32 ReLU	Linear fully connected , 10 ReLU
4×4 convolutional kernel, stride 2, 32 ReLU	Linear fully connected , 256 ReLU
4×4 convolutional kernel, stride 2, 32 ReLU	Linear fully connected , 512 ReLU
4×4 convolutional kernel, stride 2, 32	4×4 convolutional kernel, stride 2, 32 ReLU
Linear fully connected ,512 ReLU	4×4 convolutional kernel, stride 2, 32 ReLU
Linear fully connected , 256 ReLU	4×4 convolutional kernel, stride 2, 32 ReLU
Linear fully connected , 20 ReLU	4×4 convolutional kernel, stride 2, 64×64 Sigmoid

Table 5.2: Model architecture of the encoder and decoder.

The decoder is essentially the transpose of the encoder, going from three fully connected layers to four convolutional layers. Since each pixel is binary, we model the output as a Bernoulli distribution.

We do that by using the sigmoid activation function to ensure that the output is the probability of a pixel being black. By using the Bernoulli density in the ELBO, we see that the reconstruction term is the binary cross entropy,

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = \log \prod_i p_i^{y_i}(1-p_i)^{1-y_i} = \sum_i y_i \log p_i + (1-y_i)\log(1-p_i),$$

where y_i is the value of a pixel in the data and p_i is the predicted probability by the model.

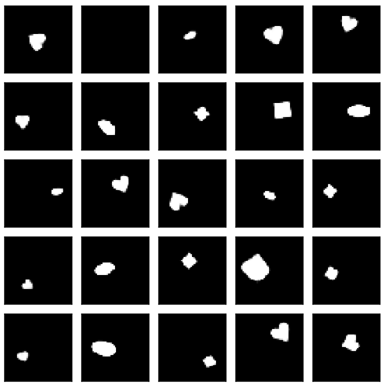
Considering both the encoder and decoder, the model contains 502 005 trainable parameters. These are initialized using the Kaiming method, a standard approach when using ReLU activation functions [He et al., 2015]. Letting n_{in} be the input dimension to a given layers, the Kaiming method initialize the weights and biases by sampling from the distribution $\mathrm{N}\left(0, \sqrt{\frac{2}{(1+a^2)n_{\text{in}}}}\right)$. Here a is the slope of a Leaky ReLU. Since we use a standard ReLU activation function, a is set to be zero.

5.2 Generating images with VAE

We first consider VAE as a generative model, where the aim is to approximate the data generating distribution. Using the learned distribution, we can draw new samples that should resemble the points we observe in the dataset. As opposed to other deep generative models, such as GAN, VAE can do inference in the latent space by utilizing the variational distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, also referred to as the encoder. In the following, we will illustrate both the generative and reconstruction setting of a VAE using the dSprites dataset. Note that the objective here is the standard ELBO, as described in Chapter 3. We will focus on visual inspection in this section, while other evaluation methods are discussed in the following sections.

Image synthesis

We train a standard VAE on the dSprites dataset using the hyperparameters in Table 5.1. To generate new samples, we first sample from the prior on the latent variable $p(\mathbf{z})$. This is set to be standard normal $\mathrm{N}(\mathbf{0}, I)$. Then these samples are passed through the decoder, which parameterizes the conditional distribution $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$. Effectively, this is an ancestral sampling from the joint distribution $p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) = p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p(\mathbf{z})$. Note that the marginal of this distribution is not the data distribution $p^*(\mathbf{x})$, but a approximate distribution $p_{\boldsymbol{\theta}}(\mathbf{x})$. A subset of 25 samples are shown in Figure 5.2a. Furthermore, we compute the mean over 10 000 samples and visualize the density in Figure 5.2b.


(a) Model samples

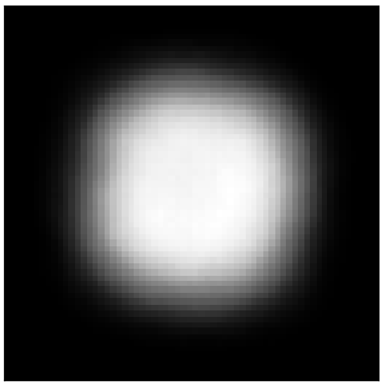

(b) Model density

Figure 5.2: Visualization of the learned distribution of the data.

Comparing to the true data samples in Figure 5.1, the learned model seems to approximate the data reasonably well. Most of the samples in Figure 5.2a are what we can expect to find in the dataset. Note, however, that there is one sample that does not contain any shape. The density of the samples generated by the model in Figure 5.2b also looks convincing compared to the true density 5.1b.

We should note that the data we are using here are relatively simple compared to typically used benchmarks for deep generative models. The binary form of the dSprites data allows us to use a Bernoulli distribution as each pixel in the sample is either black or white. By default, it is easy to perceive the generated shapes. Typically, deep generative models use datasets containing natural images to test the sample quality. For instance, this is often a set of images of faces, as shown previously in Figure 2.3. It is a known problem with VAE that it tends to generate blurry images in this setting [Kingma and Welling, 2013, Zhao et al., 2017b]. A way to remedy this is modeling with more flexible posteriors, as discussed in Section 3.4. The state-of-the-art in high-fidelity image synthesis is as of today GAN-based models [Karras et al., 2018], shown in Figure 2.3. We refer to the original work of VAE by Kingma and Welling [2013] for samples from other datasets.

Reconstruction the data

An advantage of amortized inference in VAE is that we can infer the latent variable given a data point. We can then either do inference in the latent space or reconstruct the data using the decoder. In comparison, a regular GAN will essentially transform random noise into a sample to resemble the data. In a VAE, the reconstruction path is first to obtain a latent using variational distribution $\mathbf{z}' \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, before generating a reconstructed sample using $\mathbf{x}' \sim p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}')$. As discussed in Section 3.3, this is similar to an autoencoder, where the two distribution takes on the role of an encoder and decoder, respectively.

Figure 5.3 shows the ability of VAE to reconstruct the input data. The first row displays 10 data points chosen uniformly from the dataset. These are then fed through the model to obtain the reconstructed data in the second row. We see that there is a high resemblance, which we also see in the difference in the third row. The difference shows clearly that there are only pixels along the edges of the shapes that are the wrong color.

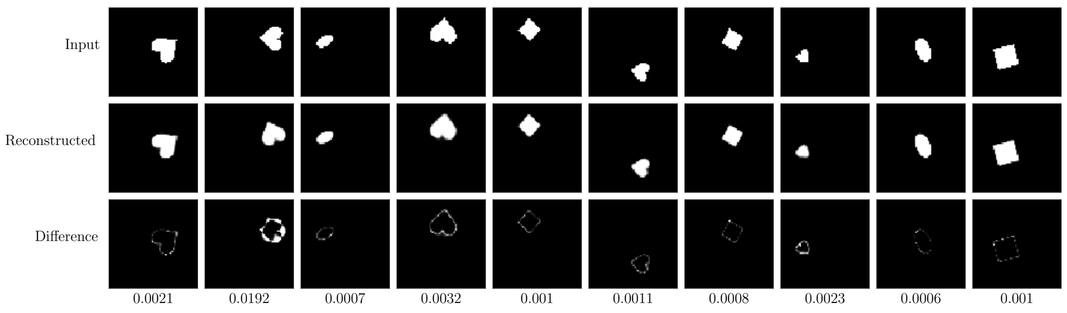


Figure 5.3: Reconstructed images using VAE

We can also evaluate the models ability to reconstruct the data using the first term in the ELBO, $\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]$. A lower value indicates a better reconstruction bounded from below by 0. Figure 5.4 shows the recorded reconstruction during training. The model seems to converge after about 20 epochs and ends with a reconstruction error or 0.274, which tells us that the model can reconstruct the data well. Next, we will see how models that enforce a disentangled structure affects the ability to reconstruct.

5.3 Inferring the latent space in β-VAE

We now consider β-VAE, where we take into account an additional penalizing parameter β in the ELBO. A larger β will enforce a structure on the variational distribution similar to the standard normal prior, allowing for the emergence of a disentangled behavior on the learned representation. In this section, we will verify the theoretical analysis in Section 4.2. This includes inspecting the learned representation and the practical ability to disentangle. Furthermore, we will verify how better disentanglement in β-VAE comes at the cost of the reconstruction error.

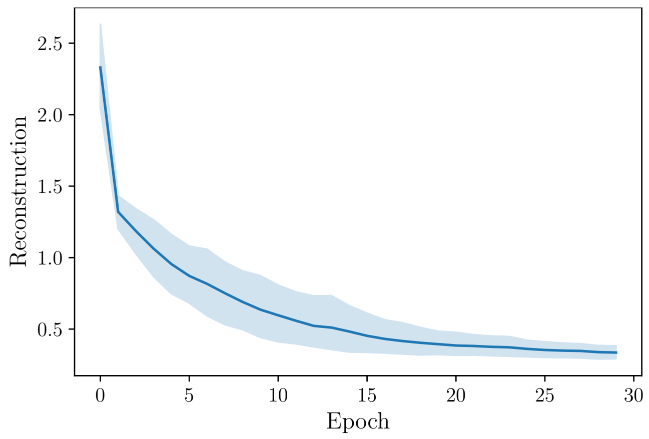


Figure 5.4: Reconstruction error for VAE using 10 random seeds.

