University College London
Department of Computer Science
MSc Machine Learning

Stochastic Adversarial Autoencoder

Robert Collyer

Supervised by
Dr David Barber
Dr Cristina Calnegru

September 2017

This report is submitted as part requirement for the MSc Degree in Machine Learning at University College London. It is substantially the result of my own work except where explicitly indicated in the text.

Abstract

This dissertation introductions a novel method coined the Stochastic Adversarial Autoencoder that attempts to encode the prior belief that that the outputs of most natural systems are smooth with respect to their inputs. It does this by injecting two sources of extra stochasticity into the latent representation of the Adversarial Autoencoder via the Gumbel-softmax trick and a pathwise stochastic gradient estimator. This dissertation also derives the evidence lower bound of the joint probability distribution $p(x, y, z)$ which underpins how the model learns in a principled statistical framework.

The SAAE is compared to the Adversarial Autoencoder (AAE) and the Gumbel-softmax Variational Autoencoder (GSVAE) at the task of semi-supervised learning on the simple MNIST dataset and the more complex SVHN dataset. It is discovered that on the MNIST dataset, the SAAE increasingly outperforms the AAE as the size of the training set is reduced. Evidence that injecting stochasticity in the latent representation can create smoother and more robust latent representation that can be used for classification.

Code for this project can be found at: https://github.com/RobertCollyer/dissertation

Acknowledgements

I would like to thank my supervisors Dr David Barber and Dr Cristina Calnegru for their advice and guidance in this dissertation. I am also grateful to Sarah Collyer for her continued support and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The aim of this chapter is to give an introduction and overview of the work conducted in this dissertation. It includes the motivation behind investigating the application of neural networks in generative models, particularly to the task of semi-supervised learning. Followed by a summary of tools and statistical techniques that are used to build the state-of-the-art models in the literature are then introduced. Along with an overview of Our contribution along with the results obtained from the experiments.

1.1 Motivation

Semi-supervised learning is a particularly important problem as it is also a practical problem, rather than only being an academic one. It is concerned with predicting the labels of observations, when only a few labels are available but observations are abundant. This is often the case in practice as labels are expensive to obtain.

A successful class of semi-supervised learning are generative models. Recently the application of neural networks to generative models have had a great deal of success, achieving state-of-the-art results. In particular, generative models within the autoencoder framework as they allow for efficient inference via a variational network that amortizes the cost of inference. They are able to discover hidden structure and are able to efficiently disentangle factors of variations, including style and content. Latent representations that correspond to content can then be used for classification. Along with semi-supervised learning, generative models are also useful for compression, denoising and in-painting, amongst other tasks.

1.2 Summary of relevant techniques and work

The dissertation starts by introducing the tools and statistical techniques that are used to build the state-of-the-art models in the literature. It gives a brief introduction to neural networks, that describes the model architecture, its objective function and how this is used to train the neural network.

Next discussed is the idea of an autoencoder, which is a neural network that tries to output a reconstruction of its input. It has a latent representation that has a dimensionality less than the input. This forces the model to learn a lower dimensional representation of the input. Autoencoders (and neural networks in general) can have a large capacity and are therefore prone to over-fitting. Consequently the concept of regularization, as a method to reduce over-fitting by limiting the models capacity, is presented. Two methods of regularization are discussed: constraining the weights of a model and introducing random perturbations.

This leads on to combining autoencoders with a probabilistic interpretation that allows them to reason about uncertainty. The decoder is considered to be a generative network and the encoder as an inference network. The reader is introduced to four statistical techniques: the evidence lower bound for the joint probability distribution $p(x, y)$, the pathwise stochastic gradient estimator, adversarial learning and the density ratio trick. Using these techniques, the variational autoencoder and the adversarial autoencoder are built from the ground up. This provides a better understanding and intuition of how the models work and are related. Later, these building blocks enable other models in the literature to be broken down and compared against each other.

Next the reader is introduced to the concept of semi-supervised learning, where different assumptions and different classes of semi-supervised learning are discussed. The class of generative models is the next focus. In a similar fashion to the previous section, statistical techniques that help us build models from the ground up are presented: the evidence lower bound for the joint probability distribution $p(x, y, z)$, marginalization over classes values, the Gumbel-softmax trick and the categorical density ratio trick. Finally the reader is shown how these techniques can be put together to create the semi-supervised Gumbel-softmax variational autoencoder and the semi-supervised adversarial autoencoder.

1.3 Our contribution

The contribution of this dissertation are:

- A comprehensive review of the tools and statistical techniques used in the probabilistic autoencoder literature.
- An instruction manual on how to build the Gumbel-softmax Variational Autoencoder and Adversarial Autoencoder from ground up using the methods previously described.
- A comparison of other models in the literature.
- The introduction of a novel method coined the Stochastic Adversarial Autoencoder that attempts to encode the prior belief that that the outputs of most natural systems are smooth with respect to their inputs. It does this by injecting two sources of extra stochasticity into the Adversarial Autoencoder via the Gumbel-softmax trick and a pathwise stochastic gradient estimator.
- A derivation of the evidence lower bound of the joint probability distribution p(x,y,z). This underpins how the model learns in a principled statistical framework.
- The comparison of the Stochastic Adversarial Autoencoder (SAAE) against:
    1. Adversarial Autoencoder (AAE) to measure the effect of injecting stochasticity into the latent representation vs not injecting stochasticity into the latent representation.
    2. Gumbel-softmax Variational Autoencoder (GSVAE) to measure the effect of regularizing by adversarial learning vs regularizing by the KL divergence.

1.4 Summary of results

The experiments in this dissertation, focus on the latent representations and semi-supervised classification error of the models. A summary of the models architecture can be found in section 4.2 and a summary of the training procedure is found in section 4.3. The motivation behind the experiments is found in section 4.4 and is summarized below.

A comparison between the models was conducted on two datasets. The MNIST dataset which has a low variation between images and the SVHN which is a more

complex distribution. The purpose of this is to compare how the complexity of the dataset affects the relative performance of the models i.e whether a more complex distribution increases or decreases the advantage or disadvantage of a particular model over another.

A comparison is also conducted across different sizes of training set: 60k, 50k, 40k, 30k, 20k, 10k and 5k. This experiment was design to test how robust the latent representations are when the amount of stochasticity available via the training set is reduced.

