Investigating the Robustness of Disentangled Variational Autoencoders to Adversarial Examples

Sten Sootla[1]

Computational Statistics and Machine Learning

Supervised by Dr. John Shawe-Taylor and Dr. Cristina Calnegru

Submission date: 07.09.2018

[1] **Disclaimer:** This report is submitted as part requirement for the MSc degree in Computational Statistics and Machine Learning at UCL. It is substantially the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged.

Abstract

We analyse the robustness of disentangled deep variational autoencoders to adversarial examples on the MNIST handwritten images dataset. We found that the degree of disentanglement is positively correlated with the effectiveness of adversarial attacks, with models exhibiting higher degree of disentanglement being easier to fool. The code for reproducing the results can be accessed from the following repository: https://github.com/sootlasten/adversarial-on-disentangled.

Contents

Chapter 1

Introduction

In recent years, deep neural networks have shown impressive results in various supervised pattern recognition tasks, including image classification [48, 35], object detection [28], and machine translation [4]. Capitalizing on the resurgence of deep neural models in the supervised learning domain, unsupervised methods have also gone through a renaissance, as they have moved from modelling relatively simple datasets [2, 19] to being able to capture exceedingly complex real-world distributions [9, 62]. One of the most prominent generative models behind these successes is the variational autoencoder (VAE) [46], which merges conventional probabilistic methods with deep learning, using neural networks to parameterize both the generative and approximate posterior distributions in the the evidence lower bound.

The successes of these ”neural” models, consisting mainly of stacks of layers, each of which is composed of a linear mapping followed by a nonlinearity, can be attributed to the fact that they can theoretically approximate arbitarily complex functions [18]. However, as the appropriate functions are discovered automatically via a black-box optimization algorithm (e.g. gradient descent or an evolutionary method), the models can have counter-inuitive properties [66].

First, the learnt functions are often non-interpretable and unintuitive. Indeed, one of the most well-known and intriguing failure modes of most machine learning algorithms is the existence of adversarial examples - datapoints that are perceptually very close to examples from the original training distribution, but which behave atypically under the model [66]. For example, in the context of generative modelling, an adversarial example is an input which is visually very similar to the original, untampered image, but whose reconstruction differs markedly from the original's [47, 67]. An adversarial image could look like a regular depiction of the digit two, but which is reconstructed as the digit four by the generative model.

The fact that state-of-the-art deep learning models are subject to such adversarial perturbations challenges the narrative that they are unreasonably effective in generalizing outside the distribution they are trained on [6]. More practically, as the models under discussion are being pushed into production for use in semi-autonomous vehicles [11], drones [59], surveillance cameras [22] and malware detection pipelines [33], the ability to saliently fool machine learning algorithms into making arbitrary decisions poses a security threat with possible real-world consequences.

Second, the internal representations learnt by deep neural networks are infamously complex and hard to analyse. Indeed, Szegedy et. al. [66] have shown that directions along the coordinate axes are often no more semantically meaningful than other directions in the latent space. Similarly, Morcos et. al. [57] discovered that even if there exist individual units that are interpretable, they are no more important to the task at hand than neurons that don't exhibit clear correlation with specific semantic concepts. In other words, the models learn representations where units are *entangled* - each individual semnatically meaningful factor of variation in the input is explained by a combination of hidden units, as opposed to a single neuron.

It has been argued [7] that such entangled representations are suboptimal for generalization, and incorporating inductive biases which would steer the representations towards *disentanglement* would be beneficial. Indeed, it seems intuitively plausible that generalizing to novel configurations of concepts that were not encountered in the training distribution is easiest when relevant factors of variation in the input data are aligned with individual neurons in the latent space. Validating the intuition, such representations have recently shown promising results in zero-shot transfer learning [37], semi-supervised learning [20], and life-long learning [1].

This thesis investigates the possible relationship between the two intriguing properties of neural networks outlined above: the existence of adversarial datapoints on one hand, and the apparent overly complex internal representations on the other. It is reasonable to assume the existence of such a connection, since one way to construct adversarial examples for encoder-decoder type models is to perturb the source image in such a way that its internal representation would be close to the representation of some target image, while simultaneously meddling with the pixels of the source image as little as possible [47, 67]. The success of this methodology implies that variational autoencoders do not preserve the locality of inputs in their latent space. It has been argued, however, that models whose internal representation is disentangled encode nearby points in pixel space to similar representations in latent space as well [36]. If true, we would expect such models to be significantly more robust to adversarial inputs of this type.

To test the preceding hypothesis, we empirically evaluated the robustness of disentanglement-inducing VAEs to adversarial examples, and compared the results to regular, entangled VAEs. The analyses were carried out for the MNIST handwritten images dataset [52]. Contextually, this approach can be viewed as a natural extension to the works of Kos et. al. [47] and Alemi et. al. [3]. The former analysed the effectiveness of adversarial attacks to regular VAEs, but did not do so for their disentangled counterparts. The latter showed that by training a classifier with the same regularization principle that is usually used for VAEs to produce disentanglement, the model becomes more robust to adversarial inputs. However, as they operated in the context of image recognition, they were unable to evaluate whether the internal representation of their neural network was indeed factorized.

Surprisingly, we found that disentangled generative models are actually *less* robust to attacks on the MNIST dataset, contradicting our initial rationale. This stands in stark contrast to the work of Alemi et. al., who arrived at the opposite conclusion for models that were tasked with image classification, highlighting a fundamental difference between generative and discriminative models when it comes to adversarial examples.

The thesis is organized as follows. In Chapter 2, a sufficiently in-depth overview of variational autoencoders, disentanglement, and adversarial examples is given. Chapter 3 outlines the experimental methodology, and provides exact details of the analyses done in this thesis for reproducibility. Chapter 4 gives the results of the experiments.

Chapter 2

Related work and background

This chapter introduces the preliminary topics that are integral to understanding the methods and results of this thesis, with the assumption that the reader is comfortable with the basics of discriminative methods in machine learning. In addition, it provides an overview of the most important previous works that are closely related to ours.

To fully appreciate deep generative models and the problems they are best suited to tackle, it is paramount to first be aware of the context that they operate in. To that end, a brief introduction to the probabilistic view of pattern recognition is given in the first section. The second section builds on the first, introducing the variational autoencoder. The third section focusses on how to regularize generative models such that the learnt representations would be disentangled. The last section introduces the general problem of adversarial examples, and discusses their applicability to generative models.

2.1 Probabilistic learning and inference

2.1.1 Discriminative approach

Usually, in the case of conventional supervised learning, one is concerned about capturing the probability distribution of some predefined labels, conditioned on a set of inputs. For example, in classification, we try to model the categorical distribution of possible classes, given an image (or some set of higher-level features derived from raw pixels) belonging to one of the categories. Relatedly, we could be interested in continuous conditional distributions, as one might be when trying to estimate the future earnings of a company based on its current key performance indicators.

As a concrete example, due to Bishop [10], of how viewing machine learning through a probabilistc lens can yield useful insights into well-known methodologies, let us see how the ubiquitous sum-of-squares error function comes about naturally from a principled probabilistic approach. Suppose we have N possibly high-dimensional training inputs $\boldsymbol{X} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(n)}\}$, and their corresponding one-dimensional target values $\mathbf{t} = \{t^{(1)}, t^{(2)}, \dots, t^{(n)}\}$. Our uncertainty about the targets can be modelled as a Gaussian with a fixed precision β, whose mean is represented by a parameterized function $y(x; \boldsymbol{\theta})$ (e.g. an artificial neural network):

$$p(t^{(i)}|\boldsymbol{x}^{(i)}; \boldsymbol{\theta}, \beta) = \mathcal{N}(t^{(i)}|y(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \beta^{-1}).$$

If we further assume that the given targets were drawn independently from this normal distribution, we can estimate the unknown parameters $\boldsymbol{\theta}$ and β by maximizing the log-likelihood of the targets:

$$\begin{gathered}
\arg\max_{\boldsymbol{\theta}} \log p(\mathbf{t}; \boldsymbol{\theta}) = \\
\arg\max_{\boldsymbol{\theta}} \log \prod_{n=1}^{N} \mathcal{N}(t^{(n)}|y(\boldsymbol{x}^{(n)}; \boldsymbol{\theta}), \beta^{-1}) = \\
\arg\max_{\boldsymbol{\theta}} \sum_{n=1}^{N} \log \mathcal{N}(t^{(n)}|y(\boldsymbol{x}^{(n)}; \boldsymbol{\theta}), \beta^{-1}) = \\
\arg\max_{\boldsymbol{\theta}} \sum_{n=1}^{N} \log \left(\frac{\beta}{2\pi}\right)^{\frac{1}{2}} e^{-\frac{\beta}{2}(t^{(n)} - y(\boldsymbol{x}^{(n)}; \boldsymbol{\theta}))^2} = \\
\arg\max_{\boldsymbol{\theta}} \sum_{n=1}^{N} \frac{1}{2}(\log \beta - \log 2\pi) - \frac{\beta}{2}(t^{(n)} - y(\boldsymbol{x}^{(n)}; \boldsymbol{\theta}))^2 = \\
\arg\min_{\boldsymbol{\theta}} \sum_{n=1}^{N} (t^{(n)} - y(\boldsymbol{x}^{(n)}; \boldsymbol{\theta}))^2,
\end{gathered}$$

where the last equality follows from the fact that the first term inside the summation does not depend on the parameter $\boldsymbol{\theta}$, and that scaling the parameter-dependent term by a negative constant is equivalent to dropping the constant and turning the maximization problem into a minimization. With that, we have convinced ourselves that even in the usual discriminative setting, we are in fact making implicit use of a probabilistic approach to machine learning.

2.1.2 Generative approach

So far we have dealt with modelling the conditional distribution of the targets, given the inputs. Sometimes, however, we might be after the marginal distribution of the input data itself. Having such an object in hand, we are able to determine if an observed example has an unusually low probability of occurrence for anomaly detection [26], fill out missing dimensions of incomplete datapoints [27], or sample new, realistic datapoints from the learnt density function [51].

Capturing the marginal distribution of the inputs is often somewhat more involved than it was in the conditional case discussed previously. This is largely due the fact that high-dimensional, multi-modal datapoints have complex densities whose modelling with well-known distributions would yield suboptimal results. In such situations, additional structure is often incorporated into the probabilistic models in the form of lower-dimensional latent variables.

The inclusion of such auxiliary variables is often motivated by the hierarchical nature of physical processes. For example, by thinking about how images could be constructed from a generative point of view, we might imagine that first it is decided which objects to lay onto the scene. Second, one might go into more detail and think about what individual parts do the objects consist of. Finally, the orientations of the edges might be decided upon, after which the actual imagine could be drawn out onto the canvas. Interestingly, having imposed such hierarchical dependecy structure on our model, we can also traverse the dependency graph bottom-up, reducing our uncertainty about the latents, given an observed datapoint. This inversion of the generative process is called inference, and plays a central role in this thesis. Figure 2.1 illustrates the generative and inferential processes just described.

