



Investigating the Robustness of Disentangled Variational Autoencoders to Adversarial Examples

Sten Sootla¹

Computational Statistics and Machine Learning

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

Submission date: 10.08.2018

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

Summarise your report concisely.

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 2 |
| 2 | Related work and background | 4 |
| 2.1 | Probabilistic learning and inference | 4 |
| 2.1.1 | Discriminative approach | 4 |
| 2.1.2 | Generative approach | 5 |
| 2.2 | Variational autoencoders | 8 |
| 2.3 | Disentangled representations | 9 |
| 2.4 | Adversarial examples | 9 |
| 3 | Methods | 10 |
| 4 | Results | 11 |
| 5 | Conclusion | 12 |
| 6 | Appendix | 13 |

Chapter 1

Introduction

In recent years, deep neural networks have shown impressive results in various supervised pattern recognition tasks, including image classification [18, 15], object detection [12], and machine translation [2]. 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 toy datasets to being able to capture exceedingly complex real-world distributions [4, 22]. One of the most prominent generative models behind these successes is the adversarial autoencoder [17], 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 models, which mainly consist 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 arbitrarily complex functions [7]. 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-intuitive properties [23].

First, the resulting decision boundaries of the trained networks 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 fool the model with high confidence [23]. 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. More practically, as the models under discussion are being pushed into production for use in semi-autonomous vehicles [6], drones [21], surveillance cameras [9] and malware detection pipelines [14], the existence of adversarial examples 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, giving them a reputation of being completely black-box learning machines. Indeed, Szegedy et. al. [23] have shown that directions along the coordinate axes are often no more

semantically meaningful than other directions in the latent space, while Morcos et. al. [20] 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 semantically meaningful factor of variation in the input is explained by a combination of units, as opposed to a single neuron.

It has been argued [3] 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 configuration of concepts that was 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 [16] and semi-supervised learning [8].

This thesis investigates the possible relationship between the two intriguing properties of neural networks outlined above: the counter-intuitive decision boundaries 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 the leading conjecture explaining the presence of adversarial examples posits that they come about due to excessive linearity of today's pattern recognizers [13]. However, models whose internal representations are disentangled are arguably less linear, since they are implicitly guided to capture the main generative factors of the data, which in datasets with enough complexity are non-linear with respect to inputs in the pixel space. This suggests that disentangled models could be less prone to be fooled by adversarial examples.

To test the preceding hypothesis, we empirically evaluated the robustness of state-of-the-art disentanglement-inducing VAEs to adversarial examples ¹, and compared the results to regular, entangled VAEs. The analyses were carried out for the MNIST handwritten images dataset, as well as for CelebA, a large-scale dataset of celebrity faces. Surprisingly, it was found that disentangled generative models are actually *less* robust to attacks on both datasets, contradicting our initial rationale. This stands in stark contrast to previous work by Alemi et. al. [1], which performed similar analysis for supervised models, but arrived at the opposite conclusion, highlighting a fundamental difference between generative and discriminative models when it comes to adversarial examples.

The thesis is organized as follows. In chapter 1, a sufficiently in-depth overview of variational autoencoders, disentanglement, and adversarial examples is given. Chapter 2 outlines the experimental methodology, and provides exact details of the analyses done in this thesis for reproducibility. Chapter 3 gives the results of the experiments.

¹In the context of generative models, an adversarial example is a datapoint which is very similar to the original, untampered image, but whose reconstruction differs markedly from the original's. As an example, 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.

Chapter 2

Related work and background

This chapter introduces the preliminary topics that are integral to understanding the methods and results of this thesis. In addition, it provides an overview of the most important previous works that are closely related to ours. However, the reader is assumed to be comfortable with the basics of discriminative methods in machine learning. The first section of this chapter familiarizes the reader with a probabilistic approach to pattern recognition. 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

To fully appreciate variational autoencoders 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 follows.

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 [5], of how viewing machine learning through a probabilistic 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 $\mathbf{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(n)}\}$, 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. a neural network):

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

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:

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

where the last equation 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 [10], fill out missing dimensions of incomplete datapoints [11], or sample new, realistic datapoints from the learnt density function [19].