1.4.1 Simple data distribution

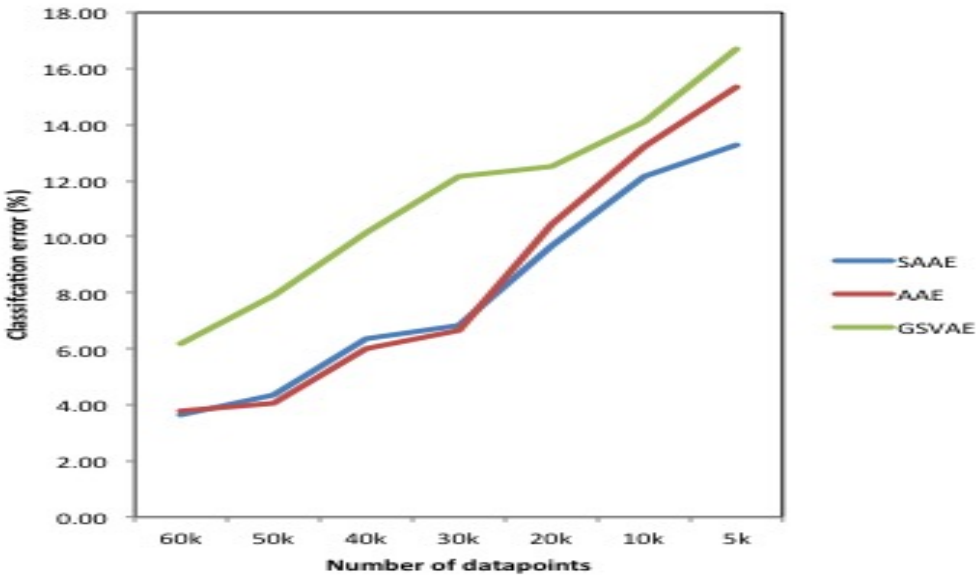
First, the hyper-parameters are tuned in three stages as described in section 4.3 and the results are reported in section 4.5.1. Then the models are trained on the full dataset, this is repeated 10 times using the same 100 labels. The average results are reported in table 1.1.

Table 1.1: Average semi-supervised classification errors of the SAAE, AAE and GSVAE on the MNIST dataset

MNIST	**SAAE**	**AAE**	**GSVAE**
Average classification error (%)	3.64	3.78	6.18

The results in table 1.1 show that the SAAE marginally outperformed the AAE, 3.64% vs 3.78% average classification error. This indicates that the injected stochasticity offers little benefit (in a dataset with low variation between images) when there exists ample observations to train the models. However the SAAE significantly outperformed the GSVAE, 3.64% vs 6.18% average classification error. This indicates that regularizing by adversarial learning is advantageous over regularizing by explicit KL divergence (again in a dataset with low variation) when there exists a large number of observations to train the models.

Figure 1.1: Plot of average semi-supervised classification error of the SAAE, AAE and GSVAE on the MNIST dataset as the number of data-points is reduced



It can be seen in figure 1.1 that the SAAE increasingly outperformed the AAE as the training set is reduced. This is evidence that as the available variation and stochasticity in the training set is reduced, the increase in the classification error can partly be offset by injecting stochasticity in the latent representation i.e the SAAE.

Against the GSVAE, as the training set is reduced, the out-performance of the SAAE remains largely constant. This is evidence that as the number of training points reduces in a low variance dataset, the adversarial regularization of the SAAE appears to consistently create a more robust latent representation than the GSVAE which regularizes with an explicit KL divergence.

1.4.2 Complex data distribution

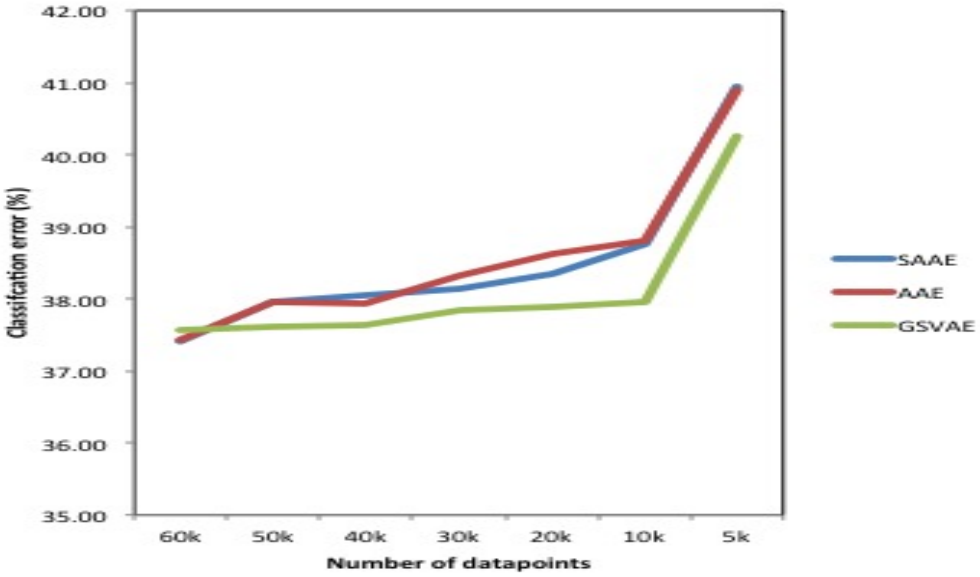
Now using a more complex data distribution, the hyper-parameters of the models are tuned with the same procedure as with the simple data distribution. The models are then trained on the full dataset, this is repeated 10 times using the same 2000 labels. The average results are reported in table 1.2.

Table 1.2: Average semi-supervised classification errors of the SAAE, AAE and GSVAE on the SVHN dataset

SVHN	**SAAE**	**AAE**	**GSVAE**
Average classification error (%)	37.41	37.42	37.57

The results in table 1.2 show that the SAAE has virtually the same performance as the AAE, 37.41% vs 37.42% average classification error. This indicates that the injected stochasticity offers little benefit (in a high variation datasets) when there exists ample observations to train the models. The SAAE also marginally outperforms the GSVAE, 37.41% vs 37.57% average classification error. This indicates that regularizing by adversarial learning gives little benefit over regularizing by explicit KL divergence (in a high variance dataset) when there exists a large number of observations to train the models.