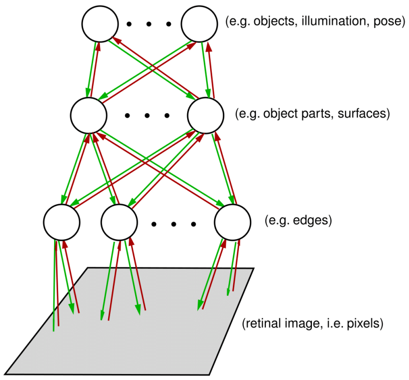

(e.g. objects, illumination, pose)
(e.g. object parts, surfaces)
(e.g. edges)
(retinal image, i.e. pixels)

Figure 2.1: Illustration of a probabilistic model with latent variables. Green arrows represent the generative process, in which higher-level concepts guide the development of lower-level ones, eventually leading to the generation of each pixel in an image. Red arrows depict inference, in which case the goal it to infer the underlying causes of an obtained input. Reprinted from M. Sahani, 2018 [64].

One immediate drawback to using more complicated latent variable models is that evaluating the likelihood turns out to be intractable, as one has to formally consider the probability of the observed datapoint under every possible setting of the latents. More specifically, consider an observed random variable $\boldsymbol{x}$, an unobserved latent $\boldsymbol{z}$, and their joint distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})$ [1], where the vector $\boldsymbol{\theta}$ denotes the parameters of the generative model [2]. Then, using the sum rule of probability, we can express the marginal of $\boldsymbol{x}$ as $p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z}$.

Naturally, as we are unable to evaluate the likelihood function, explicit optimization of its monotonic tranform, the log-likelihood, presents difficulties as well. Fortunately, there exists a workaround, which entails optimizing a lower bound of the quantity of interest instead. By introducing a new distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, parameterized by $\boldsymbol{\phi}$, into our framework, we can lower-bound the original log-likelihood as follows:

[1] Without loss of generality, let us assume the variables are continuous.

[2] Note that we can focus on the marginal likelihood of individual datapoints, rather than the marginal likelihood of the whole dataset, since under the independence assumption, the former is composed of a sum of the latter: $\log p_{\boldsymbol{\theta}}(x^{(1)}, x^{(2)}, \ldots, x^{(n)}) = \sum_{n=1}^{N} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})$.

$$
\begin{gathered}
\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \overset{\text{sum rule}}{=} \\
\log \int p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z} \overset{\text{multiply by identity}}{=} \\
\log \int p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) \frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} d\boldsymbol{z} \overset{\text{Jensen's inequality}}{\geq} \\
\int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} d\boldsymbol{z} =: \mathcal{L}(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\phi}).
\end{gathered}
\tag{2.1}
$$

To see that this bound, often called *free energy*, or the *evidence lower bound (ELBO)*, could indeed make the optimization problem easier, it helps to write it in two different forms. The first form illustrates that it can be written as the expected log-joint under the approximate posterior, plus an additive entropy term that does not depend on the parameters of the generative model:

$$
\begin{gathered}
\mathcal{L}(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\phi}) = \\
\int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z} - \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \log(\boldsymbol{z}|\boldsymbol{x}) d\boldsymbol{z} = \\
\mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) \big] + \mathcal{H}\big(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\big).
\end{gathered}
\tag{2.2}
$$

The second form reveals that the free energy is nothing more than the original log-likelihood that we are lower-bounding, minus an additional penalty term. The latter increases the gap between the bound and the log-likelihood in proportion to the distance between the approximate posterior and the real posterior, as measured by Kullback-Leibler divergence:

$$
\begin{gathered}
\mathcal{L}(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\phi}) = \\
\int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) p_{\boldsymbol{\theta}}(\boldsymbol{x})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} d\boldsymbol{z} = \\
\int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) + \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} d\boldsymbol{z} = \\
\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) - \mathcal{D}_{\mathrm{KL}}\big(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) || p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})\big).
\end{gathered}
\tag{2.3}
$$

Having laid out the two forms of the lower bound, one should notice that in Equation 2.2, only the first term depends on the generative parameters $\boldsymbol{\theta}$, while in Equation 2.3, the parameters of the approximate posterior feature only in the divergence term. This observation suggests a coordinate-wise optimization scheme where we take turns optimizing $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, leading us to the well-known Expectation-Maximization (EM) algorithm [27]. The approach is applicable when we have analytically tractable forms for the expectation of the log-joint, as well as for the true posterior under the current generative parameters $\boldsymbol{\theta}$.

What makes EM particularly appealing is that each full step of the algorithm either increases the log-likelihood, or the algorithm has converged to its local maximum. From Equation 2.3, we see that free energy is maximized with respect to $\boldsymbol{\phi}$ when the KL vanishes (as it is always non-negative), which is accomplished when the approximate posterior is set equal to the true posterior [3]. After that, the free energy equals the log-likelihood (under the current best-guess parameters). Since the next step, maximization with respect to $\boldsymbol{\theta}$, can not decrease the free energy, we can conclude from Equation 2.3 that the log-likelihood can only increase or stay constant, again due to the non-negativity of KL divergences.

Some of the latent variable models that have tractable expected log-joints and posteriors, yielding EM an appropriate tool for learning their parameters, include factor analysis, mixture of Gaussians, hidden Markov models, and linear Gaussian state-space models [42, 10]. However, we are often interested in using more complicated graphical models, particularly ones where continuous priors and conditionals have non-linear interactions. Assuming that we have the required derivatives of the lower bound in hand, one solutions would be to optimize the bound jointly using regular gradient-based methods. Keeping in mind that this approach is usually inferior to EM, as it loses the theoretical guarantees the latter provides, it is often the method of choice for handling models with intractabilities. A concrete example of such an approach is the variational autoencoder, the topic of the next section.

[3] It is important to distinguish between the best-guess parameters that we are working with when carrying out the optimization scheme, and the maximum-likelihood parameters. Our objective is to find the latter, but we are always operating with an estimated set of parameters. When referring to the *true* posterior, we are not implying that we have access to the true maximum-likelihood parameters, but the ”true” form of the posterior, i.e. $p(x|z)$ as opposed to $q(x|z)$.

2.2 Variational autoencoders

2.2.1 General framework

As already alluded to in the previous section, the variational autoencoder is a model that optimizes the lower bound of the log-likelihood using a gradient-based optimization scheme. The form of the free energy that it is explicitly working with is as follows:

$$\begin{gathered}\mathcal{L}(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\phi}) = \\ \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) p_{\boldsymbol{\theta}}(\boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} d\boldsymbol{z} = \\ \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) d\boldsymbol{z} + \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} d\boldsymbol{z} = \\ \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) d\boldsymbol{z} - \mathcal{D}_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) || p_{\boldsymbol{\theta}}(\boldsymbol{z})) = \\ \mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) \big] - \mathcal{D}_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) || p_{\boldsymbol{\theta}}(\boldsymbol{z})) \simeq \\ \frac{1}{L} \sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}^{(l)}) - \mathcal{D}_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) || p_{\boldsymbol{\theta}}(\boldsymbol{z})), \quad \text{where } \boldsymbol{z}^{(l)} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}^{(l)}|\boldsymbol{x}),\end{gathered} \tag{2.4}$$

where the first term of the last line is a Monte-Carlo estimate of the expectation above. The KL divergence is left as-is, because as we will soon see, it can be calculated analytically for commonly used distributions. In general, however, this term can be estimated by a finite set of samples as well.

An immediate problem that arises is that sampling directly from the approximate posterior is not a continuous operation. This presents a problem, as we need the gradient of the Monte-Carlo estimate of the lower bound with respect to $\boldsymbol{\phi}$ for optimization. In principle, the necessary gradients could be estimated using the score function (or REINFORCE) estimator, an ubiquitious method in the reinforcement learning literature [70]:

$$\begin{gathered}\nabla_{\boldsymbol{\phi}} \mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) \big] = \\ \int \nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) d\boldsymbol{z} \overset{\text{multiply by identity}}{=}\end{gathered}$$

$$\begin{aligned}
&\int \frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) d\boldsymbol{z} \overset{\text{derivative of log}}{=} \\
&\int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) d\boldsymbol{z} = \\
&\mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \big[\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) \big] \simeq \\
&\frac{1}{L} \sum_{l=1}^{L} \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}), \quad \text{where } \boldsymbol{z}^{(l)} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}^{(l)}|\boldsymbol{x}).
\end{aligned}$$

Unfortunately, due to its generality, the preceding estimator has a very high variance, rendering its use impractical in the context of generative modelling. One way to intuitively see the problem, due to Rezende et. al. [61], is to consider a set of K random variables $\boldsymbol{y} = \{y_1, y_2, \ldots, y_N\}$ with a joint distribution $p(\boldsymbol{y})$, and a function $f(\boldsymbol{y}) = \sum_{k=1}^{K} y_k$ whose derivative with respect to a single element $y_i \sim p(y_i)$ we are interested in evaluating. By using the score function estimator introduced above, we can express the derivative as the following expectation:

$$\begin{aligned}
\frac{\partial \mathbb{E}_{\boldsymbol{y}\sim p(\boldsymbol{y})}[f(\boldsymbol{y})]}{\partial y_i} &= \mathbb{E}_{\boldsymbol{y}\sim p(\boldsymbol{y})} \Big[\frac{\partial \log p(\boldsymbol{y})}{\partial y_i} f(\boldsymbol{y}) \Big] = \\
&\mathbb{E}_{\boldsymbol{y}\sim p(\boldsymbol{y})} \Big[\frac{\partial \log p(\boldsymbol{y})}{\partial y_i} \sum_{k=1}^{K} y_k \Big].
\end{aligned} \tag{2.5}$$

We can see that even though we are interested in the change of the function along a *single* direction, the multiplicative interaction between the log-derivative and the original function inside the expectation couples the random variables together. As a result, assuming that z_i are independent, and their variances, as well as those of their derivatives', are bounded, the variance of the the estimator in Equation 2.5 scales linearly with K, the input dimensionality of the target function. The assumption is also not restrictive in practice, since VAEs indeed assume a factored approximate posterior.

A better approach to estimating the gradient, introduced concurrently by Kingma & Welling [46], and by Rezende et. al. [61], is to reparameterize the lower bound in Equation 2.4 such that the necessary stochasticity that is injected into the model would not depend on any of its parameters. As a simple illustrative example of such an approach, suppose our approximate posterior is taken to be a univariate Gaussian: $z \sim p(z|x) = \mathcal{N}(\mu, \sigma^2)$. If we sampled z naively, we would not be able to calculate the required derivatives of this random variable, neither

$\frac{\partial z}{\partial \mu}$ nor $\frac{\partial z}{\partial \sigma}$. However, by reparameterizing the variable as $z = \mu + \sigma\epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$, we decouple the variables of our model from the stochasticity, resulting in differentiability: $\frac{\partial(\mu+\sigma\epsilon)}{\partial \mu} = 1$ and $\frac{\partial(\mu+\sigma\epsilon)}{\partial \sigma} = \epsilon$.