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

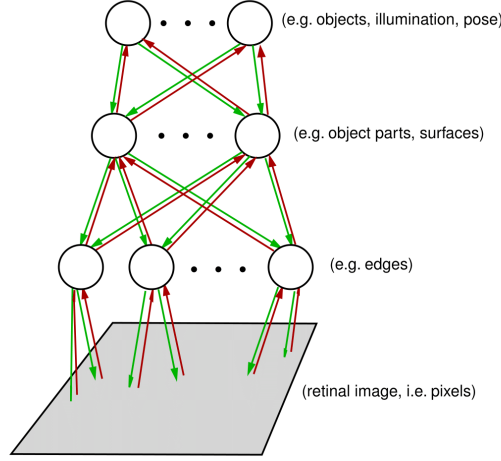


Figure 2.1: Illustration of a probabilistic model with latent variables. Green arrows represent the generative process, in which higher-level concepts guide the development of lower-level ones, eventually leading to the generation of each pixel in an image. Red arrows depict inference, in which case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

the probabilistic model 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 actual orientations of the edges might be decided upon, after which the actual image could be drawn out onto the canvas. Moreover, having imposed such hierarchical dependency 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.

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 observable random variable \mathbf{x} , an unobserved latent \mathbf{z} , and their joint distribution $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$, where the vector $\boldsymbol{\theta}$ denotes the parameters of the generative model. Then, using the sum rule of probability, we can express the marginal of \mathbf{x} as $p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) d\mathbf{z}$.

Naturally, as we are unable to evaluate the likelihood function, explicit optimization of its monotonic transform, the log-likelihood, presents difficulties as well. Fortunately, there exists a workaround,

which entails optimizing a lower-bound of the log-likelihood instead. By introducing a new distribution $q_\phi(\mathbf{z}|\mathbf{x})$, parameterized by ϕ , into our framework, we can lower-bound the original likelihood as follows:

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

To see that this lower bound could indeed make the optimization problem easier, it helps to write it in two different forms. The first form illustrates that the lower bound 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:

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

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 which 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:

$$\begin{aligned}
& \mathcal{L}(\mathbf{x}; \theta, \phi) = \\
& \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{z}|\mathbf{x}) p_\theta(\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} d\mathbf{z} = \\
& \int q_\phi(\mathbf{z}|\mathbf{x}) \log p_\theta(\mathbf{x}) + \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} d\mathbf{z} = \\
& \log p_\theta(\mathbf{x}) - \text{KL}(q_\phi(\mathbf{z}|\mathbf{x}) || p_\theta(\mathbf{z}|\mathbf{x})).
\end{aligned} \tag{2.3}$$

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 θ , 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 θ and ϕ , leading us to the well-known Expectation-Maximization (EM) algorithm [11]. This 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 θ .

What makes EM particularly appealing is that each full step of the algorithm either increases the log-likelihood, or the algorithm has converged to the latter's local maximum. From equation 2.3, we see that free energy is maximized with respect to ϕ when the KL vanishes (as it is always non-negative), which is accomplished when the approximate posterior ¹is set equal to the true posterior. After that, the lower bound is equal to the log-likelihood (under the current best-guess parameters). Since the next step, maximization with respect to θ can not decrease the free energy, we observe from equation 2.3 that the log-likelihood can also not decrease, 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. However, often we are 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 solution would be to optimize the bound directly 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.

2.2 Variational autoencoders

As already alluded to in the previous section, the variational autoencoder, introduced by Kingma and Welling [17], is a model that optimizes the lower bound of the log-likelihood (equation 2.1) using a gradient-based optimization scheme. The form of the lower bound that it is explicitly working with is as follows:

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

¹Note that 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. $p(\mathbf{x}|\mathbf{z})$ as opposed to $q(\mathbf{x}|\mathbf{z})$.

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, for common distributions, it can be calculated analytically. In general, however, this term can also be estimated by a finite set of samples as well.