Traversing the latent space

An easy way to visually inspect the representation and the ability to disentangle is to traverse the latent space in each coordinate. Specifically, we sample an arbitrary image from the dataset and obtain the representation in the latent space using the encoder. Then we vary each coordinate on the interval $[-3, 3]$ while keeping the rest fixed. The variational distribution is similar to the standard normal, so that specific interval covers most of the mass. Using ten equally spaced points on the interval, we obtain ten different representations for each coordinate. As before, the dimension of the latent space is 10. Then the decoder maps all the representation back to a generated image. Figure 5.5 shows the resulting latent traversal.

Each row corresponds to one coordinate in the latent space, showing the traversal in the horizontal direction. Recall from 3.5 that the KL divergence $D_{\mathrm{KL}}(q_\phi(\mathbf{z}\|\mathbf{x})|p(\mathbf{z}))$ has a closed-form expression that decomposes a sum over each coordinate when both distributions are normal. We sort the rows using the dimension-wise KL divergence, meaning that the coordinate carrying most information is on the top.

The first row in Figure 5.5 shows movement in the horizontal direction. The shape starts on the right side in the leftmost image and moves over to the right as we traverse the latent coordinate. Note that all other factors stay constant, meaning that the shape itself does not change. This indicates that the generative factor "x-position" has been isolated, or disentangled, from the other factors. Similarly, the second row shows clearly movement in the vertical plane while keeping other factors constant. Hence, the model seems to show the desired disentangled propriety for both the y- and x-position.

The third row appears to capture the rotation while not keeping the shape constant across the traversal. Likewise, the fourth row scales the shape, but also here the shape is not constant. The fifth row transition between three shapes, while keeping the rotation, scale, and position constant. All of these rows are examples of where the coordinate is informative and, to some degree, interpretable, but the representation is not perfectly disentangled. By that, we mean that each generative factor is not isolated, and changing a single coordinate can change multiple factors. Note that it is the shape that influences the traversals in these rows. The shape is a categorical variable, which is hard to model as a continuous latent variable by design. Nevertheless, given the setup, the rotation and scale should ideally be isolated from the shape.

Disentanglement metrics

While the latent traversal gives a visual intuition of how the model performs, it is not reliable for comparison and model selection. To investigate how β affects the disentanglement, we implement the evaluation metrics described in Section 4.4. Note that we, in this context, refer to β-VAE and FactorVAE as the respective metrics proposed alongside the models. Since we train each configuration across ten different random seeds, we report the mean and standard deviation of each metric.

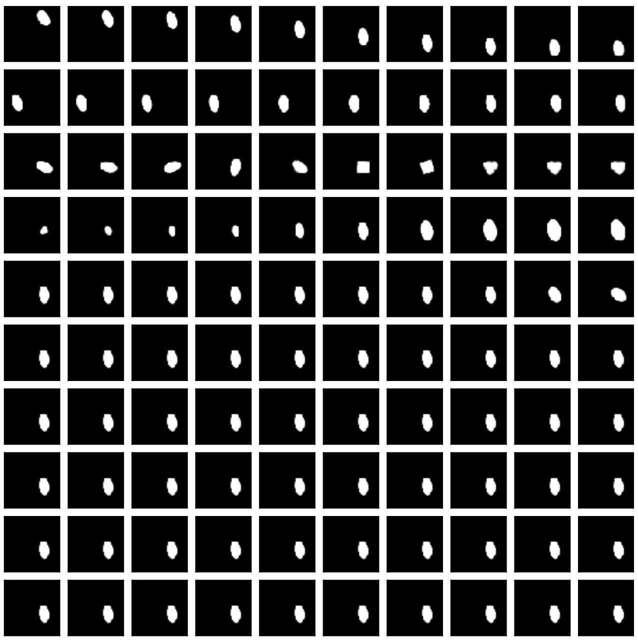


Figure 5.5: Latent traversal for β-VAE with β=3

The β-VAE metric and the FactorVAE metric use a prediction model to infer the underlying model. For both of these, we use a batch size of 64 to compute each training feature, as described in Section 4.4. Furthermore, we use 10 000 points in the training set, and 5000 in the evaluation set. The metric is the accuracy of the evaluation set. For the MIG metric, we determine the mutual information by binning each coordinate of the obtained representation from 10 000 points into 20 bins.

Table 5.3 displays the evaluation metrics for various β, where the entry with the highest mean is highlighted for each metric. Note that the highest mean metric is obtained for different β. However, a substantial standard deviation makes it difficult to make a final selection. We note that $\beta = 3$ achieves a consistently high metric for all the metrics. Later in Section 5.5, we will explore the robustness of the metrics in terms of hyperparameters seeds and the overall agreement between the metrics.

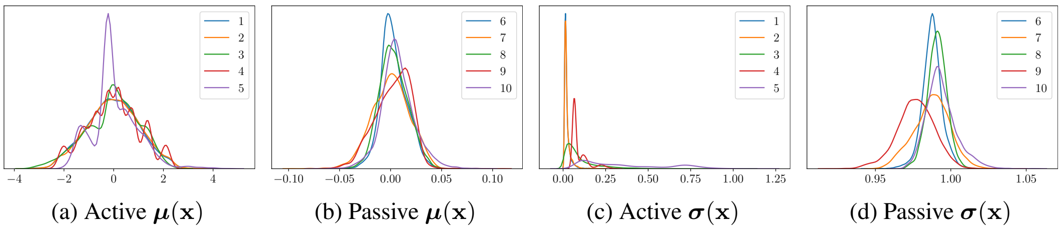
Parameter	**β-VAE**	**FactorVAE**	**MIG**
$\beta = 1$	**0.867 ± 0.079**	0.696 ± 0.065	0.201 ± 0.07
$\beta = 3$	0.860 ± 0.055	**0.737 ± 0.031**	0.269 ± 0.028
$\beta = 10$	0.762 ± 0.038	0.666 ± 0.018	**0.270 ± 0.013**
$\beta = 30$	0.729 ± 0.006	0.658 ± 0.014	0.269 ± 0.013

Table 5.3: Disentanglement metrics for β-VAE

Posterior collapse

The last five rows in Figure 5.5 all contain images that are more or less the same, and judging from the images, the representation does not contain any useful information. We hypothesize that this is because the variational posterior matches the prior. Stronger control of the KL divergence term in β-VAE restricts this bottleneck further by bringing the KL divergence towards zero. In this case, we can't differentiate samples from $q_\phi(\mathbf{z}|\mathbf{x})$ conditioned on different $\mathbf{x}$, yielding no useful information. Ideally, we want a non-zero KL divergence, while retaining a small reconstruction error.

To verify this claim, we inspect the mean and standard deviation of the variational posterior. We take a subset of the dataset and obtain the latent representation using the encoder. Then we separate the coordinates into an active and a passive group based on the information contained in the coordinate sorted by the dimension-wise KL divergence. Figure 5.6 shows kernel density plots of the encoded mean $\boldsymbol{\mu}(\mathbf{x})$ and standard deviation $\boldsymbol{\sigma}(\mathbf{x})$ separated by the two groups. The indices refer to the information in the coordinates, where 1-5 refer to the first five rows in 5.5 and 6-10 to the last five rows. We see in Figure 5.6a and 5.6c that the active components have all have a mean with significant spread around 0 and a relatively small standard deviation. The passive components in Figure 5.6b and 5.6d have a mean close to 0 and standard deviation close to 1. This confirms that the passive component in the latent traversal in Figure 5.5 has a variational posterior that collapses to the prior in the respective coordinates.



(a) Active $\boldsymbol{\mu}(\mathbf{x})$ (b) Passive $\boldsymbol{\mu}(\mathbf{x})$ (c) Active $\boldsymbol{\sigma}(\mathbf{x})$ (d) Passive $\boldsymbol{\sigma}(\mathbf{x})$

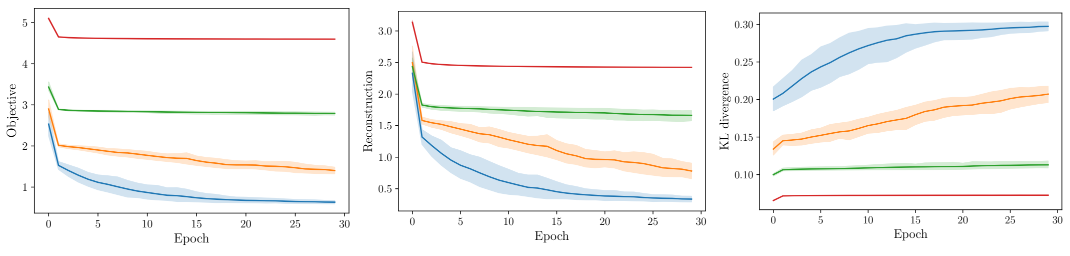
Figure 5.6: Variational mean and standard deviation for a subset of 10 000 data points. The latent coordinates are separated into an active and passive group, depending if the coordinate carry useful information.