Figure 1.2: Plot of average semi-supervised classification error of the SAAE, AAE and GSVAE on the SVHN dataset as the number of data-points is reduced



It can be seen in figure 1.2 that the SAAE performs in-line with the AAE as the training set is reduced. This indicates that the injected stochasticity has little or no effect on the robustness of the latent representations when there are large variations within a dataset, such as SVHN.

Against the GSVAE, as the training set is reduced, the under-performance of the SAAE remains largely constant. This is evidence that as the number of training points reduces in a higher variance dataset, the adversarial regularization of the

SAAE appears to consistently create a less robust latent representations than the GSVAE, which regularizes with an explicit KL divergence.

Overall the results are mixed. The benefits of the SAAE over the AAE are only apparent when the stochasticity of the training set is reduced. The comparison of the SAAE to the GSVAE showed that both could provided superior latent representation depending on the the complexity of the dataset.

Chapter 2

Background

The aim of this chapter is to introduce the tools that are used to build the models in the literature from the ground up. These tools are embedded in principled statistical framework which underpins how these models work and enables them to reason about uncertainty. The Gumbel-softmax Variational Autoencoder and Adversarial Autoencoder are then constructed using these methods. Other models in the literature are also discussed.

2.1 Neural networks

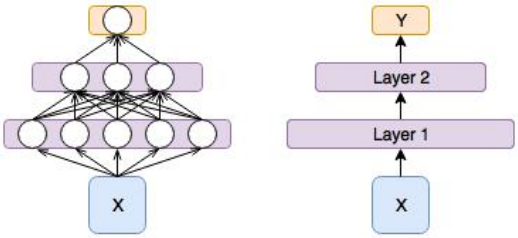
Neural networks were first introduced in 1943 [38] and gained widespread recognition in 1958 with the Perceptron [53]. They have now become one of the dominant machine learning techniques. This is due to their ability to efficiently learn complex, non-linear functions and automatically learn their own representations that can be used for classification and regression. They have also been proven to be universal function approximators [9], [10].

However the ability to scale these models has only been made possible in the past 10 years due to a number of reasons. The first is that neural networks require substantial computation power, which has now become readily available due to technological advances such as GPU's. The second is that neural networks require a lot of data to train, which again has become more accessible in recent years. The third is that neural networks are difficult to train, however this has now become easier with the development of heuristic approaches, discussed in section 2.1.3 to aid model training.

2.1.1 Model architecture

This dissertation focuses on feed-forward neural networks, however it should be noted that recurrent neural networks also exist to deal with sequential data. Feed-forward neural networks are composed of an input layer, one or more hidden layers and an output layer. The structure of a neural network can be represented as a graph. The nodes are called neurons and the vertices have weights associated to them. Each neuron in a layer is a weighted linear combination of the nodes in the previous layer, followed by a non-linearity activation, such as a sigmoid or ReLu [42]. For classification, a softmax transformation is applied to the output; for regression, only a linear transformation is applied to the output. In Figure 2.1, on the left representation, is an example of a fully connected neural network. Going forward the simplified representation on the right will be used.

Figure 2.1: Representation of a two layer neural network with nodes and weights (left). Representation of a two layer neural network without nodes or weights (right)



Two popular types of layer are the fully connected layer (depicted above) and the convolutional [34] layer. Convolutional layers are locally-connected layers with weight sharing, that allows for translation invariance.

2.1.2 Objective function

The objective function, $J(\theta)$ is used to guide the training of a neural network. It is a measure of the models error for a given set of parameters θ. The choice of objective function is application dependent. If a neural network is applied to the task of classification (discrete labels), then the negative log-likelihood (NLL) / cross entropy [56] stated in equation 2.1 is a good choice for an objective function.

$$\text{NLL} = -\sum_i y'_i \log f_\theta(x_i) \tag{2.1}$$

However if a neural network is applied to the task of regression (continuous labels) then the mean squared error (MSE) found in equation 2.2 is a good choice.

$$\text{MSE} = \frac{1}{N}\sum_{i}\left(y_i - f_\theta(x_i)\right)^2 \tag{2.2}$$

The mean absolute error (MAE) is also a prominent objective function that measures the absolute difference rather than the squared difference.

2.1.3 Learning process

To be able to learn, neural networks needs a mechanism to be able to transmit the information from the objective function back through the model architecture so that the weights can be updated appropriately. This mechanism is called back-propagation [54]. Back-propagation takes the derivatives of the objective function and iteratively applies the chain rule until the derivatives of the objective function, with respect to all the weights, have been obtained.

Armed with these derivatives, there are numerous optimization methods that can be used to updated the weights of the neural network. The simplest method is gradient descent, found in equation 2.3, which updates the weights using a learning rate and the negative gradient, which is why it is called gradient *descent*. Gradient descent continues to update the parameters until convergence.

$$\theta_{n+1} = \theta_n - \epsilon.\nabla J_x(\theta_n) \tag{2.3}$$

An issue with gradient descent though is that it is applied to all of the dataset at once, which becomes a problem when the dataset is very large. One solution is to approximate the gradient with a mini-batch of the dataset. This is known as stochastic gradient descent (SGD) and can be found in equation 2.4.

$$\theta_{n+1} = \theta_n - \epsilon.\nabla J_{x_{mini}}(\theta_n) \tag{2.4}$$

One issue with SGD is that it oscillates when travelling through surfaces that curve much more in one dimension than another [60]. A solution to this problem is the idea of momentum [54], [47] which can be described by equations 2.5 and 2.6. It works by increasing the momentum of a dimensions whose gradients consistently point in the same direction and reduces the momentum of dimensions where the direction of the gradients oscillates.

$$v_n = \mu.v_{n-1} + \epsilon.\nabla J_{x_{mini}}(\theta_n) \tag{2.5}$$

$$\theta_{n+1} = \theta_n - v_n \tag{2.6}$$

There are many other optimization algorithms, including adaptive moment estimation (Adam) [26] and Root Mean Square Propagation (RMSProp).

As mentioned in the introduction of this chapter, neural networks have historically been hard to train. Reasons for this include a non-convex objective function, along with exploding and vanishing gradients. Heuristic approaches have been developed that enable more reliable training. These include batch normalization [22] and Xavier weight initialization [14], amongst others.