Let us denote the differentiable transformation of the random variable as $\boldsymbol{z} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \boldsymbol{x})$, where $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$ is the auxiliary noise variable that does not depend on the model's parameters. The lower bound of a mini-batch of M samples, randomly drawn from the full dataset, is then given by Equation 2.6. In practice, Kingma & Wellig [46] found that the number of samples drawn from the approximate posterior can be taken to be 1 (i.e. $L = 1$), provided that the mini-batch is large enough. As the resulting bound is jointly differentiable with respect to $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, it can be optimized with any of the popular stochastic optimization methods, such as SGD, Adam [45], or RMSProp [68].

$$\begin{gathered}\frac{1}{M}\sum_{i=1}^{M}\mathcal{L}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}, \boldsymbol{\phi}) = \\ \frac{1}{M}\sum_{i=1}^{M}\Big\{\mathbb{E}_{\boldsymbol{\epsilon}\sim p(\boldsymbol{\epsilon})}\big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}|g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \boldsymbol{x}^{(i)}))\big] - \mathcal{D}_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}^{(i)})||p_{\boldsymbol{\theta}}(\boldsymbol{z}))\Big\} \simeq \\ \frac{1}{ML}\sum_{i=1}^{M}\Big\{\sum_{l=1}^{L}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}|\boldsymbol{z}^{(i,l)}) - \mathcal{D}_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}^{(i)})||p_{\boldsymbol{\theta}}(\boldsymbol{z}))\Big\}, \\ \text{where } \boldsymbol{z}^{(i,l)} \sim g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \boldsymbol{x}), \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}).\end{gathered} \tag{2.6}$$

Unlike the REINFOCE estimator, the use of the "reparameterization-trick" allows for the construction of an efficient gradient estimator whose variance is unaffected by the number of random variables in the system. Reusing our previous notation from Equations 2.5 and 2.6, suppose that $\boldsymbol{y} = g(\boldsymbol{\epsilon}, \boldsymbol{y})$ and $y_i = g(\epsilon_i, y_i)$. Then, the partial derivative of the expectation of $f(\boldsymbol{y}) = \sum_{k=1}^{K} f(y_k)$ depends only on the variable y_i it is taken with respect to, as demonstrated by the following derivation:

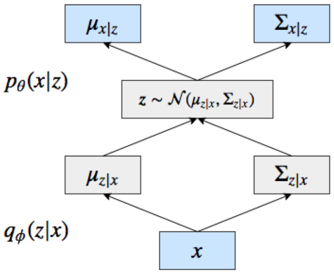
$$\begin{gathered}\frac{\partial \mathbb{E}_{\boldsymbol{y}\sim p(\boldsymbol{y})}[f(\boldsymbol{y})]}{\partial y_i} = \frac{\partial \mathbb{E}_{\boldsymbol{\epsilon}\sim p(\boldsymbol{\epsilon})}[f(g(\boldsymbol{\epsilon}, \boldsymbol{y}))]}{\partial y_i} = \\ \int p(\boldsymbol{\epsilon})\frac{\partial}{\partial y_i}\Big\{\sum_{k=1}^{K} f(g(\epsilon_k, y_k))\Big\}d\boldsymbol{\epsilon} =\end{gathered}$$

$$\int p(\boldsymbol{\epsilon}) \frac{\partial f(g(\epsilon_i, y_i))}{\partial y_i} d\boldsymbol{\epsilon} = \mathbb{E}_{\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})} \big[\frac{\partial f(g(\epsilon_i, y_i))}{\partial y_i} \big].$$

With that, we have laid out all the fundamental components that make up a variational autoencoder. The next subsection covers some of the specifics that are still left to be filled in, particularly the forms and parameteriztions of the distributions in play.

2.2.2 Specific example

The name ”variational autoencoder” is usually reserved for a particular realization of the preceding framework, one where both the approximate posterior $q_{\boldsymbol{\phi}}(\boldsymbol{x}|\boldsymbol{z})$, as well as the conditional in the generative part of the model, $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$, are parameterized by artificial neural networks. These distributions themselves are usually taken to be multivariate normals with diagonal covariances. The prior does not generally have learnable parameters, as it is taken to be an isotropic Gaussian: $p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{0}, \boldsymbol{I})$. Figure 2.2 gives a schematic of this setup.



$\boldsymbol{\mu}_{x|z}$
$\boldsymbol{\Sigma}_{x|z}$
$p_\theta(x|z)$
$z \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x})$
$\boldsymbol{\mu}_{z|x}$
$\boldsymbol{\Sigma}_{z|x}$
$q_\phi(z|x)$
x

Figure 2.2: An illustration of a variational autoencoder. The blue rectangles represent entities whose dimensionality is equal to that of the input, while the gray rectangles signify parts of the graph that deal with the latent variable.

A particularly nice property that follows from the simple parameterization is the analytic tractability of the KL divergence featuring in the lower bound. Denoting the mean of the i-th dimesion of the D-dimensional approximate posterior as μ_i, and the corresponding standard deviation as σ_i, the KL-divergence between the approximate posterior and the prior can be solved in closed-form [46]:

$$\mathcal{D}_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p_{\boldsymbol{\theta}}(\boldsymbol{z})) = \frac{1}{2}\sum_{i=1}^{D}(1 + \log((\sigma_i)^2) - \mu_i^2 - \sigma_i^2). \tag{2.7}$$

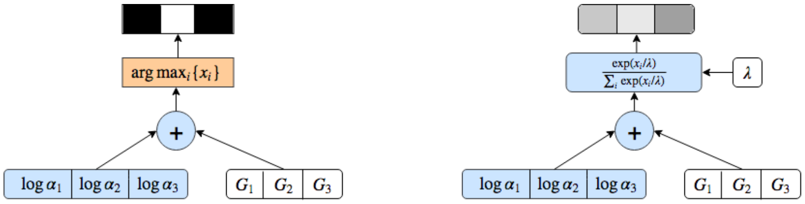
Looking at the general outline of the model in Figure 2.2, it becomes apparent why it is called an autoencoder. An input $\boldsymbol{x}$ is fed through a neural network, which outputs the mean and diagonal covariance of the approximate posterior. Making use of the reparameterization trick, a sample is then drawn from this distribution. As the the dimension of the latent variable is usually much lower than that of the input, it is akin to the bottleneck layer of a regular autoencoder, and the subnetwork is dubbed an encoder. The drawn vector $\boldsymbol{z}$ is subsequently fed through a second neural network, which outputs the estimated parameters of $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$. As the resulting mean is obtained by upsampling the latent, and the conditional is maximized if it is equal to the original input, it bears resemblence to the decoding part of a conventional autoencoder, justifying the use of the name ”decoder” for the underlying neural network.

2.2.3 Incorporating discrete variables

In the previous subsections, we focussed on variational autoencoders with continuous variables, namely ones with Gaussian distributions. Sometimes, however, it would be appropriate for the posterior to include some discrete variables as well. For example, when modelling handwritten digits, some of dimensions of the posterior could correspond to thickness, angle, size, and digit class. While the first three factor of variation are reasonably modelled by continuous distributions, a discrete one is more natural for the last. Unfortunately, the reparameterization trick that allowed joint optimization of the parameters in the Gaussian case is not applicable for discrete variables.

To get around taking direct samples from a discrete distribution, one might employ the Gumbel-Max trick [54]. Suppose our one-hot encoded random variable of interest, Y, has a discrete distribution with K choices, parameterized by unnormalized probabilities $(\alpha_1, \alpha_2, \ldots, \alpha_K)$, where $\alpha_i \in (0, \infty)$. A sample y can then be drawn from this distribution using the following scheme:

1. Draw K independent samples from a uniform distribution whose supported lies on the open unit interval: $U_i \sim \text{Uniform}(0, 1)$, where $i = 1, 2, \ldots, K$.



(a) Gumbel-max trick

(b) Gumbel-softmax trick

Figure 2.3: Schematics of the Gumbel-max and Gumbel-softmax tricks for 3 random variables. The blue shapes represent the parts of the system that we can and are interested in taking the gradients of. Note how the arg max operation in orange blocks the gradient, as its output is discrete, whereas the softmax operation relaxes the one-hot representation of the output, depicted as shades of gray instead of black and white, thus permitting gradient flow. Adapted from Maddison et. al. [55]

2. For each of the drawn samples, evaluate $x_i = \log \alpha_i + G_i$, where $G_i = -\log(-\log U_i)$ is a sample from the Gumbel distribution (hence the name of the trick).

3. Set $y_{\arg\max_i\{x_i\}}$ to 1, and all other dimensions of y to 0.

This looks promising, as the uniform distribution that we sample from does not depend on any of the parameters of the encoder. However, it is still of no use for our purposes, as we have now introduced a non-continuous arg max operation. A solution, proposed concurrently by Maddison et. al. [55], and Jang et. al. [41], is to approximate it by a softmax function with a tunable temperature parameter $\lambda \in (0, \infty)$, as shown in Equation 2.8. Note that in this case, our sample, drawn from what the authors call a Concrete distribution, is no longer discrete, but rather a continous relaxation of it that approximates a one-hot vector as $\lambda \to 0$. The difference between the original Gumbel-Max trick, and its Concrete relaxation, sometimes dubbed the Gumbel-Softmax trick, is illustrated in Figure 2.3.

$$y_i = \frac{\exp((\log \alpha_i + G_i)/\lambda)}{\sum_{j=1}^{K} \exp((\log \alpha_j + G_j)/\lambda)}. \tag{2.8}$$

Two additional considerations arise when Concrete variables are used in VAEs. First, since no gradients need to be computed at test time, the distributions can be discretized, allowing real categorical variables to be fed through the graph.

Second, the KL term in the objective needs to be modified, as simply ”plugging” the continous samples into discrete probability mass functions does not result in a valid KL divergence. This is because the samples are not drawn from the same distribution as is in the enumerator under the logarithm, as shown by Equation 2.9, where the Concrete posterior is denoted as $q^C_{\boldsymbol{\alpha},\lambda}$, while $q_{\boldsymbol{\alpha}}$ and $p_{\boldsymbol{\alpha}}$ stand for the approximate categorical posterior and prior, respectively:

$$\mathbb{E}_{\boldsymbol{z}\sim q^{\mathrm{C}}_{\boldsymbol{\alpha},\lambda}(\boldsymbol{z}|\boldsymbol{x})}\Big[\log\frac{q_{\boldsymbol{\alpha}}(\boldsymbol{z}|\boldsymbol{x})}{p_{\boldsymbol{\alpha}}(\boldsymbol{z})}\Big]. \tag{2.9}$$

The authors of the original paper [55] provide several options for modifying the KL term, and deem that replacing the discrete masses in the KL with Concrete densitites yields best results. However, unlike the Gaussian case, there is no closed form expression for the KL divergence between two Concrete distributions, so it is necessary to approximate it via sampling. For the types of models considered in this thesis, a variety of estimators, including the one that Maddison et. al. [55] advocate, have been found to exhibit too high of a variance, producing poor results as a consequence (Emilien Dupont, personal communication, August 1, 2018). Thus, we opted for replacing the original penalty with an analytic discrete KL term intead:

$$\mathbb{E}_{\boldsymbol{d}\sim q_{\boldsymbol{\alpha}}(\boldsymbol{d}|\boldsymbol{x})}\Big[\log\frac{q_{\boldsymbol{\alpha}}(\boldsymbol{d}|\boldsymbol{x})}{p_{\boldsymbol{\alpha}}(\boldsymbol{d})}\Big]=\sum_{i=1}^{K}q_{\boldsymbol{\alpha}}(d_i|\boldsymbol{x})\log\frac{q_{\boldsymbol{\alpha}}(d_i|\boldsymbol{x})}{p_{\boldsymbol{\alpha}}(d_i)}, \tag{2.10}$$

where the vector $\boldsymbol{d}$ represents the one-hot categorical variable. While this strategy provides good results empirically, it is important to emphasize that the use of this KL term in the VAE objective does not necessarily result in a lower-bound of the marginal log-likelihood.