This phenomenon is known as *posterior collapse* [Bowman et al., 2015, Oord et al., 2017]. A simple way of dealing with this is using an annealing scheme on the KL regularizer. The weight of the KL divergence term is then gradually increased from 0 to β. At the start of training, the model learns to encode as much information as possible using the reconstruction term. Then gradually, the model will impose a structure of the latent representation by matching the prior. We experimented with different annealing schedules but found it difficult to gain any improvements. Given our dataset and the number of generative factors, we do not expect each latent coordinate to be informative.

Trade-off between reconstruction and disentanglement

Recall from Equation 4.3 that the KL divergence between then variational posterior and the prior decomposes as the sum of the mutual information between $\mathbf{x}$ and $\mathbf{z}$, and the aggregate posterior. A larger $\beta > 1$ penalizes both these terms, meaning that the latent bottleneck described by the mutual information will be tighter. Less information about $\mathbf{x}$ can then be expressed through $\mathbf{z}$. However, a penalty on the aggregate posterior will lead to structured variational distribution across the dataset, which encourages better disentanglement. Penalizing both terms naturally leads to a trade-off between the reconstruction error and the ability to disentangle.

To inspect this trade-off, we first decompose the objective function for different β in Figure 5.7. Figure 5.7b shows the reconstruction error $\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]$, and Figure 5.7c is the KL divergence $D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$. Figure 5.7a is the sum of the two, where the KL divergence is weighted with β. We first note that the reconstruction error gets consistently higher for larger β. Furthermore, for larger β the KL divergence becomes constant. This is because the variational posterior matches the prior. Samples in the latent space will then be hard to differentiate, leading to a worse reconstruction error.



(a) Objective (b) Reconstruction (c) KL divergence

Figure 5.7: Decomposition of objective function for β-VAE

To further inspect disentanglement for larger β, we produce latent traversals for $\beta = 30$ in Figure 5.8. Here we display only the coordinates that carry any information. We see that three coordinates convey the position and scale of the shape. However, the shape itself is manifested by a circle, even though no circles are present in the dataset. It is more likely that the circle is an average of all the possible shapes and that reconstructing any image in the dataset would output a similar image. Therefore, the model is unable to accurately reconstruct the input, while it is still able to represent a somewhat disentangled behavior.

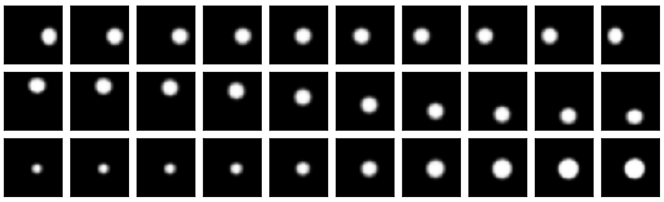


Figure 5.8: Latent traversal for β-VAE with β=30

In Figure 5.9 we plot the reconstruction error against the β-VAE metric for different regularization strength β using 10 different random seeds. We want a small reconstruction error and a disentanglement metric close to 1. We see that the reconstruction error increases for large β. Furthermore, for $\beta = 30$, the reconstruction nearly collapses to one point for different random seeds. That aligns with Figure 5.8, where the posterior collapses and the generated samples are difficult to differentiate. Note that the variability in the disentanglement metric decreases for larger β, which we also see in Table 5.3. This underpins the claim that β continuously constricts the information bottleneck in the latent variable. With higher β, the latent variables contain less information, yielding similar-looking samples with less variance.

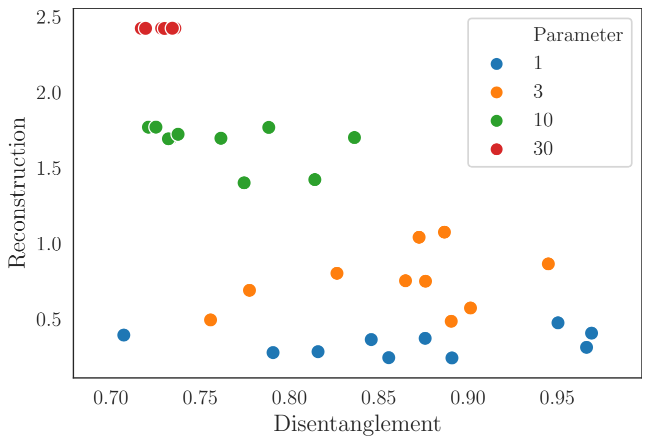


Figure 5.9: Reconstruction and disentanglement (β-VAE metric) for β-VAE.

5.4 Comparison of disentanglement models

In this section, we will concentrate on the models discussed in Section 4.3, which specifically pose some regularization on the aggregates posterior, namely FactorVAE, β-TCVAE and DIP-VAE. Motivated by the decomposition in Equation 4.3, all of these models aim to enforce a global structure on the variational approximation. FactorVAE and β-TCVAE utilize the total correlation to regularize the independence in the aggregate posterior. DIP-VAE takes a more direct approach by decorrelating the distribution using the second moment of the latent variable. We will see how these models perform compared to β-VAE, and if they can retain the ability to reconstruct the data while also enforcing disentanglement.

Table 5.4 shows the regularization parameters we have used. We selected these to cover different behavior of the model but was not extensively tuned. Note that for DIP-VAE, we used the same regularization for both the diagonal and off-diagonal terms.

Hyperparameter	**Value**
β-VAE	$\beta \in \{1, 3, 10, 30\}$
FactorVAE	$\lambda \in \{5, 10, 25, 50, 100\}$
β-TCVAE	$\beta \in \{1, 3, 6, 10\}$
DIP-VAE	$\lambda \in \{1, 3, 10, 30\}, \lambda' = \lambda$

Table 5.4: Regularization strengths for each model.

Discriminator
Linear fully connected, 1000 Leaky ReLU
Linear fully connected, 1000 Leaky ReLU
Linear fully connected, 1000 Leaky ReLU
Linear fully connected, 1000 Leaky ReLU
Linear fully connected, 1000 Leaky ReLU
Linear fully connected, 2 Leaky ReLU

(a) Architecture

Hyperparameter	**Value**
Optimizer	Adam
Learning rate	0.0001
β_1	0.5
β_2	0.9
Batch size	64

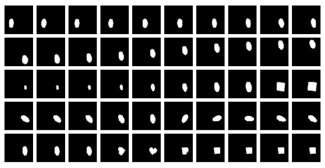
(b) Hyperparameters

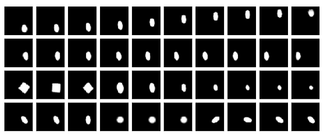
Table 5.5: Discriminator in FactorVAE and hyperparameters.

To estimate the total correlation $D_{\mathrm{KL}}(q_\phi(\mathbf{z}) \| \bar{q}_\phi(\mathbf{z}))$, FactorVAE use a discriminator to classify samples from each distribution. We implement the discriminator as a fully connected neural network. Table 5.5 shows the full architecture and hyperparameters used.

Traversing the latent space

First, we compare the latent traversals for different models. Figure 5.10 displays the traversal for the active components in the latent space. In this case, we chose the hyperparameters on a visual basis. From Figure 5.10a, we see that FactorVAE faithfully produces traversals in line with what we expect. It has five active components, each with an interpretation of the underlying generative factors. Visually, the results are comparable to β-VAE. β-TCVAE similarly produces valid latent traversals, although only with four active components. For DIP-VAE, the model is not able to generate faithful samples. We see from the traversal that the shape is often misrepresented, and several squares are completely black. This behavior also exists on the original paper [Kumar et al., 2017] to some degree. We suspect the performance could be improved by additional tuning. However, we want to fairly compare the different models on the same set of hyperparameters besides the regularization, and therefore we do not perform fine-tuning.