An immediate problem that presents itself 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 ϕ for optimization. In principle, the necessary gradients could be estimated using the score function estimator, an ubiquitous method in the reinforcement learning literature [24]:

$$\begin{aligned}
& \nabla_{\phi} \mathbb{E}_{\mathbf{z} \sim q_{\phi}(\mathbf{z}|\mathbf{x})} [\log p_{\theta}(\mathbf{x}|\mathbf{z})] = \\
& \int \nabla_{\phi} q_{\phi}(\mathbf{z}|\mathbf{x}) \log p_{\theta}(\mathbf{x}|\mathbf{z}) d\mathbf{z} \stackrel{\text{multiply by identity}}{=} \\
& \int \frac{q_{\phi}(\mathbf{z}|\mathbf{x})}{q_{\phi}(\mathbf{z}|\mathbf{x})} \nabla_{\phi} q_{\phi}(\mathbf{z}|\mathbf{x}) \log p_{\theta}(\mathbf{x}|\mathbf{z}) d\mathbf{z} \stackrel{\text{derivative of log}}{=} \\
& \int q_{\phi}(\mathbf{z}|\mathbf{x}) \nabla_{\phi} \log q_{\phi}(\mathbf{z}|\mathbf{x}) \log p_{\theta}(\mathbf{x}|\mathbf{z}) d\mathbf{z} = \\
& \mathbb{E}_{\mathbf{z} \sim q_{\phi}(\mathbf{z}|\mathbf{x})} [\nabla_{\phi} \log q_{\phi}(\mathbf{z}|\mathbf{x}) \log p_{\theta}(\mathbf{x}|\mathbf{z})] \simeq \\
& \frac{1}{L} \sum_{l=1}^L \nabla_{\phi} \log q_{\phi}(\mathbf{z}|\mathbf{x}) \log p_{\theta}(\mathbf{x}|\mathbf{z}), \quad \text{where } \mathbf{z}^{(l)} \sim q_{\phi}(\mathbf{z}^{(l)}|\mathbf{x}).
\end{aligned} \tag{2.5}$$

Unfortunately, due to its generality, the preceding estimator has a very high variance, rendering it impractical for the purposes of generative modelling.

Talk about the reparameterization trick

2.3 Disentangled representations

2.4 Adversarial examples

Chapter 3

Methods

Chapter 4

Results

| Source | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|------------------|-------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| AS ignore-target | 8.80% | 82.58% | 20.46% | 16.52% | 32.74% | 43.58% | 38.25% | 40.10% | 18.68% | 64.22% |
| AS target | 3.25% | 34.05% | 5.59% | 6.15% | 12.26% | 10.88% | 13.21% | 11.85% | 8.52% | 19.30% |

Table 4.1: Disentangled marginals

| Source | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|------------------|-------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| AS ignore-target | 5.87% | 63.87% | 11.49% | 12.52% | 64.10% | 36.82% | 26.99% | 39.75% | 16.21% | 19.36% |
| AS target | 1.88% | 28.19% | 3.24% | 4.33% | 13.48% | 10.19% | 8.81% | 7.71% | 5.71% | 8.22% |

Table 4.2: Entangled marginals

| Source | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|------------------|-------|--------|-------|--------|--------|--------|--------|--------|--------|--------|
| AS ignore-target | 3.21% | 60.25% | 7.43% | 10.57% | 53.79% | 35.07% | 25.06% | 18.52% | 23.28% | 33.15% |
| AS target | 0.86% | 23.52% | 1.46% | 2.88% | 10.88% | 9.28% | 6.34% | 4.24% | 8.24% | 12.08% |