2.3 Disentangled representations

According to Bengio et. al. [7], a representation is disentangled if it separates explanatory sources of the data into different dimensions (e.g. units) in the latent space. Learning such a representation in an unsupervised manner is challenging, as there is no obvious mathematical objective to optimize. Indeed, the issue is akin to the chicken-and-egg problem, since one would seemingly need to be already aware of the main factors of variation to be able to distill them from the data into the model.

Most earlier work on learning disentangled representations has thus tackled the problem with weakly supervised methods, which require some a-priori knowledge of the data generating factors [38, 60, 71]. DC-IGN [49] is perhaps the most well-known method among such approaches. It adopts a clever curriculum learning [8] strategy to encode the data generating factors into a small subset latent variables in a VAE, each of which is a-priori assigned a symbolic interpretation. During training, a random latent from this predefined group is chosen per each mini-batch of training data, which in turn is carefully constructed such that all training examples in it would vary in only the chosen factor. Finally, the bottleneck layer of the underlying VAE is regularized to indeed only capture all the variance in the current batch with a single latent.

A particularly interesting approach that requires no supervision is Shmidhuber's predictability minimization [65]. It introduces additional submodules for each unit in an autoencoder's bottleneck layer which try to predict the unit from the remaining ones. In turn, the model is is concurrently optimized to minimize the predictability of the units. The scheme was motivated by the idea that such a minimax game would give rise to an internal representation where each dimension is statistically independent of the others. While this 'competitive learning' framework was indeed found to be able to learn such factorial codes on toy binary datasets, it has not been shown to be able to scale up to more complicated inputs.

The first scalable, purely unsupervised method that showed reasonable level of disentanglement on real-world datasets was InfoGAN, proposed by Chen et. al. [17]. They extended the powerful generative adversarial network (GAN) framework [30] with an additional objective, which would encourage the information between a small subset of the input noise variables and the generated output to be maximized. In practice, this was achieved by adding an additional auxiliary network into the system. The subnetwork, optimized concurrently with the original GAN objective, was fed with the generated images, and tasked to predict the values of a subset of the noise variables that gave rise to it. Contextually, InfoGAN can be seen as a close cousin of both DC-IGN and predictability minimization, borrowing the idea that only a subset of latents should be interpretable from the former, while having an additional subnetwork that predicts the latent activations is akin to the approach of the latter.

The work most relevant to this thesis is that of Higgins et. al. [36], which demonstrated that simply adding more weight to the KL term in the VAE objective function encourages the posterior to become disentangled. In particular, they multiplied the divergence measure by a constant $\beta > 1$, giving the model its name, β-VAE:

$$\mathcal{L}(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\phi}) = \mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\big] - \beta\,\mathcal{D}_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p_{\boldsymbol{\theta}}(\boldsymbol{z})). \tag{2.11}$$

The method superseded InfoGAN, proving to be superior in disentangling, as well as enjoying better training stability. The use of GANs as a backbone has proved to be prohibitive for InfoGAN, as they are notorious for their extremely unstable behaviour in training. Adding further complications, vanilla GANs do not optimize an explicit density function, making it hard to measure the training progress quantitatively [32]. While some recent works have addressed these problems with some success [43, 34, 9], Kim et. al. [44] found that pluggin them into the InfoGAN framework gives inconsistent results. As β-VAE uses a principled likelihood-based approach in place of a minimax game whose solution lies at a saddle point in a high-dimensional space, its training dynamics are much more wiedly.

In a closely related follow up work to β-VAE, Burgess et. al. [13] built upon it by proposing an enhanced optimization scheme based on some well-developed intuitive arguments. They analysed the β-VAE objective via the lens of the information bottleneck principle [69]. The principle posits that a good internal representation Z of a learning system is one which is maximally informative about the output Y, while also compressing the information in the input X. Mathematically, this intuition is expressed as maximizing a Lagrangian, where "informativeness" is naturally measured by mutual information, denoted as $I(\cdot;\cdot)$:

$$\max\big[I(Z;Y) - \beta I(X;Z)\big] \tag{2.12}$$

Taking this information-theoretic view of the variational autoencoder, we can conceptualize the probabilistic encoder parameterizing a diagonal Gaussian $q(\boldsymbol{z}|\boldsymbol{x})$ as an independent set of noisy channels $\{z_i\}$, each transmitting information about the input $\boldsymbol{x}$. Accordingly, the KL divergence in Equation 2.11 can be thought of as the maximum capacity of said channels per data sample. When the posterior collapses to a unit Gaussian prior $p(\boldsymbol{z})$, the KL term vanishes, and no information about the input is let through the latent bottleneck. Armed with this intuitive framework, it is interesting to analyse how does the reconstruction loss of VAE influence the development of the latent channels.

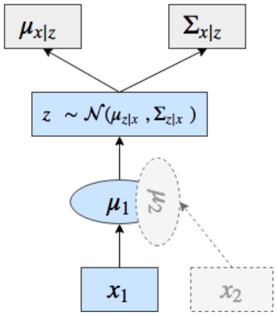


Figure 2.4: A schematic that illustrates why a VAE is pressured to preserve the locality of inputs in the latent space. If the parmeters of $q(\boldsymbol{z}|\boldsymbol{x}_1)$ are close to that of $q(\boldsymbol{z}|\boldsymbol{x}_2)$, as depicted by the overlapping circles labelled $\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2}$, a latent $\boldsymbol{z}$ could be sampled such that $\mathcal{N}(\boldsymbol{x}_2|\boldsymbol{\mu}_{\boldsymbol{x}|\boldsymbol{z}}, \boldsymbol{\Sigma}_{\boldsymbol{x}|\boldsymbol{z}}) > \mathcal{N}(\boldsymbol{x}_1|\boldsymbol{\mu}_{\boldsymbol{x}|\boldsymbol{z}}, \boldsymbol{\Sigma}_{\boldsymbol{x}|\boldsymbol{z}})$. Intuitively, this can be thought of as reconstructing the ”wrong” datapoint, a mistake whose cost is proportional to the distance between the original and ”wrong” datapoint in pixel space.

Burgess et. al. [13] argue that trading off the two terms in the β-VAE objective encourages an encoder to be learned which would have the property that nearby points in data space are close together in the latent space as well. Indeed, suppose that two very different datapoints (as measured by the L2 norm) in pixel-space have similar representations in the latent space. As random noise that is injected into the bottleneck layer, it is likely that the two codes get ”mixed up”, resulting in the decoder reconstructing the wrong input and incurring a high loss. The situation is illustred in Figure 2.4. To avoid such a scenario, we would like the encoding of each datapoint to be as far from others' encodings as possible, which is achieved by squeezing the posterior variances and dispersing their means across datapoints. This, however, is in turn discouraged by the KL penalty term, which pressures the channels to be overlapping for all datapoints. A reasonable compromise between these two opposing forces would be an arrangmenet where the latent space is such that when we do make the occasional mistake of decoding the wrong datapoint, we are at least outputting something that is not too far from the original.

A second fundamental observation Burgess et. al. [13] make is that when the latent channel capacities are constrained to be very low, the only information that gets through the bottleneck is one whose utilization by the decoder increases the value of the reconstruction term $\mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\big]$ the most. As an example, suppose we have a dataset of white shapes drawn on a black canvas, each of which is has 5 factors of variation: translation along the vertical and horizontal axes, scale, oritentation, and class (heart, square, or ellipse) [56]. Now, for a high value of β, only positional information of the sprite might get through the overly constrained latent bottleneck, as getting the position wrong incurs the most cost in terms of log-likelihood. Subsequently, by gradually increasing the capacities of the latent channels, the model starts to allocate additional latents to new factors of variation, like scale and rotation. Crucially, due to the data locality pressure described before, all the factors of variation are encoded into distinct latents, satisfying our working definition of disentanglement. Figure 2.5 illustrates this phenomenon.

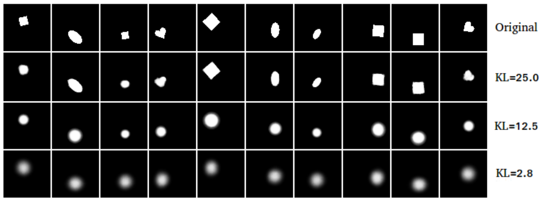


Figure 2.5: A gradual encoding of data generative factors into the latents on the dSprites dataset [56] for randomly chosen datapoints. At the start of training, when the capacities of the latent channels are kept low, only the x and y positions of the sprites is encoded (bottom row). Next, as capacity is increased, the size of the sprite is captured (third row). Finally, the rotation and class of the figure is discovered (second row).

The intuitive framework laid out above suggests a natural modification to the β-VAE objective, given by Equation 2.13. In the new formulation, the capacities of the channels, denoted as C, are made explicit, and increased gradually in the course of training. The new hyperparameter γ specifies the degree of pressure put on the latent channels to be close to their desired capacity. It is usually chosen big enough that the KL term would be relatively close to C at all times, as this makes the external control of the channel capacities rather convenient.

$$\mathcal{L}(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\phi}) = \mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\big] - \gamma\,|\mathcal{D}_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p_{\boldsymbol{\theta}}(\boldsymbol{z})) - C|. \tag{2.13}$$

While β-VAE with a controlled capacity increase showed good results disentangling continuous factors of variation, it struggled to capture discrete generative factors. For example, it was unable to disentangle the individual classes in the MNIST handwritten digits dataset. As a way to overcome this difficuly, Dupont introduced JointVAE [23], which augmented the original multivariate normal latent distirbution of β-VAE with categorical Concrete random variables, leading to the following objective:

$$\begin{gathered}\mathcal{L}(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\phi}) = \mathbb{E}_{\boldsymbol{z},\boldsymbol{c}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z},\boldsymbol{c}|\boldsymbol{x})}\big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z},\boldsymbol{c})\big] - \\ \gamma\big\{|\mathcal{D}_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p_{\boldsymbol{\theta}}(\boldsymbol{z})) - C_z| + |\mathcal{D}_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{c}|\boldsymbol{x})||p_{\boldsymbol{\theta}}(\boldsymbol{c})) - C_c|\big\},\end{gathered} \tag{2.14}$$

where $\boldsymbol{z}$ and $\boldsymbol{c}$ denote the continous Gaussian and Concrete latent variables, respectively. The new model was shown to be able to outperform the original formulation of β-VAE on datasets where capturing discrete generative factors is paramount to achieving good reconstruction quality, as is the case for MNIST. However, complete disentanglement of the sprites dataset is still beyond the capabilities of current models, as the types of shapes are all quite similar in terms of the area they cover, making the possible increase in log-likelihood marginal.