(a) FactorVAE, $\lambda = 5$


(b) β-TCVAE, $\beta = 3$

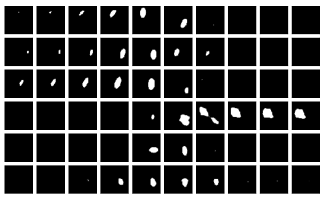
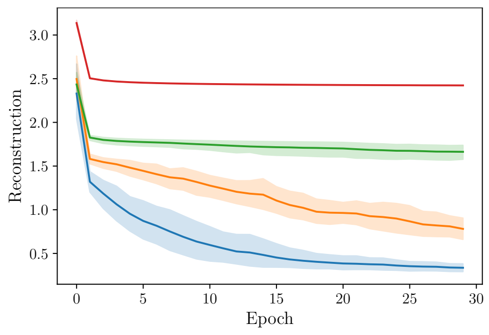

(c) DIP-VAE, $\lambda = \lambda' = 3$

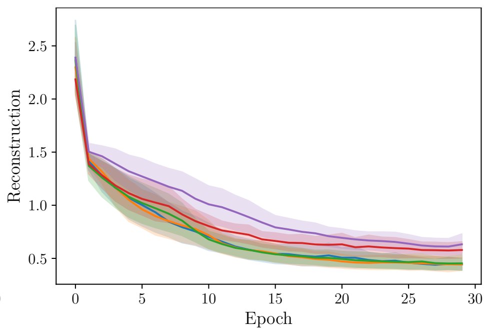
Figure 5.10: Latent traversal for different models.

Reconstruction error

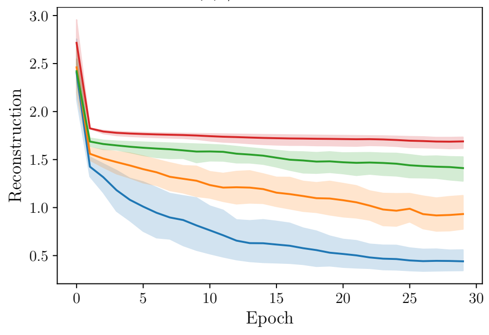
To address the trade-off between the disentanglement and reconstruction, we plot the reconstruction error for the different models in Figure 5.11. Most notably, we see that FactorVAE obtains a low reconstruction error for a range of λ. The model can reconstruct the data while enforcing a larger penalty on the total correlation. DIP-VAE also displays an ability to reconstruct the data across the parameters. However, as we say in Figure 5.10, the same model struggled with traversing the latent space. β-TCVAE displays similar behavior to β-VAE as the reconstructions error worsens with a larger regularization strength. Note that FactorVAE and β-TCVAE estimate the same objective, but with two different estimators of the total correlation. However, FactorVAE displays a superior ability in terms of reconstruction. The Monte Carlo estimator used in β-TCVAE is biased, and we also observed the estimate to be negative. As the KL divergence should be non-negative, β-TCVAE seems to give an insufficient estimate.



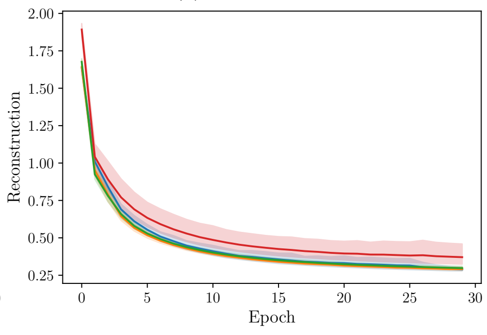
(a) β-VAE



(b) FactorVAE



(c) β-TCVAE



(d) DIP-VAE

Figure 5.11: Reconstruction loss across disentanglement models.

Disentanglement metrics

We evaluate all the models on every metric and display the results in Table 5.6. Note that we only include the metric for the best performing regularization strength, as specified by the parenthesis. Since we train each model using ten different random seeds, we report the mean and standard deviation of the metric.

For all metrics, FactorVAE outperforms the other models ranked by the mean. However, both β-VAE and β-TCVAE are within one standard deviation from FactorVAE on all metrics. DIP-VAE is consistently worse, which confirms the behavior observed in Figure 5.10c.

Inspecting the aggregate posterior

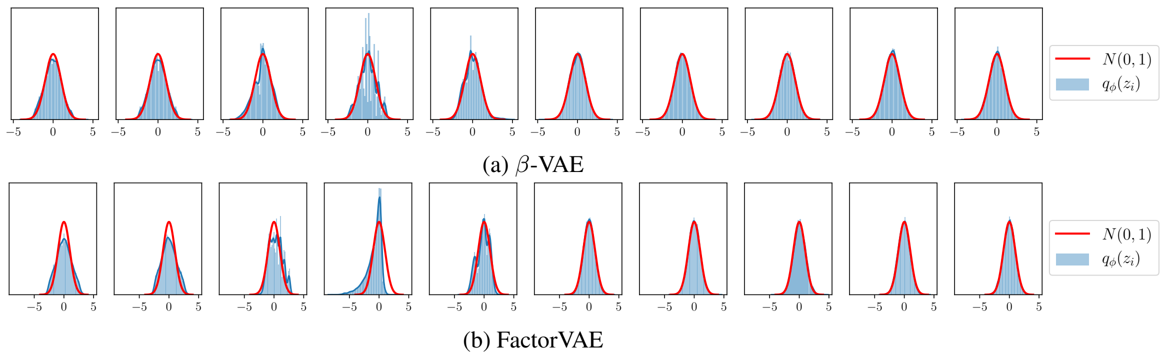
We cab further inspect the aggregated posterior to exemplify the role of the regularizer. We consider both β-VAE and FactorVAE and plot the aggregated posterior in Figure 5.12. We sample by first selecting data points uniformly from the dataset, then we draw samples from $q_{\phi}(\mathbf{z}|\mathbf{x})$. We draw 10 000 samples and plot the marginal distribution for each coordinate. The columns are sorted after the

Model	β-VAE	FactorVAE	MIG
β-VAE	0.867 ± 0.079 (1)	0.737 ± 0.031 (3)	0.270 ± 0.013 (10)
FactorVAE	**0.894** ± **0.033** (10)	**0.758** ± **0.054** (10)	**0.297** ± **0.061** (25)
β-TCVAE	0.884 ± 0.049 (1)	0.736 ± 0.028 (1)	0.285 ± 0.06 (1)
DIP-VAE	0.612 ± 0.041 (1)	0.479 ± 0.038 (1)	0.039 ± 0.011 (1)

Table 5.6: Disentanglement metrics across models and metrics.

average KL divergence for each coordinate left to right. The marginal that deviates the most from the standard normal prior is on the left. Furthermore, we also plot the density of a standard normal distribution for comparison.

From Figure 5.12a we see that the histogram of $q_\phi(z_i)$ matches the standard normal prior in almost all the coordinates z_i. In contrast, FactorVAE displays a greater discrepancy in Figure 5.12b. For the five leftmost plots, the histogram is substantially different from the prior. It is also consistent with the five active components we observed in the latent traversal. The result tells us that FactorVAE can retain information in the variational distribution by focusing only on the coordinate-wise independence of $q_\phi(z_i)$ instead of pushing it towards some specified prior $p(\mathbf{z})$.



(a) β-VAE

(b) FactorVAE

Figure 5.12: Aggregate posterior for β-VAE and FactorVAE.

5.5 Evaluating disentanglement

There is no general agreement in the literature on how to evaluate disentanglement. We have considered three of the commonly used metrics. However, in recent years several other metrics have been proposed [Kumar et al., 2017, Eastwood and Williams, 2018, Ridgeway and Mozer, 2018]. Proper evaluation metrics are necessary for comparing and selecting models. We will investigate the importance of selecting a metric and how much each of the metrics we have considered agree. The evaluation metrics we consider all use some ground truth factor. However, this is generally not available for other than synthetic datasets. We argue that unsupervised evaluation is crucial for applying disentanglement models on real data. To investigate this, we will look at the agreement between the objective function and the disentanglement metrics.

Agreement between metrics

Figure 5.13 shows the Spearman rank correlation between each pair of metrics. We compute the correlation on each parameter selection and a random seed. There is a strong correlation between the three metrics. In particular, the β-VAE and FactorVAE metric seem to capture similar notions. It is not surprising since both similarly construct the metric using a prediction model. Overall, the correlation plot indicates that the metrics we have considered measure the same underlying phenomena.

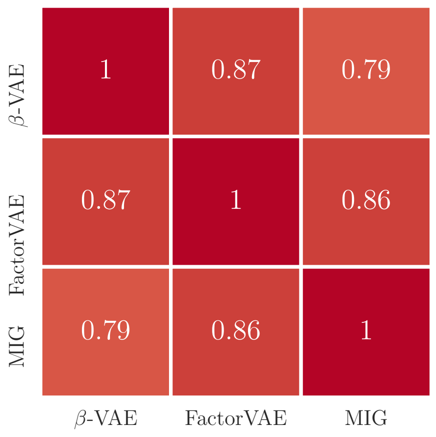
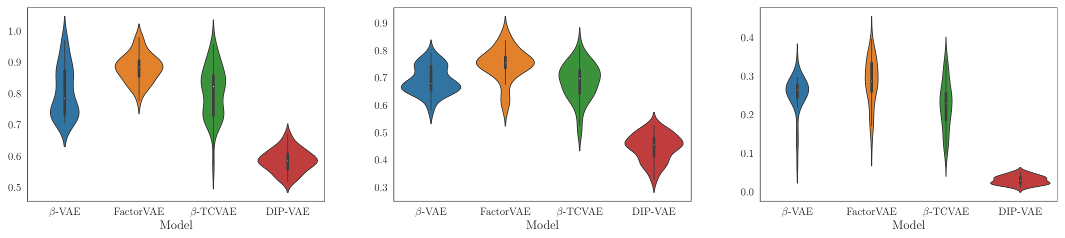


Figure 5.13: Spearman rank correlation between disentanglement metrics.