Table 4.3: Entangled weight decay marginals

Chapter 5

Conclusion

Chapter 6

Appendix

| Source | Targets | | | | | | | | | |
|--------|-------------------|-------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 0 | - | 7.02% (0.00%) | 9.80% (5.74%) | 7.64% (4.29%) | 9.62% (1.88%) | 11.60% (5.86%) | 8.63% (5.43%) | 6.68% (0.10%) | 9.45% (5.67%) | 8.80% (0.31%) |
| 1 | 99.16% (0.00%) | - | 98.41% (93.66%) | 95.32% (4.48%) | 74.11% (44.89%) | 94.50% (10.42%) | 98.07% (26.55%) | 23.25% (13.57%) | 95.06% (73.08%) | 65.31% (39.78%) |
| 2 | 8.75% (1.01%) | 23.32% (0.43%) | - | 16.29% (5.93%) | 25.75% (4.79%) | 23.54% (1.58%) | 6.86% (2.29%) | 25.03% (12.78%) | 26.04% (21.08%) | 28.59% (0.44%) |
| 3 | 3.23% (0.58%) | 19.82% (0.34%) | 16.09% (12.37%) | - | 19.91% (0.81%) | 16.50% (14.61%) | 12.22% (1.90%) | 19.39% (1.70%) | 23.23% (18.40%) | 18.29% (4.60%) |
| 4 | 24.32% (0.37%) | 42.75% (0.49%) | 38.03% (24.18%) | 28.12% (6.60%) | - | 22.59% (4.10%) | 19.77% (11.01%) | 50.60% (7.13%) | 34.12% (25.46%) | 34.38% (30.99%) |
| 5 | 46.78% (3.42%) | 45.72% (0.92%) | 64.34% (8.55%) | 34.30% (26.49%) | 30.23% (7.59%) | - | 51.84% (11.15%) | 41.29% (1.72%) | 49.41% (31.49%) | 28.35% (6.56%) |
| 6 | 28.31% (6.89%) | 34.79% (2.51%) | 41.54% (32.64%) | 37.96% (6.35%) | 37.31% (22.98%) | 32.55% (10.34%) | - | 39.48% (2.40%) | 49.02% (28.82%) | 43.31% (5.98%) |
| 7 | 46.96% (2.03%) | 3.51% (0.11%) | 38.03% (23.94%) | 47.00% (2.54%) | 37.46% (5.23%) | 50.65% (5.63%) | 43.61% (0.60%) | - | 51.19% (29.52%) | 42.53% (37.01%) |
| 8 | 26.31% (0.91%) | 13.01% (1.47%) | 31.15% (25.29%) | 23.31% (13.83%) | 11.44% (5.95%) | 20.38% (11.75%) | 22.88% (6.85%) | 10.97% (5.37%) | - | 8.69% (5.26%) |
| 9 | 80.48% (0.80%) | 40.13% (1.77%) | 79.96% (5.46%) | 81.37% (22.60%) | 38.78% (36.85%) | 74.43% (24.06%) | 75.00% (3.90%) | 27.42% (20.96%) | 80.37% (57.27%) | - |

Table 6.1: Disentangled

| Source | Targets | | | | | | | | | |
|--------|--------------------|-------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 0 | - | 3.33% (0.00%) | 8.67% (2.82%) | 6.93% (2.10%) | 6.33% (0.10%) | 5.49% (1.83%) | 4.80% (2.24%) | 3.96% (0.00%) | 9.04% (5.99%) | 4.28% (1.88%) |
| 1 | 60.50% (0.21%) | - | 75.57% (51.43%) | 71.26% (40.99%) | 61.83% (24.38%) | 66.78% (14.38%) | 50.48% (18.72%) | 62.99% (28.26%) | 74.19% (56.95%) | 51.22% (18.36%) |
| 2 | 13.65% (1.53%) | 4.04% (0.10%) | - | 14.32% (11.05%) | 10.55% (0.66%) | 13.62% (0.24%) | 10.94% (1.80%) | 8.49% (1.15%) | 16.67% (11.00%) | 11.15% (1.61%) |
| 3 | 8.89% (0.56%) | 5.50% (0.11%) | 8.79% (4.45%) | - | 21.85% (0.33%) | 14.23% (10.80%) | 12.49% (0.57%) | 5.60% (0.54%) | 26.02% (19.91%) | 9.30% (1.73%) |
| 4 | 70.34% (2.76%) | 42.32% (0.33%) | 68.65% (9.35%) | 89.19% (10.26%) | - | 77.03% (4.84%) | 23.84% (3.95%) | 71.62% (4.77%) | 75.28% (30.85%) | 58.65% (54.19%) |
| 5 | 34.34% (2.67%) | 22.46% (0.25%) | 47.30% (4.90%) | 36.06% (24.04%) | 36.29% (0.37%) | - | 25.98% (6.62%) | 36.05% (0.37%) | 57.67% (43.56%) | 35.26% (8.97%) |
| 6 | 23.43% (16.49%) | 7.59% (0.22%) | 29.71% (7.35%) | 42.27% (8.71%) | 14.30% (6.60%) | 24.36% (11.59%) | - | 37.38% (0.00%) | 41.28% (25.68%) | 22.61% (2.61%) |
| 7 | 33.44% (1.55%) | 24.53% (0.32%) | 29.70% (6.77%) | 32.36% (15.10%) | 53.05% (6.09%) | 39.25% (1.22%) | 55.95% (0.45%) | - | 54.59% (8.72%) | 34.88% (29.17%) |
| 8 | 13.36% (1.03%) | 7.41% (0.11%) | 19.88% (8.44%) | 34.31% (28.09%) | 10.35% (1.25%) | 22.35% (5.74%) | 10.06% (1.73%) | 18.65% (0.80%) | - | 9.49% (4.23%) |
| 9 | 8.41% (1.44%) | 6.57% (0.32%) | 32.33% (1.72%) | 36.72% (20.34%) | 14.97% (11.20%) | 24.48% (7.67%) | 7.81% (0.23%) | 12.81% (6.35%) | 30.10% (24.70%) | - |