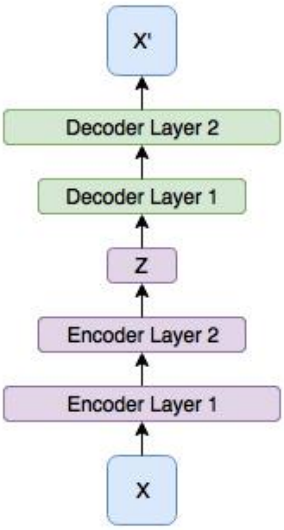
2.2 Autoencoders

The previous section describes neural networks and their application in a supervised learning setting. However neural networks can also be used in the task of unsupervised learning where they are called autoencoders [54].

2.2.1 Model architecture

Broadly speaking, an autoencoder can be split into two neural networks, shown in figure 2.2. Firstly, an encoder network that is a mapping from the observed data space into a latent representation space. Secondly, a decoder network that is a mapping from the latent representation space back to the observed data space.

Figure 2.2: Representation of an autoencoder with two layers in both the encoder and decoder



The goal is to reconstruct the input $x' = g(f(x))$. As with neural networks, an appropriate objective function is application dependent. For dimensionality reduction, a requirement is for the latent representation dimension to be less that the observed data, so that a lower dimensional representation can be learned. It has been shown [20] that this lower dimensional representation of the observed data can be useful for tasks such as classification. The lower dimension also reduces the curse of dimensionality.

An interesting observation is that if the activations of an autoencoder are linear, then it will learn the same subspace at principle component analysis (PCA) [2]. Whereas if an autoencoder uses non-linear activations, it learns a more powerful non-linear generalization of PCA.

2.2.2 Training and regularization

Autoencoders are just neural networks with a specific architecture. This means that they can be trained in the same way using all the techniques discussed in section 2.1. However what was not discussed in section 2.1 is that neural networks and therefore autoencoders, have the ability to over-fit to the data if given too much capacity. This problem can be managed with a concept called regularization, which limits the capacity of the model. This therefore stops the model over-fitting and makes sure that it generalizes well.

Constraining weights

It has been noted [30] that large weights in a machine learning model are an indication of over-fitting. Therefore one logical solution is to include a penalty term to the loss function that penalizes the size of an autoencoder's weights. This can be used to limit the capacity of the autoencoder. A common penalty is the L1 regularization, which penalizes the absolute magnitude of the weights. The L2 regularization penalizes the squared magnitude of the weights. The amount of regularization is determined via cross-validation. L2 regularization heavily penalizes large weights relative to small weights, which broadly diffuses the weights. This is in contrast to L1 regularization which penalizes the weights equally and therefore encourages weights to become zero, which makes the model more sparse.

The weights of an autoencoder can also be constained by limiting the size of the derivatives of the encoder with respect to the input. This is done by constraining the encoder's Jacobian matrix using the Frobenius norm. This is called a contractive autoencoder [51]. The amount of contraction is again chosen via cross-validation. This term pushes the representation to be contractive around a lower dimensional manifold of the observed data. This encourages the derivative to be flat and therefore encourages robustness of the representation with regard to the input.

Random perturbations

Another method of regularization is to inject random perturbations in the model. One way to do this is by adding random Gaussian noise [7] to the input during training. If this technique is applied to an autoencoder then it is called a denoising autoencoder [63]. The model first corrupts the observed data and then tries to reconstruct the original uncorrupted observed data i.e denoise. This allows the observed data to be different from the corrupted version but still be near each other in the observed data space. It has been shown [62] that denoising autoencoders can lead to more robust representations than those learned by a standard autoencoder.

Another method to inject random perturbations is by a technique called dropout [21], [59]. It works by randomly dropping hidden units in the network with a certain probability during training. This forces the model not to rely on any neuron too much. It is similar to how ensemble methods learn and can be interpreted as a prior distribution on the weights [13].

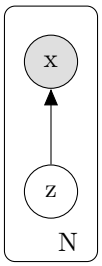
2.3 Probabilistic autoencoders

Deterministic autoencoders become a lot more powerful when combined with a probabilistic interpretation as this allows the models to reason about uncertainty and gives a deeper understanding into how the models work.

2.3.1 Latent variable models

Latent variable models describe a stochastic mechanism from which the observed data is generated. They define a joint probability distribution over the observed data and unobserved (latent) variables, $p(x, z)$. The unobserved variables represent underlying hidden factors of variation, while the observed data is conditionally generated from these hidden factors. Figure 2.3 shows a latent variable model represented as a directed probabilistic graphical model (plate notation).

Figure 2.3: Graphical model of generative process in plate notation



In this model, $p(z)$ is known as the prior and $p(x|z)$ as the likelihood. The goals of the investigation and the attributes of the data are used to guide the assumptions i.e if observed data is binary then the likelihood will be a Bernoulli distribution; if it is continuous then a Gaussian distribution maybe suitable.

Integrating over the latent variables, z, results in the marginal likelihood $p(x)$. It can be used as a measure of how good the model fits the data and is defined as follows:

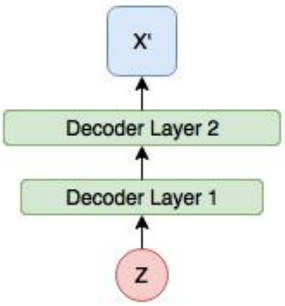
$$p(x) = \int p(x|z)p(z)dz \tag{2.7}$$

In fully observed models (no latent variables), the likelihood is directly computable. However in latent variable models, the marginal likelihood is intractable to compute and requires approximation.

Generative network

Within the autoencoder framework, the decoder network can be thought of as a likelihood function, $p_\theta(x|z)$, that is parametrized by θ. It is a non-linear function that describes how the high dimensional observed data, x, with complex dependencies (i.e images) are generated from the latent variables, z.

Figure 2.4: Representation of generator network with two layers that takes as input a random source of noise



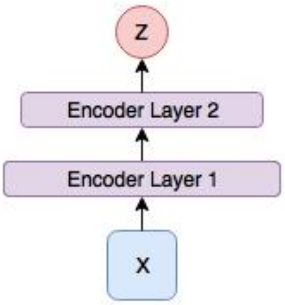
A generative model requires that the latent variables in the hidden representation, z, be random variables that can be easily sampled from. Given that the observed data is continuous, an appropriate likelihood needs to be chosen. If a standard normal distribution is assumed for the observation noise, then this equates to using a mean squared error or L2 reconstruction loss. Whereas if a Laplace distribution is assumed, then this equates to using a mean absolute error or L1 reconstruction loss.