Robustness to hyperparameters and randomness

In our experiments, the main difference between the models we have considered is the objective function. All other sources of variations are kept constant to have a fair comparison. A natural question is whether the best performance is due to the objective function we choose or other sources of variations such as hyperparameter and randomness. The only hyperparameter that varies is the regularization strength, and the randomness is due to the random seed.

To investigate this, we first plot all the considered distribution metrics for each model in Figure 5.14. The distribution for each model arises from different regularization strengths and random seeds. For each metric, the mean for FactorVAE is better, while β-VAE and β-TCVAE is about the same. However, there is considerable overlap between each model. That means that FactorVAE with a bad selection of seed and regularization strength can perform considerably worse than a good selection for β-VAE. We can then qualitatively conclude that the selection of seed and hyperparameters is substantially more important than which objective function we choose.



(a) β-VAE metric (b) FactorVAE metric (c) MIG

Figure 5.14: Distribution of all disentanglement metrics for each model.

To further isolate how randomness affects the metrics, we consider only FactorVAE and the FactorVAE metric. Figure 5.15 displays the distribution of the disentanglement metric across ten random seeds for different regularization strengths. While there is a considerable difference between the maximum value, the mean is about the same. However, we also see a worrying trend in that random seed has a notable effect on the metric. A model with a regularization strength of 5, attains both the highest and lowest disentanglement metric. The disentanglement scores are highly dependent on randomness and tuning of hyperparameters. It is a challenge in reproducing results for practitioners and creating robust models to apply to real data. Furthermore, the result questions the trustworthiness of the model. It is difficult to tell if the result comes from a superior model or a good random seed. Even though the metrics are designed using ground truth, the dependence of random seeds makes model selection a challenge.

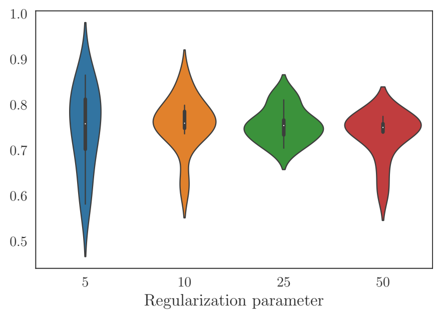
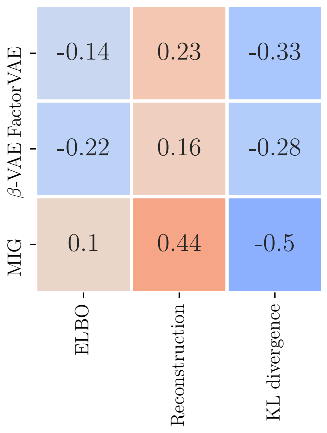


Figure 5.15: Distribution of the FactorVAE metric against the regularization strength in FactorVAE.

Unsupervised model selection

All the metrics we have considered have used ground truth labels to some extent. We can do this since we synthetically generated. However, most of the data available does not have such labels attached. The point of unsupervised representation learning is that there is no access to labels. While synthetic data is practical to benchmark and analyze models, they should generalize to data from the real world. That involves having reliable unsupervised evaluation metrics.

We investigate how the objective function itself corresponds with the disentanglement metrics. We consider the ELBO, as well as the reconstruction and the KL divergence between the variational distribution and the prior separately. We are interested in the agreement between these unsupervised metrics with the supervised disentanglement metrics. Figure 5.16 shows the full table of Spearman rank correlation. From this, we see that there seems to be no substantial correlation pattern between the unsupervised metrics and the disentanglement metrics. Based on this, there is no clear strategy of choosing a model purely based on unsupervised metrics, and this will remains an open problem.



	ELBO	Reconstruction	KL divergence
FactorVAE	-0.14	0.23	-0.33
β-VAE	-0.22	0.16	-0.28
MIG	0.1	0.44	-0.5

Figure 5.16: Spearman rank correlation between disentanglement metrics and unsupervised metrics.

5.6 Usefulness of disentanglement models

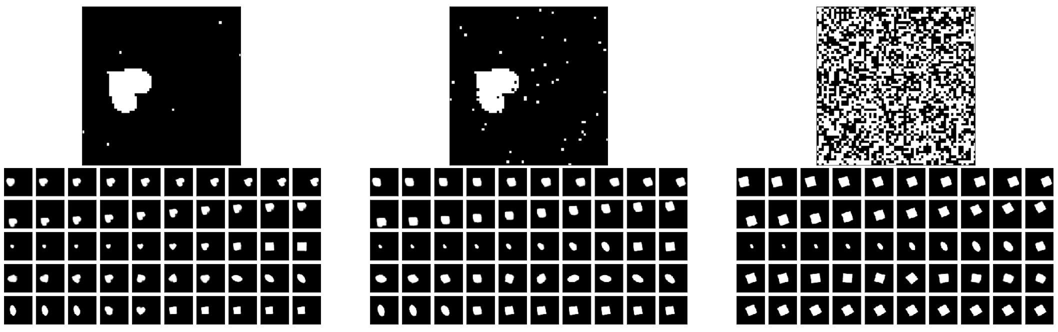
The goal of a disentangled representation is to capture the information about $\mathbf{x}$ in a compact and interpretable structure in the latent variable $\mathbf{z}$. Ultimately, we want the representation to be useful and explainable. A pressing challenge in deep learning is that the model acts as a black-box, making it difficult to explain inferences. Using an interpretable representation combined with simple models may be a step towards an explainable and trustworthy practice. Although this property is often hypothesized [Bengio et al., 2012, Schölkopf, 2019], there is still a lack of practical examples where disentangled representations using unsupervised techniques demonstrate its usefulness. In this section, we will consider some simple experimental setups to investigate this for the models we have considered out of the box. First, we will consider the case when the noise corrupts the input. Then we will investigate if the model can generalize to input outside of the input domain. Lastly, we train a simple prediction model on the representation and consider the performance and sample efficiency on a downstream task.

Domain adaption

Current deep learning models achieve tremendous generalization for a large amount of i.i.d. data, but is lacking in performance for data outside of distribution. Capturing the underlying structure of the data, while disregarding nuisance, may be beneficial for generalizing in settings where the model is evaluated on data outside of the training distributions [Schölkopf, 2019]. For instance, in transfer learning, the aim to estimate a representation using a larger dataset before fine-tuning it to domain-specific data. Another related task is domain generalization, where a model is trained on multiple domains to perform on an unseen domain. Similarly, domain adaption refers to the task in which a model is trained on one domain and evaluated on a different, but a similar domain.

Throughout all of our experiments, we have used a carefully designed dataset aimed at showcasing unsupervised disentanglement. Real data, however, are often messy. Some sources may corrupt the data out of our control. Small changes in the input may easily fool deep neural networks. Adversarial examples are instances of the data with small feature perturbations designed to deceive the model. For instance, a small source of noise can cause an image classification model to change the prediction with high confidence [Goodfellow et al., 2015, Szegedy et al., 2014]. Similarly, we will consider whether we can recover the latent traversals when we add a noise source.

We add noise to an image from the dataset and use FactorVAE to produce the latent traversals. Since each pixel is binary, we change the value for each pixel with a certain probability. The probability is set to be 0.001, 0.01 and 0.05 as displayed as (a-c) in Figure 5.17. When only a small amount of noise is present (Figure 5.18a), the traversal is as expected. The representation captures five active coordinates, and the shape is preserved while the position changes. With increased noise, the traversal does not conserve the shape as the position changes (Figure 5.18b). None of the generated images in the traversal resemble the heart, which is the input. Even when the input only contains noise, the model still produces a valid traversal (Figure 5.17c) similar to Figure 5.18b. It indicates that the latent spaces collapse to a region that produces similar generated samples.



(a) (b) (c)

Figure 5.17: Latent traversal (bottom) where the input image (top) is corrupted with noise.

A similar problem is when the model is evaluated on data that are different but related to the training distribution, which is known as domain adaption. Ideally, we hope that the model can transfer some of the distilled knowledge on to the new domain. For instance, the model can produce a disentangled representation for a shape not seen in the dataset. We consider two different cases of domain adaption and evaluate the property visually using the latent traversal. First, evaluate the model on an image of a triangle. Note that this is a shape that is not included in the dataset. Second, we use an image where the colors are inverted. Both of these experiments are rudimentary, but they are designed to display any transfer of knowledge without modifying the model itself.