2.4 Adversarial examples

The problem of adversarial examples in the context of deep neural networks was first articulated by Szegedy et. al. [66]. They showed that by perturbing the pixels of the input image by an (almost) imperceptibly small amount, it is possible to cause neural-network based classifiers to assign high likelihoods to arbitrary classes. It is important to stress that the pixels of the original image are not changed randomly, but are rather done so in a principled way. More specifically, the authors use box-constrained L-BFGS [58] to solve the following optimization problem [4]:

$$\begin{aligned} &\text{minimize } c\|\boldsymbol{r}\|_2 + \text{loss}_f(\boldsymbol{x}+\boldsymbol{r}, l) \\ &\text{subject to } \boldsymbol{x}+\boldsymbol{r} \in [0,1]^m, \end{aligned} \tag{2.15}$$

where $\boldsymbol{r}$ is the m-dimensional perturbation vector that is added to the original image $\boldsymbol{x}$, such that the classifier f would output the class l. loss_f is some continuous measure of classification loss, commmonly taken to be cross-entropy, while the coefficient $c > 0$ balances the the size of the perturbation and the loss for the generated adversarial image. Of course, the problem is only of interest when $f(\boldsymbol{x}) \neq l$. Figure 2.6 illustrates some adversarial examples $\boldsymbol{x}+\boldsymbol{r}$ generated by this procedure.

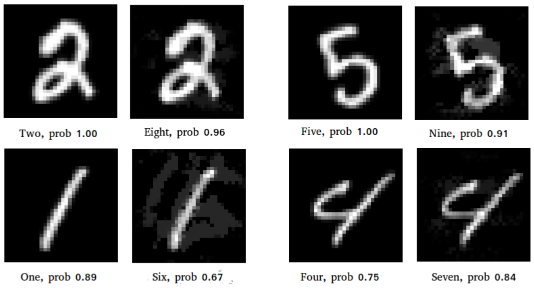


Figure 2.6: Adversarial examples for the MNIST handwritten digits dataset. The left image of each pair depicts the original, correctly classified image. The right image, however, is the adversarial input, derived from the original, that fools the classifier.

In subsequent work, Goodfellow et. al. [31] argue that the existence of adversarial examples can be at least partially explained by the linear nature of the underlying classifiers. Linear models have the property that their outputs change proportionally to the change in the intensity of any of the pixels. Thus, given an input with high enough dimensionality, one can make small perturbtions to a large number of pixels, all of which add up to produce a large shift in the output. Goodfellow et. al. make the observation that this same phenomenon

[4] Note that this specific formulation constrains the images to have a single channel and either be normalized or grayscale. However, generalizing it to unnormalized RGB images is straightforward.

can manifest itself in modern deep neural networks, as they are often explicitly constructed to be rather linear for ease of optimization. Illustrative examples include the rectified linear units as general-purpose activation functions [29], LSTMs for time-series analysis [39], and residual networks for compute vision [35]. Moreover, adding further evidence in favour of the linearity hypothesis, they observed that RBF networks [12] - a method that classifies inputs based on their similarity (as measured by the exponentiated quadratic kernels) to a set of predefined centroids - are significantly more robust to adversarial examples.

One of the most surprising properties of adversarial examples is their generality and transferability. Early work on the problem found that they generalize across different partitions of the same dataset [66] and across models with different architectures [31]. More recently, Kurakin. et. al. [50] discovered that they can also be extrapolated to the physical world, showing that models trained on the regular ImageNet [21] dataset are misclassifying adversarial examples even if they are printed on paper and viewed through a physical camera. Somewhat reassuringly, they have also been found to fool time-limited humans [25]. While most research on adversarial examples has focussed on image classification, models trained to solve other kinds of machine learning tasks are subject to them as well, including ones specializing in speech-to-text [15], neural machine translation [24], and deep reinforcement learning [5].

Pertinent to this thesis is the work of Kos. et. al. [47], who also investigate adversarial examples in the context of generative models. They found that VAEs and VAE-GANs [51], a derivative of VAEs which enjoys better sample quality due to the fact that its reconstruction loss operates in the feature space of a GAN, are also suspect to them. To generate adversarial examples for such models, they took inspiration from Sabour et. al. [63], and perturbed the source image in such a way that its predicted posterior mean would be close to the posterior mean of some target image. Mathematically, this optimization problem is equivalent to the one in Equation 2.15, with the classification loss replaced by the L2 norm measuring the discrepancy of the latent representations. They did, however, use Adam instead of L-BFGS for optimization, as advocated by Carlini & Wagner [14].

Another relevant work from our point of view is that of Alemi et. al. [3], in which they parameterize the information bottleneck in Equation 2.12 using neural networks. By subsequently deriving a variational lower bound to this objective and making use of the reparatemerization trick, they end up with a model anologus to that of β-VAE, given that the output variable Y in Equation 2.12 is taken to be the reconstruction of the input. However, as their treatment is more broad, they are also able to generalize this approach to a classification setting, by taking Y to be the class label. Intuitively, this amounts to swapping the decoder of β-VAE with an appropriate classifier. To demonstrate the usefulness of the bottleneck principle, they measured the robustness of the resulting classifier to adversarial examples on the MNIST and ImageNet datasets. They found that the model becomes progressively harder to fool as the capacity of the bottleneck is increased.

Chapter 3

Methods

3.1 Dataset

We used the MNIST handwritten digits dataset to measure the effect that a disentangled representation has on the ability to generate adversarial examples for VAEs. The dataset consists of $70,000$ images (partitioned into a train-eval-test split of $50,000$, $10,000$ and $10,000$ datapoints, respectively), each of which is assigned to a numeric class from 0 to 9. MNIST is one of the few datasets that have been extensively analysed both in the disentanglement, as well as in the adversarial example literature. It was the first dataset for which adversarial examples were constructed [66], and it has been shown that at least 3 factors of variation of handwritten digits can be disentangled on this dataset: class, angle, and thickness [17, 23]. Moreover, the two most closely related works to this thesis, that of Kos et. al. and Alemi et. al., concentrate on MNIST as well, making drawing comparisons easier.

We also experimented with generating adversarial examples for other datasets, including dSprties and CelebA, a large-scale dataset of celebrety faces [53]. Unfortunately, we found that dSprites is not sensitive to adversarial examples in the generative setting. This phenomenon is possibly due to the relatively simple visual appearance of the datapoints, resulting in 2 different inputs to have a relatively sparse set of common features that could be made use of while generating an adversarial input. Similarly, for CelebA, we were unable to construct adversarial examples with enough visual fidelity that would permit reasonable quantitative analysis to be carried out.

3.2 Model architecture and optimization

As our disentangled model, we used JointVAE [23], an equivalent model to β-VAE, but with both continuous and discrete (or Concrete) latent variables, whose channel capacities were gradually increased during training. A regular VAE without any regularization was taken to be a baseline. However, as decreasing the capacity of the latent channels constrains the amount of information that gets through the bottleneck, the reconstructions produced by JointVAE are somewhat worse than that of a regular VAE. In order to account for that effect, we also trained a second baseline model, which was regularized using weight-decay and achieved similar reconstruction quality as our disentangled model. All 3 trained models had the same architecture as was used by Dupont [23], outlined in Table 3.1. Since the MNIST dataset is grayscale, with most values near the boundaries (i.e. near 0 or 1), we modelled each dimension of the conditional distribution of the output as an independent Bernoulli random variable, as is common in literature [46, 36]. Accordingly, the outputs of the decoder were the expected values of these random variables.

Encoder	**Decoder**
$\boldsymbol{x} \in \mathbb{R}^{32\times32}$ grayscale image	$\boldsymbol{z} \in \mathbb{R}^{20}$
Conv 4×4. 32 ReLU, 2 stride	Dense. 128 ReLU
Conv 4×4. 32 ReLU, 2 stride	Dense. 64×4×4 ReLU
Conv 4×4. 64 ReLU, 2 stride	Upconv 4×4. 64 ReLU, 2 stride
Dense. 128 ReLU	Upconv 4×4. 32 ReLU, 2 stride
Dense. 20 linear	Upconv 4×4. 32 sigmoid, 2 stride

Table 3.1: VAE architecture used for the experiments carried out in this work. The bottleneck layer parameterized a 10-dimensional diagonal multivariate normal distribution, as well as a 10-way categorical Concrete distribution, which results in a 20-dimensional latent vector to be fed into the decoder.

All models were trained with the Adam optimizer with a constant learning rate of 0.0001 for $100,000$ randomly chosen batches from the training set with a mini-batch size of 64. Following the guidelines of Maddison et. al. [55] [1], the

[1] They show that at temperatures lower than $(n-1)^{-1}$, where n is the dimensionality of

temperature parameter for the 10-way categorical Concrete distribution was taken to be 0.1. For JointVAE, the initial latent capacities for both the continuous and discrete KL terms (C_z and C_c in in Equation 2.14, respectively) were taken to be 0, which were linearly increased to 5 over the whole training procedure. The capacity coefficient γ for FactorVAE was taken to be 30, which ensured that the KL was close to its target value at all times. The weight-decay coefficient for VAE with L2 regularization was chosen to be 0.5.

3.3 Attack methodology

To generate adversarial examples for the preceding models, we used the latent-attack methodology of Kos et. al. [47]. In particular, the optimization problem in Equation 3.1 is solved with the Adam optimizer for 1000 steps, with a learning rate of 0.001 and a relative weighting coefficient $\lambda = 1$. The inital starting point of the optimization procedure was constructed by adding zero-centered Gaussian noise with a standard deviation of 0.2 to the source image.

$$\begin{aligned} &\underset{\boldsymbol{x}_*}{\arg\min}\ \ \lambda\|\boldsymbol{x}_s - \boldsymbol{x}_*\|_2 + \|[\boldsymbol{\mu}_t, \boldsymbol{\alpha}_t]^T - [\boldsymbol{\mu}_*, \boldsymbol{\alpha}_*]^T\|_2 \\ &\text{subject to } \boldsymbol{x}_{\boldsymbol{*}} \in [0,1]^m, \end{aligned} \tag{3.1}$$

where $\boldsymbol{x}_s$ is the original m-dimensional source image, $\boldsymbol{\mu}_t$ represents the approximate posterior mean of the target image which we are trying to adversarially reconstruct, $\boldsymbol{\alpha}_x$ represents the logits of the target's latent Concrete distribution, and $\boldsymbol{\mu}_*$, $\boldsymbol{\alpha}_*$ represent the same quantities, but for the adversarial image $\boldsymbol{x}_*$.

Following Kos et. al. [47], a classifier, trained to recognize the types of MNIST digits, was used to derive two metrics for quantifying the effectiveness of the attacks. The model, whose arcitecture is outlined in Table 3.2, was trained with momentum-based stochastic gradient descent for 10 epochs on the MNIST training set with a batch size of 64. After training, it achieved a test set accuracy of 98%.

the Concrete variable, the probability simplex corresponding to the Concrete distribution is guaranteed not to have any modes in its interior. This gives us some confidence that our continuous relaxation is not too far from the corresponding discretized variable.

Classifier
$\boldsymbol{x} \in \mathbb{R}^{32\times 32}$ grayscale image
Conv 5×5. 10 ReLU, 1 stride
Maxpool 2×2. 2 stride
Conv 5×5. 10 ReLU, 1 stride
Maxpool 2×2. 2 stride
Dropout (p=0.5)
Dense. 50 ReLU
Dense. 10 softmax

Table 3.2: Architecture of a simple MNIST classifier.