Inference network

Along with being able to generate likely data, a good generative model should also be useful for inference. Inference, $q(z|x)$, is the process of disentangling the observed data to a latent representation. It inverts the generative process i.e it takes observed data and returns a distribution over the latent values that are likely to generate the observed data.

In the autoencoder framework, the encoder network can be thought of as an inference model, $q_\phi(z|x)$, that is parametrized by ϕ. It approximates the true but intractable posterior distribution, $p(z|x)$. This can be seen in figure 2.5.

Figure 2.5: Representation of inference network with two layers that takes as input an observation



This type of inference is called amortized or variational inference. This is because the inference network contains a shared set of global variational parameters, ϕ, which enables the cost of inference to be amortized across all data-points.

2.3.2 Training and regularization

As mentioned in section 2.3.1, calculating the marginal likelihood in equation 2.7 is intractable. Therefore a method to approximate it is required. One method is to derive a lower bound for the marginal likelihood. This transforms the integration problem into one of optimizing a lower bound.

Evidence lower bound

The evidence lower bound (ELBO) or variational free energy is a lower bound on the evidence or marginal likelihood. It is a principled bound on the marginal likelihood. It is defined in equation 2.8 and can be derived from equation 2.7 via importance sampling, an application of Jensen's inequality, and by taking the expectation over the variational distribution, $q_\phi(z|x)$. A derivation [41] is in appendix A. Optimization of the ELBO can be performed using the methods in section 2.1.3 by adjusting the variational parameters, ϕ.

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x|z)\right] - \mathrm{KL}\left[q_\phi(z|x) \,||\, p(z)\right] \tag{2.8}$$

It is interesting to note that the first term is the expected log-likelihood, which relates to reconstruction loss in the autoencoder framework. This term encourages the decoder to learn to reconstruct the observed data.

The second term can be viewed as a regularizer. It is the KL divergence between the encoder's approximate variational posterior distribution, $q_\phi(z|x)$, and the prior distribution, $p(z)$. It is a measure of how much information is lost when representing the prior with the approximate posterior. Optimizing this enforces a prior, $p(z)$, on the latent code. In other words, the encoder can restrict the capacity of the latent space to output values that have likely come from the prior distribution. The KL divergence encourages the approximate posterior, $q_\phi(z|x)$, to pick the modes of the prior, $p(z)$.

Combing these two terms creates an objective function that focuses on both the fidelity of the reconstruction and conciseness of the latent representation.

Pathwise stochastic gradient estimator

For the autoencoder framework to be optimized, it requires the model to be differentiable with respect to its weights. This requires that the neurons in the model be deterministic. To avoid stochastic nodes, the pathwise stochastic gradient estimator (PSGE) [50], also known as the reparameterization trick [27], can be used. The PSGE avoids direct sampling from the distribution. It instead allows the random variable to be calculated as a deterministic transformation of the noise [5].

When the posterior, $q_\phi(z|x)$ is assumed to be a Gaussian distribution, the encoder outputs the parameters $\mu_\phi(x)$ and $\Sigma_\phi(x)$. z can be sampled using these deterministic parameters and auxiliary random variable, ϵ. $z = \mu_\phi(x)+\Sigma_\phi(x)\circ\epsilon$ where $\epsilon \sim N(0,1)$. Or more generally, $z = f(x,\epsilon)$ where ϵ is a source of random noise.

In the ELBO equation 2.8 the second term is the KL divergence between the approximate posterior $q_\phi(z|x)$ and prior $p(z)$. However it is worth noting that only a limited number of distributions can analytically integrate the KL divergence in closed form. One of which is the Gaussian distribution, which has the following closed form KL divergence $KL\big[q_\phi(z|x) \;||\; p(z)\big]$ [45].

$$KL\big[N(\mu_\phi(x),\Sigma_\phi(x)) \;||\; N(0,1)\big] = \frac{1}{2}\sum \mu_\phi^2(x) + \Sigma_\phi(x) - 1 - \log\,\Sigma_\phi(x) \qquad (2.9)$$

Another distribution that has a closed form KL divergence is the Laplace distribution [45], as is shown in equation 2.10.

$$KL\big[L(\mu_\phi(x), \Sigma_\phi(x)) \;||\; L(0,1)\big] = \sum \frac{1 - \Sigma_\phi(x)}{\Sigma_\phi(x)} + \log \Sigma_\phi(x) \tag{2.10}$$

Adversarial learning

Adversarial learning, also known as estimation-by-comparison [18], has become prominent with the introduction of generative adversarial networks (GANs) [15]. It is a model that is made up of two neural networks, a discriminator network, D_ψ, and a generator network, G_ϕ.

The aim of the discriminator network is to discriminate between observations from the true data distribution, $p_*(x)$, and observations from the adversarial generator network, q_ϕ. This should not be confused with the generator network (decoder) in the autoencoder framework. The discriminator outputs a probability that the observation is real. The aim of the adversarial generator network is to generate adversarial (fake) observations from a source of noise, $p(z)$, that tricks the discriminator.

While the model is trained, the adversarial generator network, q_ϕ, becomes better at generating observations as it learns the true distribution of the data. This happens until the discriminator cannot tell between real and fake examples, i.e it outputs 0.5 probability for both real and fake samples. This process allows adversarial training to learn very complicated distributions. Adversarial learning has demonstrated [48] an ability to generate images that look real.

The model can be thought of as using binary labels to represent the real (y = 1) observations drawn from $p_*(x)$ and fake (y = 0) observations drawn from $q_\phi(x)$. The discriminator is then used to estimate the probability of whether an observation is real or fake. This is used to guide the optimization of the discriminator network parameters, ϕ, by minimizing the negative log likelihood.

$$\mathbb{E}_{p_*(x)}\,\big[-\log\, D_\psi(x)\big] + \mathbb{E}_{q_\phi(x)}\,\big[-\log\,(1 - D_\psi(x))\big] \tag{2.11}$$

To update the generator network parameters ϕ, equation 2.12 is maximized. This is the expected log probability that the discriminator classifies incorrectly.

$$\mathbb{E}_{q_\phi(x)}\,\big[\log\,(1 - D_\psi(x))\big] \tag{2.12}$$

A two-player, min-max game V (D, G) can be created, where the discriminator and the generator are updated alternatively:

$$\min_{G_\phi} \max_{D_\psi} V(D_\psi, G_\phi) = \mathbb{E}_{x\sim p_*(x)}\big[\log\, D_\psi(x)\big] + \mathbb{E}_{z\sim p_z(z)}\big[\log\,(1 - D_\psi(G_\phi(z)))\big] \quad (2.13)$$

An issue with generative adversarial networks is that they lack an inference network. This prevents the observed data from being disentangled into a latent representation. Another issue is that they suffer from mode collapse which can make them difficult to train.