Figure 5.18 shows the input images and the generated traversal. Again, we see from the almost identical traversal plots, that the model ignores the input. It is likely due to that the model encodes the unseen input into an uninformative region of the latent space. Interestingly, both the triangle, the inverted image, and the noisy images all generate similar reconstructed images. The fact that the reconstructed images are valid seems to come from the expressiveness of the decoder network, rather than the information in the latent variable. Overall, these results lead us to conclude that the model cannot transfer meaningful knowledge outside of the training distribution without specific modification aimed at this purpose.

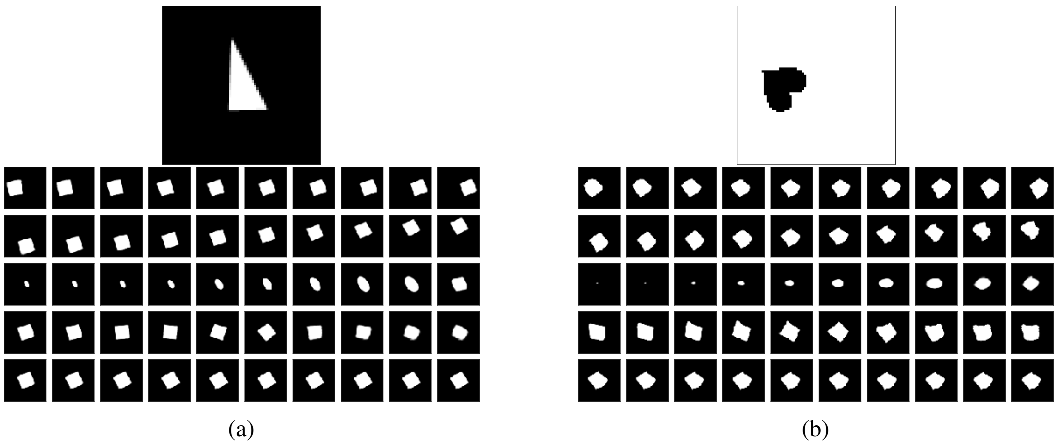

(a) (b)

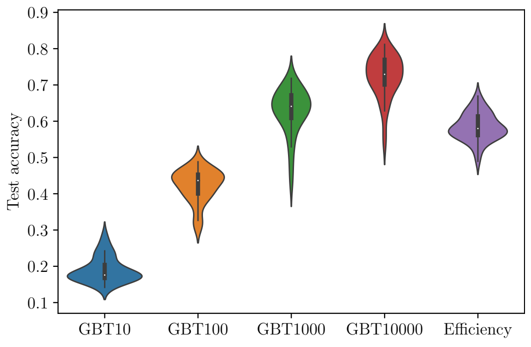
Figure 5.18: Latent traversal for domain adaption.

Downstream inference task

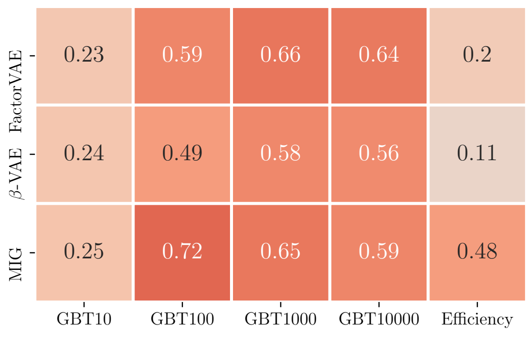
A key motivation of disentangled representation is the usefulness in downstream inference tasks. In particular, it is often argued that such a representation can distill the information in a compact structure such that fewer labeled samples are needed for a downstream task [Schölkopf, 2019]. Let us say we want to use the representation in a supervised task, but there are limited labels. We could hope that the representation captures the factors of variation in an unsupervised manner, such that fewer labels are necessary for the supervised task. The property is also known as the *sample efficiency* of the model. Inspired by the experimental setup by Locatello et al. [2018], we consider a simple downstream classification task where we want to recover the factors of variation using a gradient boosting tree (GBT) [Hastie et al., 2009, Chapter 10] on the representation.

To address the accuracy and sample efficiency, we use training sets of different sizes. From the five ground truth factors, we sample training sets of size 10, 100, 1 000, and 10 000. Then the task of the classification model is to recover the ground truth factors. Note that the possible values for each factor are different, as described previously. To evaluate, we use a test set containing 5 000 samples. We report the mean test accuracy across the five factors. The classification model is a standard gradient boosting tree from the Python package Scikit-learn with the default parameters. We also report the sample complexity by defining a metric as the mean accuracy based on 100 samples divided by the mean accuracy based on 10 000 samples. If the model can use each sample efficiently, this ratio should be close to 1. The downstream task is evaluated for ten random seeds using different regularization strengths on FactorVAE.

Figure 5.19a shoes the rank correlation between the accuracy and the various disentanglement metrics. All the metrics seem to have a certain positive correlation with the downstream performance. Furthermore, Figure 5.19b shows the accuracy of the various training set sizes and sample efficiency. Both the models with 10 and 100 training samples have a lacking performance, while the larges model achieves a mean accuracy of 0.72. Accordingly, the mean efficiency is about 0.58. We argue that for the model to be useful for a downstream task like this one, it should achieve comparable results with few samples. The point of unsupervised learning of disentangled representation is that we do not have access to labels. Otherwise, we could incorporate the labels itself into the training procedure. We usually can label a small set of points manually. If we only have access to 100 labels, we could hope that a disentangled representation could utilize this to archive useful predictions. Unfortunately, this does not seem to be the case for this experimental setup.



(a)



(b)

Figure 5.19: (a) Mean accuracy for different sizes of the training set and sample efficiency. (b) Spearman rank correlation between the downstream performance and various disentanglement metrics.

Chapter 6

Summary and Outlook

Summary

In this thesis, we have presented the *Variational Autoencoder* and motivated its use for unsupervised learning of disentangled representation. The key idea is that model aims to recover a small set of explanatory factors of variation from which the data is generated. We have built up the theory stone by stone, starting from probability theory and deep learning, followed by an information-theoretic perspective of representation learning. We then gave a thorough tutorial of the Variational Autoencoder, a deep generative model that has been highly influential in recent years. We considered the model in light of other deep generative models such as GAN and normalizing flows, and we pointed out directions for improving Variational Autoencoders going forward.

Next, we presented β-VAE and three additional methods that all intend to impose an interpretable and disentangled structure on the latent variable. We analyzed β-VAE using information theory, and we deduced a connection between the objective in β-VAE and the information bottleneck principle. Furthermore, we discussed the challenge of evaluating disentanglement and considered three specific supervised metrics.

Last, we provided implementations of all the models and metrics and performed a variety of experiments to inspect and challenge some common assumptions. We made an extensive empirical comparison of the four different disentanglement models and the three supervised metrics. The results lead us to conclude that there is lacking robustness to different hyperparameters and random seeds and that choosing these parameters can matter more than choosing the model. A common belief is that unsupervised learning of disentangled representation can be useful for domain adaption and downstream inference tasks. We found no such evidence using the models out-of-the-box with simple experimental setups.

Outlook

We should acknowledge some of the limitations of our study and interpret the results with some care. While we have aimed at a fair and experimental study, further testing is required to draw any definite conclusion. First, we have focused on a particular dataset with synthetically generated images. Second, we fixed model architecture to a convolutional network, latent dimension, batch size, and other hyperparameters. A larger study that considers additional model choices and datasets is necessary to conclude on the role of the disentanglement model. Nevertheless, we urge further research to demonstrate the concrete, practical benefits of the notion of disentanglement of the learned representations. Furthermore, unsupervised evaluation and model selection persists as key challenges needed to be solved to be useful for the practitioner.

We also pose a deeper theoretical understanding of why a disentangled representation emerges in Variational Autoencoder as a line of future research. Many proposed models are heuristically motivated without grounding in a theoretical framework. The information bottleneck principle, as we have thoroughly discussed in Chapter 2 and 4 provides a formal way of analyzing some of the behavior. Rolinek et al. [2018] has made steps towards understanding disentanglement by drawing the connection to PCA. However, we argue that there remains progress of a better theoretical understanding of why these methods disentangle. In a similar vein, the literature is still missing a formal definition of disentanglement that is widely agreed upon.

Last, we echo the key motivation for a disentangled representation. Interpretable models in deep learning are imperative widespread adoption and truthful technologies. While the research on disentanglement needs to overcome some key challenges, it remains a promising field of research from which we expect progress in the years to come.

Bibliography

A. Achille and S. Soatto. Emergence of Invariance and Disentanglement in Deep Representations. *Journal of Machine Learning Research*, 19:1–34, 6 2017. URL http://arxiv.org/abs/1706.01350.

A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy. Deep Variational Information Bottleneck. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 12 2016. URL http://arxiv.org/abs/1612.00410.

A. A. Alemi, B. Poole, I. Fischer, J. V. Dillon, R. A. Saurous, and K. Murphy. Fixing a Broken ELBO. 11 2017. URL http://arxiv.org/abs/1711.00464.