The first metric, $AS_{\text{ignore-target}}$, measures the adversarial example's ability to fool the VAE to reconstruct an image that depicts a digit that is different from that of the source image. Under this metric, an attack is successful even if the reconstructed digit has a different class than the source image *and* the target image. Mathematically, for a single source-target pair, the metric is defined as follows:

$$AS_{\text{ignore-target}} = 1 - \mathbb{1}[f_c(f_{\text{vae}}(\boldsymbol{x}_*)), c(\boldsymbol{x}_s)],$$

where $\boldsymbol{x}_*$ is the adversarial example for the original source image $\boldsymbol{x}_s$, $f_{\text{vae}}(\boldsymbol{x})$ is the variational autoencoder outputting the natural parameters of the conditional distribution $p(\boldsymbol{x}|\boldsymbol{z})$ as the "reconstruction" of the input $\boldsymbol{x}$, $f_c(\cdot)$ is the digit classifier, $c(\boldsymbol{x})$ is the true class of $\boldsymbol{x}$, and $\mathbb{1}(x, y)$ is an indicator function outputting 1 is $x = y$ and 0 otherwise.

The second metric, AS_{target}, considers an attack successful only if the digit present in the target image is reconstructed. Thus, $AS_{\text{target}} \leq AS_{\text{ignore-target}}$ for any pair of images. In symbolic notation, the metric can be written as follows:

$$AS_{target} = \mathbb{1}[f_c(f_{\text{vae}}(\boldsymbol{x}_*)), c(\boldsymbol{x}_t)],$$

where $\boldsymbol{x}_t$ is the target image.

To calculate the values of the metrics over the whole dataset, the following scheme was utilized. First, suppose the MNIST test set is partitioned into 9 disjoint subsets, with subset S_j containing the images corresponding to class j, with $j = 0, 1, \ldots, 9$. For an image $\boldsymbol{x}$ belonging to class i in the MNIST test set, a random example $\boldsymbol{x}_j$ was drawn from each of the partitions S_j, $j \neq i$. The two success metrics were then measured by carrying out 9 adversarial attacks with $\boldsymbol{x}$ as source and $\boldsymbol{x}_j$ as targets. The process was repeated for every image in the test set, and results averaged. To account for the imperfectness of the neural-network based classifier, we omitted source images whose reconstruction was misclassified from our analysis. When reconstructing the adversarial examples, no sampling was performed in the latent bottleneck layer. Rather, we used the predicted mean of the multivariate normal and the $\arg\max$ over the predicted logits of the Concrete distribution as the latent vector that is propagated through the decoder.

Chapter 4

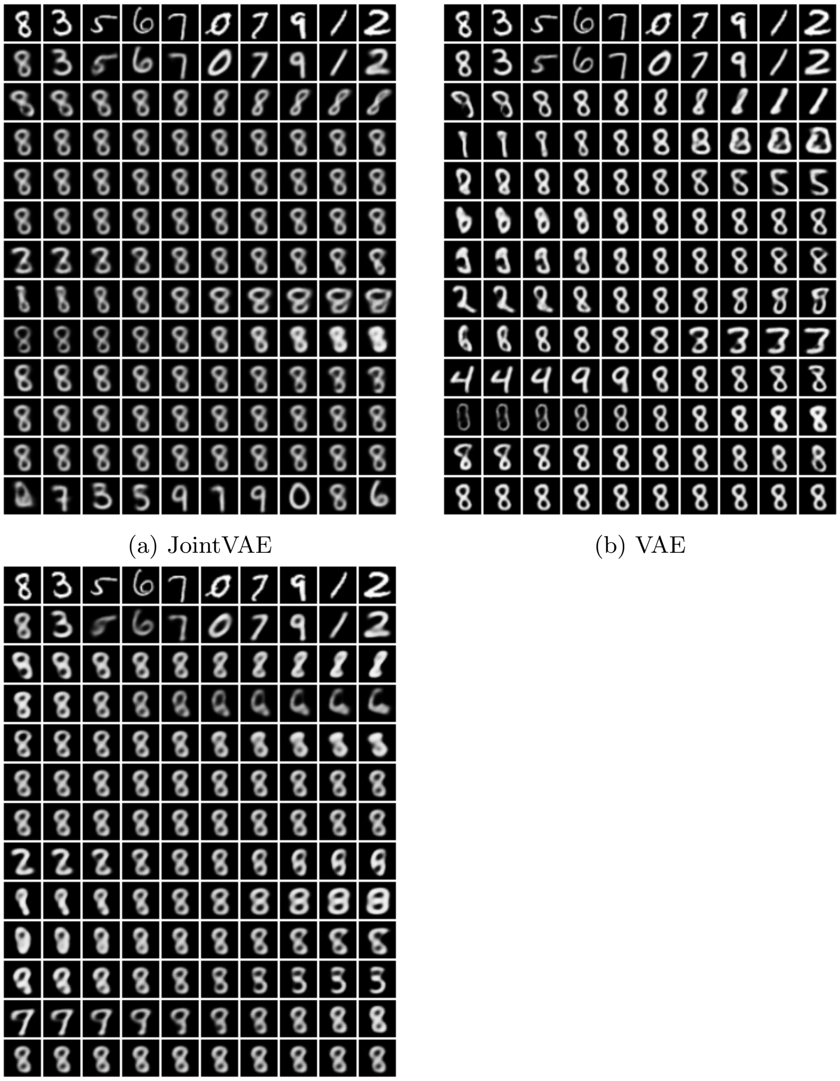
Results

Figure 4.1 illustrates the reconstructions and latent traversals of the trained disentangled and regular variational autoencoders. The reconstructions in the second row are generated by feeding the corresponding original image in first row into the encoder, obtaining the mean of the Gaussian and the logits of the Concrete distribution. Next, a one-hot vector is constructed, in which the bit that is set is the one whose index is equal to the $\arg\max$ over the logits. Next, the mean of the normal distribution and the one-hot vector are concatenated, and fed into the decoder. The traversal of the i-th continuous latent is generated by feeding the top-left image in each of the figures into the encoder, obtaining the mean and one-hot vector as before. However, now the i-th unit [1] of this vector is manually set to values linearly spaced in the range $[-3, 3]$ in the continuous case, or to integer from 0 to 9 in the discrete case.

First, as expected, we see that the quality of reconstructions of the regular VAE is superior to that of the regularized VAEs. The reconstruction qualities of JointVAE and VAE with weight decay are roughly on the same level. Second, careful inspection of Figure 4.1a reveals that JointVAE clearly disentangles 4 major factors of variation in MNIST digits: angle (3rd row), width (8th row), thickness (9th row), and digit class (last row). From Figures 4.1b and 4.1c we see that VAEs whose latent bottleneck are largely unable to completely disentangle any factors of variation. Most notably, the last row for both of these models does not exhibit any variation, meaning that the categorical Concrete

[1] Note that in this context, we consider the 10-dimensional one-hot vector as single unit.

variable that encoded digit type in the JointVAE case is left unutilized.



(a) JointVAE

(b) VAE

(c) VAE L2

Figure 4.1: Reconstructions and latent traverals of various VAEs. The first row depicts 10 randomly sampled images from the MNIST test set, while the second row visualizes their reconstructions. The remaining 11 rows depict the latent traversals of each unit in the bottleneck layer, with each of the first 10 rows correspdonding to one of the dimensions of the multivariate normal distribution, and the last row corresponding to the categorical random variable.

Table 4 gives the results of adversarial attacks for the 3 models, averaged over the whole training set. First, we see that targeted attacks are much less successful than untargeted ones. This is expected, because some digit types, like 3, 8, and 9, are quite similar, resulting in phenomenon where the reconstruction of an adversarial example is not of the same type as the target, but still belongs to a visually closely related class of digits. For example, the adversarial image of 9 in first column of row 3 in Figure 4.2 had a target of 8, but its reconstruction strongly resembles a 5 or a 3 instead.

Second, and most importatly, the table reveals that the the disentangled variational autoencoder is significantly less robust to adversarial attacks than entangled models. This observation holds for both untargeted, as well as for target attacks, with JointVAE being roughly 24% easier to fool for untargeted attacks, and 32% for targeted, as compared to regular VAE. The effect can not be explained by the inferior reconstruction ability of JointVAE either, as VAE with weight decay, whose reconstructions are visually comparable to these of JointVAE, is even more robust to adversarial attacks than VAE. Indeed, the average success rate of untargeted attacks for the disentangled model is approximately 37% higher than for the entangled model regularized with weight decay, with the number increasing to 59.7% for targeted attacks.

	AS ignore-target	**AS target**
JointVAE	36.86%	12.54%
VAE	29.83%	9.49%
VAE L2	26.88%	7.85%

Table 4.1: The results of adversarial attacks for JointVAE, regular VAE, and VAE regularized with an L2 norm (i.e. weight decay), averaged over the entire MNIST test set.

Tables 4.2 and 4.3 show the average marginals of the full attack matrices, while the complete matrices themselves, which outline the effectiveness of the attacks for every source-target pair, are given in the Appendix. In both tables, for each digit, the cell highligted in gray corresponds to the model which was easiest to fool, as measured by the untargeted attack success metric $AS_{\text{ignore-target}}$. We see that for the vast majority of digit classes, the disentangled models are most prone to be fooled by adversarial examples. A notable exception is the case when the source image depicting a 4 is perturbed to any other class, in which case JointVAE substantially outperforms (as in, is harder to fool) the other models in terms of the $AS_{\text{ignore-target}}$ metric.

Source	0	1	2	3	4	5	6	7	8	9
Disentangled	8.89% (3.18%)	82.62% (33.76%)	20.92% (5.36%)	16.72% (6.20%)	33.73% (13.02%)	44.25% (11.24%)	38.49% (13.48%)	40.57% (11.65%)	18.23% (8.47%)	64.23% (18.97%)
Entangled	6.12% (1.93%)	63.75% (28.58%)	11.12% (3.06%)	12.11% (4.30%)	64.40% (13.46%)	37.79% (11.04%)	27.83% (9.55%)	39.70% (7.89%)	16.27% (6.43%)	19.19% (8.70%)
Entangled wdec	3.30% (0.97%)	59.78% (23.44%)	7.09% (1.41%)	10.83% (3.02%)	53.99% (10.94%)	34.74% (8.65%)	24.94% (6.10%)	18.45% (4.26%)	23.46% (7.61%)	32.28% (12.09%)

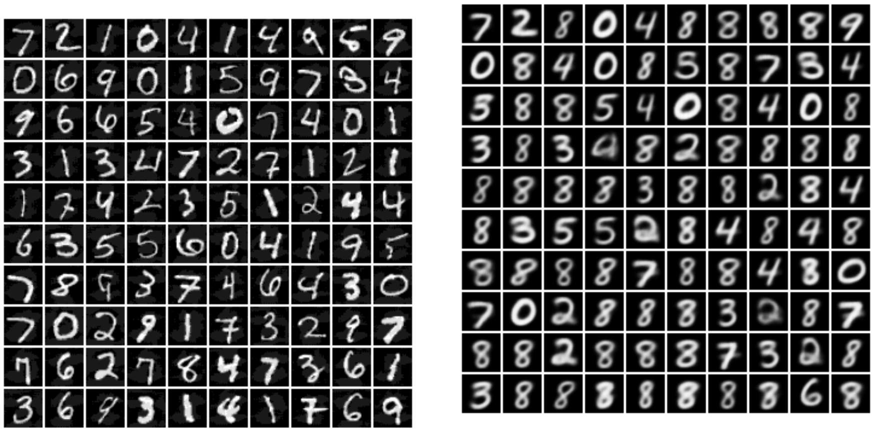
Table 4.2: Average source marginals of the full attack success matrices. In each cell, $AS_{\text{ignore-target}}$ is written above, while AS_{target} is indicated below in brackets.