Table 6.2: Entangled

| Source | Targets | | | | | | | | | |
|--------|--------------------|-------------------|--------------------|--------------------|-------------------|-------------------|--------------------|--------------------|--------------------|--------------------|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 0 | - | 2.22% (0.00%) | 5.60% (1.27%) | 4.95% (1.90%) | 2.11% (0.00%) | 4.15% (0.92%) | 2.05% (1.08%) | 1.68% (0.00%) | 3.74% (2.14%) | 2.40% (0.42%) |
| 1 | 70.15% (2.19%) | - | 75.69% (59.23%) | 65.31% (24.92%) | 39.46% (3.43%) | 56.55% (8.51%) | 39.42% (16.67%) | 64.58% (50.20%) | 78.83% (36.27%) | 52.23% (10.24%) |
| 2 | 8.79% (2.05%) | 4.53% (0.11%) | - | 6.61% (3.41%) | 6.69% (0.68%) | 9.53% (0.12%) | 8.34% (1.16%) | 7.46% (1.71%) | 7.74% (3.35%) | 7.19% (0.55%) |
| 3 | 9.45% (1.08%) | 5.85% (0.11%) | 9.47% (6.47%) | - | 14.82% (0.00%) | 14.01% (9.51%) | 12.02% (0.74%) | 8.44% (2.43%) | 9.60% (4.32%) | 11.51% (1.27%) |
| 4 | 54.66% (2.64%) | 28.52% (1.62%) | 57.04% (17.21%) | 73.44% (5.99%) | - | 74.53% (2.28%) | 25.22% (3.95%) | 60.29% (12.50%) | 64.06% (10.97%) | 46.33% (40.72%) |
| 5 | 40.51% (13.69%) | 20.58% (1.37%) | 42.05% (3.56%) | 32.15% (19.35%) | 39.34% (0.69%) | - | 33.92% (8.72%) | 34.32% (6.89%) | 37.85% (22.15%) | 34.92% (7.09%) |
| 6 | 18.98% (14.46%) | 8.28% (1.20%) | 35.20% (11.62%) | 33.00% (3.51%) | 15.51% (4.84%) | 19.98% (5.52%) | - | 31.84% (0.33%) | 39.14% (14.33%) | 23.60% (1.21%) |
| 7 | 13.27% (1.72%) | 7.84% (1.18%) | 21.82% (11.18%) | 18.72% (8.08%) | 19.12% (1.04%) | 18.45% (1.12%) | 23.76% (0.12%) | - | 29.38% (3.72%) | 14.30% (10.01%) |
| 8 | 19.11% (4.97%) | 15.19% (1.42%) | 37.68% (28.19%) | 33.12% (18.57%) | 16.25% (0.40%) | 17.43% (3.86%) | 25.20% (4.96%) | 30.09% (6.18%) | - | 15.43% (5.58%) |
| 9 | 25.66% (2.89%) | 16.65% (1.50%) | 49.94% (11.74%) | 56.67% (22.12%) | 13.68% (6.49%) | 37.21% (8.22%) | 21.00% (0.49%) | 38.96% (35.61%) | 38.54% (19.69%) | - |

Table 6.3: Entangled with weight decay

Bibliography

- [1] Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck. *CoRR*, abs/1612.00410, 2016.
- [2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.
- [3] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR*, abs/1206.5538, 2012.
- [4] David Berthelot, Tom Schumm, and Luke Metz. BEGAN: boundary equilibrium generative adversarial networks. *CoRR*, abs/1703.10717, 2017.
- [5] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.
- [6] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Praseem 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.
- [7] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec 1989.
- [8] 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.
- [9] 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.
- [10] 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.
- [11] 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.

- [12] 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.
- [13] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.
- [14] 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.
- [15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.
- [16] 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.
- [17] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.
- [18] 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.
- [19] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *CoRR*, abs/1512.09300, 2015.
- [20] 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.
- [21] 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.
- [22] 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.
- [23] 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.
- [24] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, May 1992.