S. M. Ali and S. D. Silvey. A General Class of Coefficients of Divergence of One Distribution from Another. *Journal of the Royal Statistical Society: Series B (Methodological)*, 28(1):131–142, 1966. ISSN 00359246. doi: 10.1111/j.2517-6161.1966.tb00626.x. URL http://www.jstor.org/stable/2984279.

M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. 1 2017. URL http://arxiv.org/abs/1701.07875.

Y. Bengio, P. Simard, and P. Frasconi. Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. ISSN 19410093. doi: 10.1109/72.279181.

Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems*, 2007. ISBN 9780262195683. doi: 10.7551/mitpress/7503.003.0024.

Y. Bengio, A. Courville, and P. Vincent. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 6 2012. URL http://arxiv.org/abs/1206.5538.

C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 1 2016. doi: 10.1080/01621459.2017.1285773. URL http://arxiv.org/abs/1601.00670http://dx.doi.org/10.1080/01621459.2017.1285773.

D. Bouchacourt, R. Tomioka, and S. Nowozin. Multi-Level Variational Autoencoder: Learning Disentangled Representations from Grouped Observations. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 2095–2102, 5 2017. URL http://arxiv.org/abs/1705.08841.

S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio. Generating Sentences from a Continuous Space. *CoNLL 2016 - 20th SIGNLL Conference on Computational Natural Language Learning, Proceedings*, pages 10–21, 11 2015. URL http://arxiv.org/abs/1511.06349.

Y. Burda, R. Grosse, and R. Salakhutdinov. Importance weighted autoencoders. In *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016.

C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner. Understanding disentangling in β-VAE. 4 2018. URL `http://arxiv.org/abs/1804.03599`.

G. Chechik, A. Globerson, N. Tishby, and Y. Weiss. Information bottleneck for Gaussian variables. *Journal of machine learning research*, 6(Jan):165–188, 2005. URL `http://www.jmlr.org/papers/volume6/chechik05a/chechik05a.pdf`.

R. T. Q. Chen, X. Li, R. Grosse, and D. Duvenaud. Isolating Sources of Disentanglement in Variational Autoencoders. *6th International Conference on Learning Representations, ICLR 2018 - Workshop Track Proceedings*, 2 2018. URL `http://arxiv.org/abs/1802.04942`.

T. M. Cover and J. A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, USA, 2006. ISBN 0471241954.

I. Csiszár. A class of measures of informativity of observation channels. *Periodica Mathematica Hungarica*, 2(1):191–213, 1972. ISSN 1588-2829. doi: 10.1007/BF02018661. URL `https://doi.org/10.1007/BF02018661`.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 1:4171–4186, 10 2018. URL `http://arxiv.org/abs/1810.04805`.

P. J. Diggle and R. J. Gratton. Monte Carlo Methods of Inference for Implicit Statistical Models. *Journal of the Royal Statistical Society. Series B (Methodological)*, 46(2):193–227, 1984. ISSN 00359246. URL `http://www.jstor.org/stable/2345504`.

J. Duchi, J. Duchi, E. Hazan, and Y. Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. 2010. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.164.213`.

R. M. Dudley. *Real Analysis and Probability*. Cambridge University Press, Cambridge, 2002. ISBN 9780511755347. doi: 10.1017/CBO9780511755347. URL `http://ebooks.cambridge.org/ref/id/CBO9780511755347`.

C. Eastwood and C. K. I. Williams. A framework for the quantitative evaluation of disentangled representations. 2018.

P. Glasserman. *Monte Carlo methods in financial engineering*, volume 53. Springer Science & Business Media, 2013.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. 6 2014. URL `http://arxiv.org/abs/1406.2661`.

I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, 12 2015. URL `https://arxiv.org/abs/1412.6572v3`.

GoogleResearch. TensorFlow: Large-scale machine learning on heterogeneous systems. *Google Research*, 2015.

K. Gregor, G. Papamakarios, F. Besse, L. Buesing, and T. Weber. Temporal Difference Variational Auto-Encoder. *7th International Conference on Learning Representations, ICLR 2019*, 6 2018. URL `http://arxiv.org/abs/1806.03107`.

A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test, 2012. ISSN 15324435.

T. Hastie, R. Tibshirani, and J. Friedman. *Springer Series in Statistics*, volume 27. 2009. ISBN 9780387848570. doi: 10.1007/b94608. URL `http://www.springerlink.com/index/D7X7KX6772HQ2135.pdf`.

K. He, X. Zhang, S. Ren, and J. Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, 2015. URL https://arxiv.org/abs/1502.01852.

I. Higgins, D. Amos, D. Pfau, S. Racaniere, L. Matthey, D. Rezende, and A. Lerchner. Towards a Definition of Disentangled Representations. 12 2018. URL http://arxiv.org/abs/1812.02230.

I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. B-VAE: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2019.

G. E. Hinton and R. S. Zemel. Autoencoders, minimum description length and Helmholtz free energy. In *Advances in neural information processing systems*, pages 3–10, 1994.

G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006. ISSN 08997667. doi: 10.1162/neco.2006.18.7.1527.

S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997. ISSN 08997667. doi: 10.1162/neco.1997.9.8.1735.

M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic Variational Inference. *Journal of Machine Learning Research*, 14:1303–1347, 2013. URL http://jmlr.org/papers/v14/hoffman13a.html.

K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2): 251–257, 1991. ISSN 08936080. doi: 10.1016/0893-6080(91)90009-T.

A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000.

J. Jacod and P. Protter. *Probability essentials*. Springer Science & Business Media, 2012.

I. T. Jolliffe. Principal components in regression analysis. In *Principal component analysis*, pages 129–155. Springer, 1986.

M. I. Jordan. *Learning in graphical models*. MIT Press, 1999. ISBN 9780262600323.

T. Karras, S. Laine, and T. Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. 12 2018. URL http://arxiv.org/abs/1812.04948.

R. W. Keener. Estimating Equations and Maximum Likelihood. pages 151–194. 2009. doi: 10.1007/978-0-387-93839-4{_}9.

R. W. Keener. *Theoretical statistics: Topics for a core course*. Springer, 2011.

H. Kim and A. Mnih. Disentangling by Factorising. *35th International Conference on Machine Learning, ICML 2018*, 6:4153–4171, 2 2018. URL http://arxiv.org/abs/1802.05983.

D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, 2015.

D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. 12 2013. URL http://arxiv.org/abs/1312.6114.

D. P. Kingma and M. Welling. An Introduction to Variational Autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 6 2019. doi: 10.1561/2200000056. URL http://arxiv.org/abs/1906.02691http://dx.doi.org/10.1561/2200000056.

D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling. Semi-Supervised Learning with Deep Generative Models. *Advances in Neural Information Processing Systems*, 4(January):3581–3589, 6 2014. URL http://arxiv.org/abs/1406.5298.

D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improving Variational Inference with Inverse Autoregressive Flow. *Advances in Neural Information Processing Systems*, pages 4743–4751, 6 2016. URL http://arxiv.org/abs/1606.04934.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012. ISBN 9781627480031.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 2017. ISSN 15577317. doi: 10.1145/3065386.

S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 3 1951. ISSN 00034851. URL `http://www.jstor.org/stable/2236703`.

A. Kumar, P. Sattigeri, and A. Balakrishnan. Variational Inference of Disentangled Latent Concepts from Unlabeled Observations. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 11 2017. URL `http://arxiv.org/abs/1711.00848`.

B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman. Building Machines That Learn and Think Like People. *Behavioral and Brain Sciences*, 40, 4 2016. URL `http://arxiv.org/abs/1604.00289`.

H. Larochelle and I. Murray. The neural autoregressive distribution estimator. In *Journal of Machine Learning Research*, 2011.

A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. *33rd International Conference on Machine Learning, ICML 2016*, 4: 2341–2349, 12 2015. URL `http://arxiv.org/abs/1512.09300`.

D. Levin and Y. Peres. *Markov Chains and Mixing Times*. 10 2017. ISBN 9781470429621. doi: 10.1090/mbk/107.

F. Liese and I. Vajda. On divergences and informations in statistics and information theory. *IEEE Transactions on Information Theory*, 52(10):4394–4412, 10 2006. ISSN 00189448. doi: 10.1109/TIT.2006.881731.

F. Locatello, S. Bauer, M. Lucic, G. Rätsch, S. Gelly, B. Schölkopf, and O. Bachem. Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations. *36th International Conference on Machine Learning, ICML 2019*, 2019-June:7247–7283, 11 2018. URL `http://arxiv.org/abs/1811.12359`.

Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang. The Expressive Power of Neural Networks: A View from the Width. 9 2017. URL `http://arxiv.org/abs/1709.02540`.

L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther. Auxiliary Deep Generative Models. 2 2016. URL `http://arxiv.org/abs/1602.05473`.

A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. Adversarial Autoencoders. 11 2015. URL `http://arxiv.org/abs/1511.05644`.