Target	0	1	2	3	4	5	6	7	8	9
Disentangled	40.36% (1.78%)	25.91% (0.93%)	46.50% (25.91%)	41.81% (10.19%)	31.76% (14.23%)	38.98% (10.04%)	37.46% (7.80%)	27.77% (7.42%)	46.23% (32.19%)	31.87% (14.87%)
Entangled	29.39% (3.52%)	13.88% (0.24%)	35.42% (11.50%)	40.39% (18.68%)	24.71% (5.53%)	32.75% (7.38%)	22.94% (4.23%)	28.20% (4.54%)	43.49% (25.44%)	27.11% (13.87%)
Entangled wdec	28.36% (4.50%)	12.62% (1.06%)	37.50% (16.60%)	36.40% (12.30%)	19.14% (1.88%)	27.39% (4.44%)	20.63% (3.89%)	29.92% (12.57%)	33.21% (12.55%)	23.67% (8.72%)

Table 4.3: Average target marginals of the full attack success matrices. In each cell, $AS_{\text{ignore-target}}$ is written above, while AS_{target} is indicated below in brackets.

Figures 4.2 nad 4.3 illustrates the adversarial examples for JointVAE and VAE, respectively. The adversarial inputs are generated with the first 100 images of the MNIST test set taken as sources, and the top-left image of the digit 8 in Figure 4.1a considerd as target. Although not explicitly shown here, we validated that the adversarial examples shown in Figures 4.2a and 4.3a are visually almost indistinguishable from the corresponding original images from which the adversarial input is derived.

For both models, we can observe a variety of successful attacks, in the sense that the reconstructed adversarial input is of different class than the source image. Interestingly, it can be observed that in some cases, even though the digit



(a) Adversarial examples

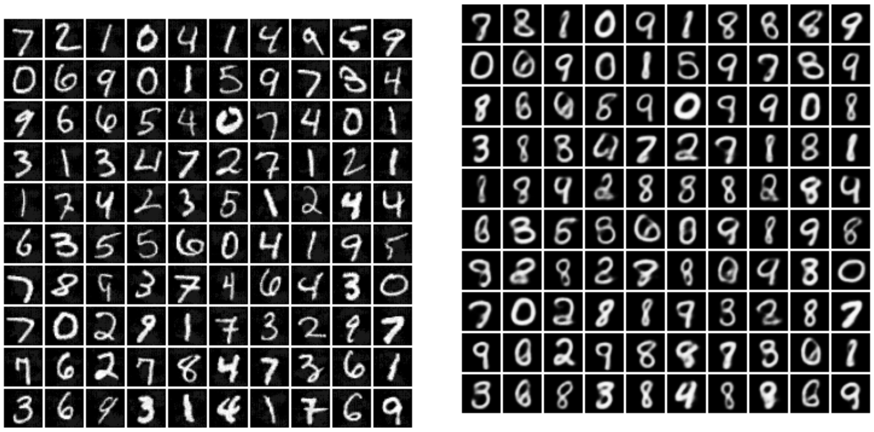
(b) Reconstructions of the adversarial examples

Figure 4.2: Adversarial examples for JointVAE.

class changes shape, other factors of variation stay more similar to the source image than to that of the target. For example, consider the adversarial images in the 9-th column, in rows 4 and 5. For both models, we see that the relatively slim 2 is reconstructed as an 8 that falls to the slimmer side as well, while the reconstruction of the thicker 4 is reconstructed as a thick 8. This qualitative observations could indicate that the target's digit class might be an easier factor of variation to capture for the adversarial example than some other, continuous properties of digits.

Another noteworthy observation worth highlighting is that while JointVAE's reconstructions of the adversarial examples are more blurry, the majority of them look much more natural and smooth, as compared to the reconstructions of regular VAE. As an example, compare the images in the first column of row 6 in Figures 4.2b and 4.3b. This could be attributed to the fact that constraining the latent bottleneck gives rise to enhanced generalizability, which manifests when one reconstructs latent vectors sampled from esoteric regions in the latent space.

From Figure 4.1b, we see that regular VAE is prone to produce uninterpretable reconstructions at extreme values of latent units that were likely not seen during training (corresponding to images near the left and right side of the figure), regular VAE is prone to produce uninterpretable reconstructions. This is most



(a) Adversarial examples

(b) Reconstructions of the adversarial examples

Figure 4.3: Adversarial examples for VAE.

clearly illustrated by the 4-th row of said figure, where a clear reconstruction is achieved only by a small set of values near the mean of the distribution. On the other hand, a disentangled model generalized gracefully even to out-of-sample latent values, as can be observed from Figure 4.1a. Connecting this effect back to the context of adversarial examples, the fact that some of the reconstructions of adversarial inputs by VAE look unnatural indicates that while optimizing Equation 3.1, one occasionally arrives at a solution whose encoded representation is far from what is usually seen in the training set.

Chapter 5

Conclusion

5.1 Summary

The thesis compared the effect of adversarial attacks to disentangled and entangled variational autoencoders on the MNIST handwritten images dataset. It was found that models which are more strongly regularized to have a factorial internal representation are less robust to adversarial perturbations than their entangled counterparts. However, the reconstructions of the adversarial examples for disentangled VAEs were qualitatively more natural-looking than these of regular VAE.

The author of this thesis was responsible for proposing the research direction, implementing the models in code, carrying out the subsequent analysis, and interpreting the results.

5.2 Future work and discussion

One drawback of the quantitative evaluation approach taken in this thesis is that our metric of attack success depended only on one of the factors of variation in the dataset, the digit class. This restriction was mainly due to our choice of dataset, as the type of digit is the only factor of variation labelled in MNIST. Future work should extend our analysis to more diverse datasets with more labelled attributes, like CelebA. This, however, is complicated by the fact that vanilla VAE architectures produce results of poor quality on such rich datasets. Thus, it might be necessary to employ structural modifications to the original model that was described in Section 2.3. For example, one might use VAE-GAN [51] or DFC-VAE [40] to achieve better reconstructions.

An additional avenue worth investigating is the use of more recently proposed VAE-based regularization schemes in the disentanglement literature. For example, Qi et. al. [16] identify that the KL-divergence in the latent bottleneck decomposes into 3 terms. They found that putting more weight onto only one of the terms, the total correlation between the latents, yields state-of-the art results for disentanglement. It is interesting to examine which of the identified terms are most responsible for the observed decrease in robustness to adversarial attacks. Relatedly, we did carry out some preliminary experiments with FactorVAE [44], which is a model that also minimizes the same total correlation term. However, we found it not to be able to encode digit type into the Concrete latent on the MNIST dataset.

Finally, a more thorough understanding of deep probabilistic latent variable models would help shed light to some of the conundrums stemming from our results. First, the question as to why constraining the latent bottleneck in a classification setting diminishes the effectiveness of adversarial inputs, but increases it in a generative setting is left open. Second, the intuiton laid out by Burgess et. al. [36], positing that β-VAE encodes nearby points in pixel space close together in latent space, is not supported by our observations.

Appendix

Tables 5.1, 5.2 and 5.3 give the attack success rates for all possible pairs of MNIST digits, for JointVAE, VAE, and VAE with weight decay, respectively. For all tables, In each cell, $AS_{\text{ignore-target}}$ is written above, while AS_{target} is indicated below in brackets.

Table 5.1: Adversarial attack successes for JointVAE.

Source	**Targets**									
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	-	6.58% (0.00%)	11.12% (5.98%)	8.36% (3.34%)	9.21% (1.99%)	11.15% (6.44%)	8.13% (4.71%)	7.60% (0.31%)	8.74% (5.89%)	9.10% (0.00%)
1	99.48% (0.00%)	-	97.62% (93.64%)	95.63% (4.98%)	73.25% (42.63%)	95.49% (10.87%)	97.64% (26.85%)	23.43% (13.36%)	95.48% (72.58%)	65.52% (38.95%)
2	7.74% (0.66%)	24.84% (0.42%)	-	14.63% (4.33%)	27.98% (4.24%)	23.36% (1.58%)	6.64% (2.29%)	26.38% (12.29%)	26.85% (22.36%)	29.84% (0.11%)
3	2.90% (0.58%)	20.00% (1.14%)	16.61% (11.86%)	-	19.27% (0.80%)	17.33% (14.38%)	12.00% (1.43%)	19.86% (1.47%)	24.27% (19.49%)	18.22% (4.61%)
4	24.70% (0.85%)	44.48% (0.48%)	37.18% (25.28%)	31.27% (8.15%)	-	21.63% (4.27%)	21.31% (11.65%)	51.10% (6.48%)	34.71% (26.55%)	37.16% (33.50%)
5	46.66% (3.54%)	45.71% (0.65%)	67.40% (8.54%)	34.48% (26.92%)	31.63% (8.79%)	-	51.50% (12.26%)	42.93% (2.36%)	46.55% (30.24%)	31.36% (7.87%)
6	28.87% (7.41%)	34.20% (2.72%)	39.89% (33.11%)	39.65% (6.35%)	38.06% (24.21%)	33.49% (10.85%)	-	38.18% (2.74%)	50.00% (27.84%)	44.10% (6.11%)
7	47.31% (1.43%)	4.44% (0.23%)	40.25% (24.94%)	47.69% (2.08%)	38.66% (4.94%)	52.41% (6.63%)	41.66% (0.37%)	-	49.03% (27.36%)	43.71% (36.91%)
8	25.11% (0.79%)	13.13% (1.71%)	30.48% (24.20%)	24.06% (14.71%)	11.16% (5.69%)	18.67% (12.36%)	22.80% (6.02%)	10.87% (5.03%)	-	7.79% (5.73%)
9	80.48% (0.80%)	39.78% (1.01%)	77.94% (5.65%)	80.51% (20.82%)	36.64% (34.79%)	77.28% (22.97%)	75.44% (4.60%)	29.54% (22.74%)	80.48% (57.39%)	-

Table 5.2: Adversarial attack successes for VAE.