Density ratio trick

If the GAN process is reversed (inference), then the observed data becomes the random source of noise. Adversarial learning can then be used to match the output of the generator to a specified distribution i.e Gaussian. The generator becomes a variational inference network i.e it takes observations as input, and outputs a latent representation with a specified distribution. The discriminator attempts to distinguish between real and fake samples of z.

Using the notation that observations drawn from the real distribution, $p_*(z)$, have a label of 1, allows the observations to be expressed conditionally as $p(z|y = 1)$. The same can be done with the fake observations, with the model q_ϕ being conditionally expressed as $p(z|y = 0)$. Using Bayes rule and the assumption that $p(y = 1) = p(y = 0)$ then the density ratio [41] can be defined as:

$$\frac{p_*(z)}{q_\phi(z)} = \frac{p(z|y=1)}{p(z|y=0)} = \frac{p(y=1|z)}{p(y=0|z)} = \frac{D_\psi(z)}{1 - D_\psi(z)} \quad (2.14)$$

This means that to calculate the density ratio only requires the ability to sample observations from the two distributions, $p_*(z)$ and $q_\phi(z)$ and pass them through the discriminator, $D_\psi(z)$.

Revisiting the ELBO equation 2.8 and considering the second term, the density ratio trick can be used to estimate the KL divergence term.

$$-\mathrm{KL}\big[q_\phi(z|x)\;||\;p(z)\big] = \mathbb{E}_{q_\phi(z|x)}\left[\log\frac{p(z)}{q_\phi(z|x)}\right] \approx \mathbb{E}_{q_\phi(z|x)}\left[\log\,\frac{D_\psi(z)}{1 - D_\psi(z)}\right] \quad (2.15)$$

The density ratio trick [41], [37],[25],[39] is a really powerful tool, as it enables the

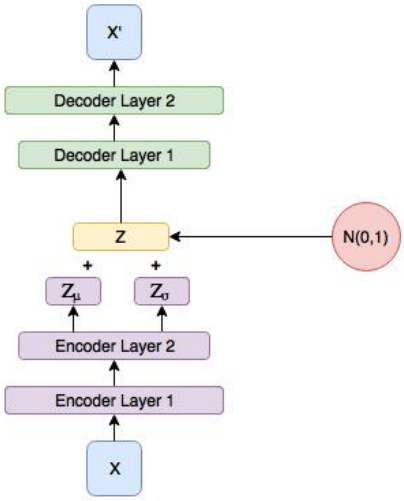
approximation of the KL divergence with any implicit distribution i.e the distribution only needs to be sampled.

2.3.3 Variational autoencoder

The latent variable model in an autoencoder framework, the ELBO, and the PSGE can be used to define the variational autoencoder [28]. It is a probabilistic graphical model that maximizes a lower bound on the model evidence.

From the ELBO objective in equation 2.8, the first term is the likelihood $p_\theta(x|z)$, which is parametrized by the generative network and a Gaussian distribution is assumed. For the second term, the KL divergence is approximated with the inference network $q_\phi(z|x)$ and a Gaussian prior $p(z)$ is assumed. The model can be seen in figure 2.6.

Figure 2.6: Representation of a variational autoencoder with two layers and a Gaussian auxiliary random variable



By applying PSGE to the latent representation, the inference and generator networks are jointly trained by optimising the ELBO. This is done using all the same techniques covered in section 2.1.

The variational autoencoder is a powerful probabilistic model that can be used for representation learning [4] via its inference network. It also has a generative model that can be used to generate data and impute missing data [44].

2.3.4 Adversarial autoencoder

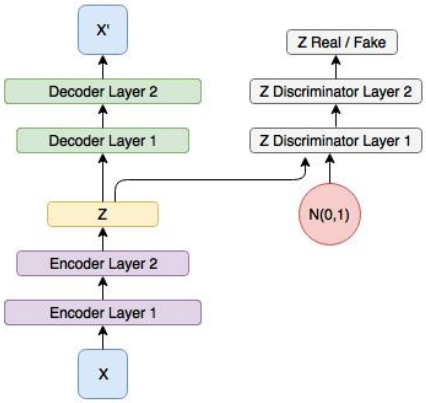
The adversarial autoencoder [37] is created using the latent variable model, the ELBO, adversarial learning and the density ratio trick. It is worth noting that as adversarial learning is being used rather than an explicit KL term, then a PSGE is not required.

As discussed in section 2.3.2, the density ratio can be used as an approximation of the KL divergence. It can therefore be used to replace the KL divergence term in the ELBO equation 2.8, as shown in equation 2.16.

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x|z)\right] - \mathbb{E}_{q_\phi(z|x)} \left[\log \frac{D_\psi(z)}{1 - D_\psi(z)}\right] \tag{2.16}$$

The latent representation can be regularized via adversarial learning from any implicit distribution. Similar to the variational autoencoder, for the first term of the ELBO, a Gaussian distribution is assumed for the likelihood, which is parametrized by the generative network $p_\theta(x|z)$. However it is worth noting that in the literature a Laplace likelihood is also a popular choice in models such as AGE, [61], BEGAN [6] and CycleGAN [66]. For the second term, adversarial learning is used to impose a prior distribution, $p(z)$, on the inference network, $q_\phi(z|x)$. This means that the inference network acts as both an inference model and also as an adversarial generator. The discriminator network, D_ψ, is then used to match the approximate posterior distribution, $q_\phi(z|x)$, to the prior distribution, $p(z)$. The model can be seen in figure 2.7.

Figure 2.7: Representation of an adversarial autoencoder with two layers, a Gaussian auxiliary random variable and a discriminator with two layers



The inference network (adversarial generator network), generator network and discriminator network are jointly trained by optimizing the ELBO. This is done in

two stages: The first stage runs a mini-batch through the model and updates the inference network and generator network (autoencoder) with respect to the reconstruction loss. The second stage runs the same mini-batch through the model but this time updates the discriminator network with respect to the discriminator loss and then the adversarial generator network (inference network) with respect to the adversarial generator loss.