E. Mathieu, T. Rainforth, N. Siddharth, and Y. W. Teh. Disentangling Disentanglement in Variational Autoencoders. *36th International Conference on Machine Learning, ICML 2019*, 2019-June: 7744–7754, 12 2018. URL `http://arxiv.org/abs/1812.02833`.

L. Matthey, I. Higgins, D. Hassabis, and A. Lerchner. dSprites: Disentanglement testing Sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 12 1943. ISSN 1522-9602. doi: 10.1007/BF02478259. URL `https://doi.org/10.1007/BF02478259`.

L. Mescheder, S. Nowozin, and A. Geiger. Adversarial Variational Bayes: Unifying Variational Autoencoders and Generative Adversarial Networks. *34th International Conference on Machine Learning, ICML 2017*, 5:3694–3707, 1 2017. URL `http://arxiv.org/abs/1701.04722`.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*. International Conference on Learning Representations, ICLR, 1 2013. URL `https://arxiv.org/abs/1301.3781v3`.

T. P. Minka. Expectation Propagation for approximate Bayesian inference. 1 2013. URL http://arxiv.org/abs/1301.2294.

S. Mohamed, M. Rosca, M. Figurnov, and A. Mnih. Monte Carlo Gradient Estimation in Machine Learning. 6 2019. URL http://arxiv.org/abs/1906.10652.

A. Müller. Integral Probability Metrics and Their Generating Classes of Functions. *Advances in Applied Probability*, 29(2):429–443, 6 1997. ISSN 0001-8678. doi: 10.2307/1428011. URL https://www.cambridge.org/core/product/identifier/S000186780002807X/type/journal_article.

X. Nguyen, M. J. Wainwright, and M. I. Jordan. On Surrogate Loss Functions and f-Divergences. *The Annals of Statistics*, 37(2):876–904, 2009. ISSN 00905364. URL http://www.jstor.org/stable/30243651.

X. Nguyen, M. J. Wainwright, and M. I. Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 11 2010. ISSN 00189448. doi: 10.1109/TIT.2010.2068870.

S. Nowozin, B. Cseke, and R. Tomioka. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. 6 2016. URL http://arxiv.org/abs/1606.00709.

A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. 9 2016. URL http://arxiv.org/abs/1609.03499.

A. v. d. Oord, O. Vinyals, and K. Kavukcuoglu. Neural Discrete Representation Learning. *Advances in Neural Information Processing Systems*, 2017-December:6307–6316, 11 2017. URL http://arxiv.org/abs/1711.00937.

G. Peyré and M. Cuturi. Computational Optimal Transport, 2019.

G. C. Pflug. Sampling derivatives of probabilities. *Computing*, 42(4):315–328, 1989.

S. T. Rachev, L. B. Klebanov, S. V. Stoyanov, and F. J. Fabozzi. *The methods of distances in the theory of probability and statistics*, volume 9781461448693. Springer New York, 11 2013. ISBN 9781461448693. doi: 10.1007/978-1-4614-4869-3.

R. Ranganath, S. Gerrish, and D. M. Blei. Black Box Variational Inference. *Journal of Machine Learning Research*, 33:814–822, 12 2013. URL http://arxiv.org/abs/1401.0118.

D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. In *32nd International Conference on Machine Learning, ICML 2015*, 2015a. ISBN 9781510810587.

D. J. Rezende and S. Mohamed. Variational Inference with Normalizing Flows. *32nd International Conference on Machine Learning, ICML 2015*, 2:1530–1538, 5 2015b. URL http://arxiv.org/abs/1505.05770.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *31st International Conference on Machine Learning, ICML 2014*, volume 4, pages 3057–3070. International Machine Learning Society (IMLS), 1 2014. ISBN 9781634393973.

K. Ridgeway. A Survey of Inductive Biases for Factorial Representation-Learning. 12 2016. URL http://arxiv.org/abs/1612.05299.

K. Ridgeway and M. C. Mozer. Learning Deep Disentangled Embeddings with the F-Statistic Loss. *Advances in Neural Information Processing Systems*, 2018-December:185–194, 2 2018. URL http://arxiv.org/abs/1802.05312.

C. P. Robert and G. Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2005. ISBN 0387212396.

M. Rolinek, D. Zietlow, and G. Martius. Variational Autoencoders Pursue PCA Directions (by Accident). 12 2018. URL http://arxiv.org/abs/1812.06775.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain., 1958.

P. K. Rubenstein, O. Bousquet, J. Djolonga, C. Riquelme, and I. Tolstikhin. Practical and Consistent Estimation of f-Divergences. 5 2019. URL `http://arxiv.org/abs/1905.11112`.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. ISSN 1476-4687. doi: 10.1038/323533a0. URL `https://doi.org/10.1038/323533a0`.

B. Schölkopf. Causality for Machine Learning. 11 2019. URL `http://arxiv.org/abs/1911.10500`.

B. Schölkopf and A. J. Smola. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond Adaptive computation and machine learning. page 626, 2002.

C. E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review*, 5(1):3–55, 2001.

N. Siddharth, B. Paige, J.-W. van de Meent, A. Desmaison, N. D. Goodman, P. Kohli, F. Wood, and P. H. S. Torr. Learning Disentangled Representations with Semi-Supervised Deep Generative Models. *Advances in Neural Information Processing Systems*, 2017-December:5926–5936, 6 2017. URL `http://arxiv.org/abs/1706.00400`.

D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. Van Den Driessche, T. Graepel, and D. Hassabis. Mastering the game of Go without human knowledge. *Nature*, 2017. ISSN 14764687. doi: 10.1038/nature24270.

M. Själander, M. Jahre, G. Tufte, and N. Reissmann. EPIC: An Energy-Efficient, High-Performance GPGPU Computing Research Infrastructure. pages 6–8, 12 2019. URL `http://arxiv.org/abs/1912.05848`.

L. Song, X. Zhang, A. Smola, A. Gretton, and B. Schölkopf. Tailoring Density Estimation via Reproducing Kernel Moment Matching. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 992–999, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582054. doi: 10.1145/1390156.1390281. URL `https://doi.org/10.1145/1390156.1390281`.

H. W. Sorenson and D. L. Alspach. Recursive Bayesian estimation using Gaussian sums. *Automatica*, 7(4):465–479, 1971.

B. K. Sriperumbudur, K. Fukumizu, A. Gretton, B. Schölkopf, and G. R. G. Lanckriet. On integral probability metrics, \phi-divergences and binary classification. 1 2009. URL `http://arxiv.org/abs/0901.2698`.

B. Steiner, Z. Devito, S. Chintala, S. Gross, A. Paszke, F. Massa, A. Lerer, G. Chanan, Z. Lin, E. Yang, A. Desmaison, A. Tejani, A. Kopf, J. Bradbury, L. Antiga, M. Raison, N. Gimelshein, S. Chilamkurthy, T. Killeen, L. Fang, and J. Bai. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *NeuroIPS*, 2019. URL `https://github.com/pytorch`.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, 12 2014. URL `https://arxiv.org/abs/1312.6199v4`.

L. Theis, A. v. d. Oord, and M. Bethge. A note on the evaluation of generative models. 11 2015. URL `http://arxiv.org/abs/1511.01844`.

N. Tishby and N. Zaslavsky. Deep Learning and the Information Bottleneck Principle. *2015 IEEE Information Theory Workshop, ITW 2015*, 3 2015. URL `http://arxiv.org/abs/1503.02406`.

N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. 4 2000. URL `http://arxiv.org/abs/physics/0004057`.

I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schölkopf. Wasserstein auto-encoders. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018.

J. M. Tomczak and M. Welling. VAE with a VampPrior. *International Conference on Artificial Intelligence and Statistics, AISTATS 2018*, pages 1214–1223, 5 2017. URL `http://arxiv.org/abs/1705.07120`.

A. W. v. d. Vaart. *Asymptotic Statistics*. Cambridge University Press, 10 1998. ISBN 9780521496032. doi: 10.1017/CBO9780511802256. URL `https://www.cambridge.org/core/product/identifier/9780511802256/type/book`.

M. J. Wainwright. *High-Dimensional Statistics*. Cambridge University Press, 2 2019. doi: 10.1017/9781108627771.

M. J. Wainwright and M. I. Jordan. Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008. ISSN 1935-8237. doi: 10.1561/2200000001. URL `http://dx.doi.org/10.1561/2200000001`.

C. Zhang, J. Butepage, H. Kjellstrom, and S. Mandt. Advances in Variational Inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):2008–2026, 11 2017. URL `http://arxiv.org/abs/1711.05597`.

S. Zhao, J. Song, and S. Ermon. InfoVAE: Information Maximizing Variational Autoencoders. 6 2017a. URL `http://arxiv.org/abs/1706.02262`.

S. Zhao, J. Song, and S. Ermon. Towards Deeper Understanding of Variational Autoencoding Models. 2 2017b. URL `http://arxiv.org/abs/1702.08658`.