Source	**Targets**									
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	-	3.46% (0.00%)	8.93% (2.60%)	7.47% (1.89%)	6.85% (0.31%)	5.59% (1.60%)	5.74% (2.34%)	4.58% (0.42%)	8.82% (6.51%)	3.65% (1.67%)
1	58.13% (0.21%)	-	76.68% (52.17%)	69.11% (41.26%)	60.94% (24.38%)	66.59% (17.45%)	50.53% (19.64%)	64.42% (28.05%)	73.72% (55.29%)	53.65% (18.76%)
2	13.13% (1.89%)	4.53% (0.10%)	-	13.32% (9.73%)	9.25% (0.54%)	14.37% (0.36%)	11.17% (1.02%)	7.71% (1.04%)	16.65% (12.13%)	9.94% (0.74%)
3	8.26% (1.00%)	4.71% (0.11%)	6.93% (4.01%)	-	20.51% (0.11%)	14.46% (10.78%)	12.30% (0.68%)	4.75% (0.22%)	26.07% (19.89%)	11.00% (1.92%)
4	65.87% (3.26%)	42.55% (0.65%)	66.48% (9.88%)	89.27% (10.51%)	-	79.10% (4.01%)	26.20% (5.15%)	71.54% (3.79%)	79.87% (30.09%)	58.75% (53.80%)
5	37.52% (4.06%)	23.74% (0.12%)	49.51% (6.25%)	40.81% (30.51%)	38.44% (0.49%)	-	26.43% (6.82%)	33.75% (0.62%)	55.09% (42.21%)	34.78% (8.29%)
6	25.22% (17.32%)	6.80% (0.44%)	29.56% (8.57%)	41.88% (9.42%)	14.19% (6.22%)	27.86% (15.47%)	-	38.40% (0.33%)	42.39% (25.33%)	24.19% (2.82%)
7	33.82% (1.12%)	24.40% (0.21%)	30.49% (8.33%)	28.37% (12.51%)	50.22% (6.47%)	40.85% (1.09%)	55.27% (0.11%)	-	56.51% (9.71%)	37.35% (31.43%)
8	12.83% (1.03%)	7.80% (0.00%)	21.08% (9.91%)	36.89% (30.99%)	8.67% (0.57%)	21.05% (6.97%)	10.69% (2.21%)	16.72% (0.80%)	-	10.72% (5.36%)
9	9.74% (1.79%)	6.89% (0.54%)	29.15% (1.82%)	36.42% (21.34%)	13.30% (10.64%)	24.85% (8.65%)	8.11% (0.11%)	11.96% (5.60%)	32.30% (27.78%)	-

Table 5.3: Adversarial attack successes for VAE with weight decay.

Source	**Targets**									
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	-	2.85% (0.00%)	6.19% (1.47%)	3.70% (1.37%)	1.99% (0.00%)	4.59% (1.38%)	2.78% (1.71%)	2.43% (0.32%)	3.41% (2.13%)	1.78% (0.31%)
1	68.30% (1.36%)	-	73.49% (59.25%)	67.68% (25.63%)	39.62% (3.75%)	57.65% (10.70%)	38.69% (16.51%)	61.00% (47.46%)	76.51% (35.59%)	55.09% (10.69%)
2	9.64% (2.15%)	4.82% (0.21%)	-	6.37% (3.08%)	6.12% (0.91%)	8.84% (0.25%)	6.18% (0.93%)	7.40% (1.39%)	6.90% (2.94%)	7.53% (0.87%)
3	9.82% (1.75%)	5.70% (0.35%)	10.50% (6.96%)	-	14.30% (0.00%)	12.83% (8.60%)	14.10% (0.96%)	8.55% (2.66%)	9.54% (4.05%)	12.09% (1.86%)
4	55.38% (2.22%)	30.42% (2.12%)	58.08% (17.51%)	72.58% (4.84%)	-	70.14% (1.37%)	27.46% (4.53%)	56.47% (12.21%)	66.54% (10.78%)	48.83% (42.91%)
5	38.78% (12.70%)	21.45% (1.37%)	43.55% (3.36%)	34.33% (19.27%)	40.65% (0.27%)	-	32.88% (7.37%)	32.69% (5.75%)	34.68% (20.38%)	33.61% (7.41%)
6	17.76% (12.83%)	8.63% (1.55%)	35.99% (11.12%)	33.48% (3.71%)	16.85% (5.03%)	18.91% (5.44%)	-	30.75% (0.66%)	37.07% (12.98%)	25.00% (1.54%)
7	13.38% (1.70%)	9.21% (1.41%)	22.97% (12.09%)	19.82% (9.03%)	19.93% (0.68%)	18.98% (1.36%)	21.93% (0.00%)	-	26.29% (3.21%)	13.52% (8.83%)
8	19.32% (3.26%)	16.40% (1.59%)	37.96% (26.31%)	34.68% (20.08%)	19.24% (0.38%)	16.34% (4.40%)	21.80% (2.42%)	29.87% (6.03%)	-	15.54% (4.05%)
9	22.89% (2.49%)	14.12% (0.92%)	48.74% (11.30%)	54.99% (23.67%)	13.57% (5.88%)	38.27% (6.44%)	19.83% (0.61%)	40.14% (36.64%)	37.96% (20.88%)	-

Bibliography

[1] Alessandro Achille, Tom Eccles, Loic Matthey, Christopher P. Burgess, Nick Watters, Alexander Lerchner, and Irina Higgins. Life-long disentangled representation learning with cross-domain latent homologies, 2018.

[2] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. Connectionist models and their implications: Readings from cognitive science. chapter A Learning Algorithm for Boltzmann Machines, pages 285–307. Ablex Publishing Corp., Norwood, NJ, USA, 1988.

[3] Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck. *CoRR*, abs/1612.00410, 2016.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[5] Vahid Behzadan and Arslan Munir. Vulnerability of deep reinforcement learning to policy induction attacks. *CoRR*, abs/1701.04143, 2017.

[6] Yoshua Bengio. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2(1):1–127, January 2009.

[7] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR*, abs/1206.5538, 2012.

[8] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 41–48, New York, NY, USA, 2009. ACM.

[9] David Berthelot, Tom Schumm, and Luke Metz. BEGAN: boundary equilibrium generative adversarial networks. *CoRR*, abs/1703.10717, 2017.

[10] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[11] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016.

[12] D.S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

[13] Christopher P. Burgess, Irina Higgins, Arka Pal, Loïc Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in β-vae. *CoRR*, abs/1804.03599, 2018.

[14] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. *CoRR*, abs/1608.04644, 2016.

[15] Nicholas Carlini and David A. Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. *CoRR*, abs/1801.01944, 2018.

[16] Tian Qi Chen, Xuechen Li, Roger B. Grosse, and David K. Duvenaud. Isolating sources of disentanglement in variational autoencoders. *CoRR*, abs/1802.04942, 2018.

[17] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *CoRR*, abs/1606.03657, 2016.

[18] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec 1989.

[19] Peter Dayan, Geoffrey E. Hinton, Radford M. Neal, and Richard S. Zemel. The helmholtz machine, 1995.

[20] Rodrigo de Bem, Arnab Ghosh, Thalaiyasingam Ajanthan, Ondrej Miksik, N. Siddharth, and Philip H. S. Torr. Dgpose: Disentangled semi-supervised deep generative models for human body analysis. *CoRR*, abs/1804.06364, 2018.

[21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[22] Pratik Dubal, Rohan Mahadev, Suraj Kothawade, Kunal Dargan, and Rishabh Iyer. Deployment of customized deep learning based video analytics on surveillance cameras. *CoRR*, abs/1805.10604, 2018.

[23] Emilien Dupont. Learning disentangled joint continuous and discrete representations, 2018.

[24] Javid Ebrahimi, Daniel Lowd, and Dejing Dou. On adversarial examples for character-level neural machine translation. 2018.

[25] Gamaleldin F. Elsayed, Shreya Shankar, Brian Cheung, Nicolas Papernot, Alex Kurakin, Ian J. Goodfellow, and Jascha Sohl-Dickstein. Adversarial examples that fool both human and computer vision. *CoRR*, abs/1802.08195, 2018.

[26] Yaxiang Fan, Gongjian Wen, Deren Li, ShaoHua Qiu, and Martin D. Levine. Video anomaly detection and localization via gaussian mixture fully convolutional variational autoencoder. *CoRR*, abs/1805.11223, 2018.

[27] Zoubin Ghahramani and Michael I. Jordan. Supervised learning from incomplete data via an em approach. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*, pages 120–127. Morgan-Kaufmann, 1994.

[28] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[29] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudk, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[30] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence,

and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[31] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[32] Ian J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *CoRR*, abs/1701.00160, 2017.

[33] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. Adversarial examples for malware detection. In Simon N. Foley, Dieter Gollmann, and Einar Snekkenes, editors, *Computer Security – ESORICS 2017*, pages 62–79, Cham, 2017. Springer International Publishing.

[34] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5767–5777. Curran Associates, Inc., 2017.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[36] Irina Higgins, Loïc Matthey, Xavier Glorot, Arka Pal, Benigno Uria, Charles Blundell, Shakir Mohamed, and Alexander Lerchner. Early visual concept learning with unsupervised deep learning. *CoRR*, abs/1606.05579, 2016.

[37] Irina Higgins, Arka Pal, Andrei A. Rusu, Loic Matthey, Christopher P Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning, 2017.

[38] Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. Transforming auto-encoders. In *Proceedings of the 21th International Conference on Artificial Neural Networks - Volume Part I*, ICANN'11, pages 44–51, Berlin, Heidelberg, 2011. Springer-Verlag.

[39] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[40] Xianxu Hou, LinLin Shen, Ke Sun, and Guoping Qiu. Deep feature consistent variational autoencoder. *CoRR*, abs/1610.00291, 2016.

[41] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *CoRR*, abs/1611.01144, 2016.

[42] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[43] Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2649–2658, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[44] Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2649–2658, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[45] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[46] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

[47] Jernej Kos, Ian Fischer, and Dawn Song. Adversarial examples for generative models, 2017.

[48] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[49] Tejas D. Kulkarni, Will Whitney, Pushmeet Kohli, and Joshua B. Tenenbaum. Deep convolutional inverse graphics network. *CoRR*, abs/1503.03167, 2015.

[50] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016.

[51] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *CoRR*, abs/1512.09300, 2015.

[52] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[53] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. *CoRR*, abs/1411.7766, 2014.

[54] R. Duncan Luce. *Individual Choice Behavior: A Theoretical analysis.* Wiley, New York, NY, USA, 1959.

[55] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *CoRR*, abs/1611.00712, 2016.

[56] Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

[57] Ari S. Morcos, David G.T. Barrett, Neil C. Rabinowitz, and Matthew Botvinick. On the importance of single directions for generalization. In *International Conference on Learning Representations*, 2018.

[58] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization.* Springer, New York, NY, USA, second edition, 2006.

[59] Daniele Palossi, Antonio Loquercio, Francesco Conti, Eric Flamand, Davide Scaramuzza, and Luca Benini. Ultra low power deep-learning-powered autonomous nano drones. *CoRR*, abs/1805.01831, 2018.

[60] Scott Reed, Kihyuk Sohn, Yuting Zhang, and Honglak Lee. Learning to disentangle factors of variation with manifold interaction. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1431–1439, Bejing, China, 22–24 Jun 2014. PMLR.

[61] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286, Bejing, China, 22–24 Jun 2014. PMLR.

[62] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A hierarchical latent vector model for learning long-term structure in music. *CoRR*, abs/1803.05428, 2018.

[63] Sara Sabour, Yanshuai Cao, Fartash Faghri, and David J. Fleet. Adversarial manipulation of deep representations. *CoRR*, abs/1511.05122, 2015.

[64] Maneesh Sahani. Beyond linear-gaussian models and mixtures. University lecture, 2018.

[65] Jrgen Schmidhuber. Learning factorial codes by predictability minimization. *Neural Computation*, 4(6):863–879, 1992.

[66] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.

[67] Pedro Tabacof, Julia Tavares, and Eduardo Valle. Adversarial images for variational autoencoders. *CoRR*, abs/1612.00155, 2016.

[68] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[69] Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. pages 368–377, 1999.

[70] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, May 1992.

[71] Jimei Yang, Scott E. Reed, Ming-Hsuan Yang, and Honglak Lee. Weakly-supervised disentangling with recurrent transformations for 3d view synthesis. *CoRR*, abs/1601.00706, 2016.