This allows the posterior distribution, $q_\phi(z|x)$, to have arbitrary complexity. This is in contrast to the variational autoencoder where the posterior distribution is limited to a handful of distributions where the integral of the KL divergence term has a closed form analytical solution. The adversarial autoencoder is also a powerful probabilistic model that can be used for a large variety of tasks.

2.3.5 Related work

Both the VAE and AAE, assume a prescribed likelihood distribution such as a Gaussian or Laplace. The difference between the two is how the posterior, $q_\phi(z|x)$, is regularized. The VAE uses the KL divergence, which requires the assumed distribution to analytically integrate the KL divergence in closed form. The adversarial autoencoder uses adversarial learning to match the posterior, $q_\phi(z|x)$, to the prior, $p(z)$. This is a much more flexible approach, as any distribution can be used as long as it can be sampled from.

Prescribed vs implicit

It has been found that models with prescribed unimodal likelihoods i.e VAE and AAE, can lead to images that are blurry. This is because an unimodal distribution might not be the right assumption given the data. Whereas a GAN, which assumes an implicit likelihood distribution, can learn far more complex distributions and therefore produce images that are less blurry.

This seems to suggest that an implicit likelihood, $p_\phi(y|z)$, that is trained via adversarial learning, is better. However the main issue with models that use adversarial learning is that they are often hard to train due to mode collapse. In contrast prescribed likelihood models assign probability mass everywhere so mode collapse is not an issue. Also models with an implicit likelihood do not by definition have an explicit likelihood, which makes it more difficult to quantitatively measure their performance.

Hybrids

Given the benefits of both prescribed and implicit likelihoods, it is natural to combine the two approaches to create hybrid likelihoods, $p_\phi(x|z)$, that are more complicated but do not suffer from mode collapse. Hybrid posteriors, $q_\phi(z|x)$, can also be created that reap both sets of benefits. Some of the combinations of likelihoods and posteriors from the literature can be seen in table 2.1.

Table 2.1: Table showing likelihood and posterior assumptions made by VAE-GAN hybrid models in literature

Posterior \Likelihood	**Prescribed**	**Implicit**	**Hybrid**
Prescribed	VAE [28]	VAE-GAN [32]	
Implicit	AAE [37]	ALI [12]	α-GAN [52]
Hybrid			CycleGAN [66]

ALI [12], like the adversarial autoencoder, uses adversarial training to learn the posterior, however it uses an implicit likelihood. What is also interesting is that its discriminator learns to discriminate on the joint x, z space. The BiGAN [11] is similar to ALI except that it employs a deterministic posterior rather than a stochastic one.

The α-GAN [52] model uses a weighted sum of the prescribed and implicit likelihoods to learn a more complex distribution of the observation data that avoids both image blurriness and mode collapse. The Cycle-GAN [66] is similar, however it also uses a hybrid approach on the posterior distribution, $q_\phi(z|x)$, to obtain a more complex distribution.

2.4 Semi-supervised autoencoders

Semi-supervised learning is concerned with the task of classification when only a small number of the observations have labels. This is an issue encountered in practical applications of machine learning as labels can be expensive to obtain. It would be good to leverage the information and structure in the unlabelled data, $p(x)$, such that it enables inference, $p(y|x)$, to be performed.

2.4.1 Semi-supervised models

To be able to model the underlying data generating process, some assumptions need to be made. The four main assumptions [8] that can be made about the data are:

1. Smoothness assumption: If two points, x1 and x2, in a high-density region are close, then so should be the corresponding outputs, y1 and y2.
2. Cluster assumption: If points are in the same cluster, then they are likely to be of the same class.
3. Low density separation: The decision boundary should lie in a low-density region.
4. Manifold assumption: The (high-dimensional) data lie (roughly) on a low-dimensional manifold.

With these four assumptions, four classes of semi-supervised learning can be defined. Each method is built around one or more of the above assumptions.

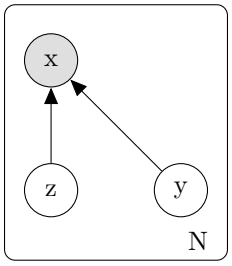
- Generative models, such as Gaussian mixture models or hidden Markov models [3];
- Low-density separation, such as transductive support vector machines [24];
- Graph-based methods, such as label propagation [67];
- Change of representation methods, such as graph kernels by spectral transformation [65].

Of these four classes of semi-supervised algorithm, generative models have become one of the most popular approaches. Generative models applied to semi-supervised learning can be thought of as a missing data imputation task for the classification. They try to learn the structure of the data. This means that they can disentangle the label from other hidden factors of variation [29], [35]. For instance in handwriting, the labels correspond to the letters whereas the factors of variation correspond to the style. The concept of semi-supervised learning can be combined into the probabilistic autoencoder framework to create models that obtain results that are state-of-the-art.

2.4.2 Semi-supervised latent variable model

In the semi-supervised latent variable model setting, it is assumed that the observed data is generated from two independent sets of random variables, y and z. A latent variable model represented as a directed probabilistic graphical model (plate notation) can be seen in Figure 2.8:

Figure 2.8: Graphical model of generative process in plate notation



This defines a joint probability distribution over the observed data and unobserved (latent) variables, $p(x, y, z)$. The marginal likelihood is calculated from the joint distribution, $p(x, y, z)$, by integrating over y and z:

$$p(x) = \int p(x|y, z)p(y)p(z)dydz \tag{2.17}$$

However, again this calculation is intractable and needs to be approximated.

2.4.3 Training and regularization

ELBO from joint probability distribution p(x,y,z)

To train a semi-supervised autoencoder, a lower bound on the marginal likelihood is required. The form of the ELBO from a joint probability distribution $p(x, y, z)$ can be found in equation 2.18 and the derivation is part of this dissertation's contribution which is found in chapter 3, 'Our contribution'.

$$\log p(x) \geq \mathbb{E}_{q_\phi(y,z|x)} \left[\log p_\theta(x|y, z)\right] - \mathrm{KL}\big[q_\phi(y|x) \;||\; p(y)\big] - \mathrm{KL}\big[q_\phi(z|x) \;||\; p(z)\big] \tag{2.18}$$

Optimization of the ELBO can be performed using the methods in section 2.1.3 and by adjusting the variational parameters, ϕ. The first and third terms in equation 2.18 are the same as equation 2.8. They are the expected log-likelihood and the KL divergence between the encoders approximate variational posterior distribution, $q_\phi(z|x)$ and the prior distribution, $p(z)$.

The second term is similar to the third and can be viewed as a regularizer. It is the KL divergence between the encoders approximate variational posterior distribution, $q_\phi(y|x)$ and the prior distribution, $p(y)$. Again it measures of how much information is lost when representing the prior with the approximate posterior. Optimizing then enforces a prior, $p(y)$ on the latent code. In other words, the encoder can restrict the capacity of the latent space to output values that have likely come from the prior

distribution. The KL divergence on a categorical divergence is defined in equation 2.19.

$$\mathrm{KL}\big|\big[\mathrm{Cat}_\phi(10) \;||\; \mathrm{Cat}(10)\big] = \sum_i \mathrm{Cat}_\phi(i)\, \log\left(\frac{\mathrm{Cat}_\phi(i)}{\mathrm{Cat}(i)}\right) \tag{2.19}$$

Combing these three terms creates an objective function that focuses on both the fidelity of the reconstruction and the conciseness of the latent representations of y and z. However, the issue here is that the PSGE can only be applied to continuous random variables, not discrete ones.

Marginalization over class values

One way to deal with the issue of back-propagating through discrete random variables in a semi-supervised setting is to avoid doing it [29]. For labelled data, inference of $q_\phi(z|y,x)$ is straight forward as y is observed. However for unlabelled data, y has to be marginalized out over all classes so that inference is still on $q_\phi(z|y,x)$. The main issue with this approach is that the marginalization scales linearly with the number of classes. The method is also found to be difficult to train end-to-end and required a pre-trained feature extractor .

Gumbel-softmax trick

Marginalizing out the unobserved latent random variable does not allow for end-to-end training and the cost scales linearly with the number of classes. The Gumbel-softmax trick [23] avoids these issues and allows for gradients to be directly back-propagated through $q_\phi(y|x)$. It replaces the non-differentiable sample from a categorical distribution with a differentiable sample from a continuous Gumbel-softmax distribution. Now that the sample is continuous, the PSGE can be used to easily back-propagate through $q_\phi(y|x)$. The Gumbel-softmax distribution can be smoothly decayed into a categorical distribution.

To implement the Gumbel-softmax trick, the Gumbel distribution [17] first needs to be defined so that it can be sampled from. The distribution is shown in equation 2.20.

$$g = -\log\big(-\log(u)\big) \quad \text{where} \quad u \sim \mathrm{Uniform}(0,1) \tag{2.20}$$

The Gumbel-argtmax trick [16] is shown in equation 2.21. It is a way to sample from a categorical distribution.

$$y = \underset{i}{\text{argmax}}[\ g_i + \log\ \pi_i\ ] \tag{2.21}$$

In equation 2.21, π_i are the class probabilities and $g_1 \dots g_k$ are samples from a Gumbel(0,1). However this is non-differentiable due to the argmax function. The argmax can be replaced with a softmax function in order to make the distribution differentiable.

$$y_i = \frac{\exp\ (\ (\ \log(\pi_i) + g_i\ )\ /\ \tau\ )}{\sum_{j=1}^{k} \exp\ (\ (\ \log(\pi_j) + g_j\ )\ /\ \tau\ )} \tag{2.22}$$

In equation 2.22, as the temperature, τ, decreases to 0, the Gumbel-softmax distribution becomes the categorical distribution. However there is a balance between low temperatures that are close to 0 but have a large variance in the gradients versus larger temperatures that have a lower variance in gradients that become more like a uniform distribution.

Categorical density ratio trick

The same density ratio trick used earlier can also be used for a categorical distribution, as all that is required is to be able to sample from the distribution. Revisiting the ELBO equation 2.18, the second term is replaced using the density ratio trick that estimates the KL divergence term.

$$-\text{KL}\big[q_\phi(y|x)\ ||\ p(y)\big] = \mathbb{E}_{q_\phi(y|x)}\left[\log \frac{p(y)}{q_\phi(y|x)}\right] \approx \mathbb{E}_{q_\phi(y|x)}\left[\log\ \frac{D_\omega(y)}{1 - D_\omega(y)}\right] \tag{2.23}$$

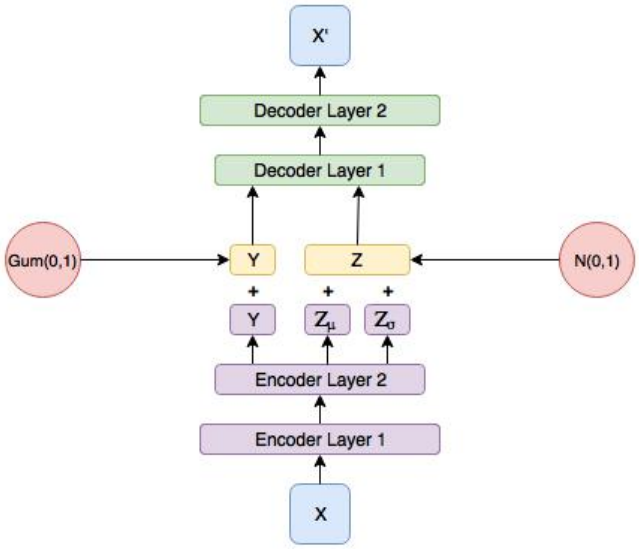
Where $D_\omega(y)$ in equation 2.23 is a discriminator that discriminates between real and fake samples of a categorical distribution.

2.4.4 Semi-supervised Gumbel-softmax variational autoencoder

The semi-supervised Gumbel-softmax variational autoencoder (ss-GSVAE) [23] is created from the methods previously discussed. The latent variable model in the autoencoder framework uses a generator network and an inference network. The Gumbel-softmax trick and PSGE trick are used to enable back-propagate the gradients of the categorical y and continuous z latent representations respectively. The structure of the model forces it to keep the label information independent from other

factors of variation. This can be seen in figure 2.9.

Figure 2.9: Representation of a ss-GSVAE with two layers, a Gumbel auxiliary random variable and a Gaussian auxiliary random variable



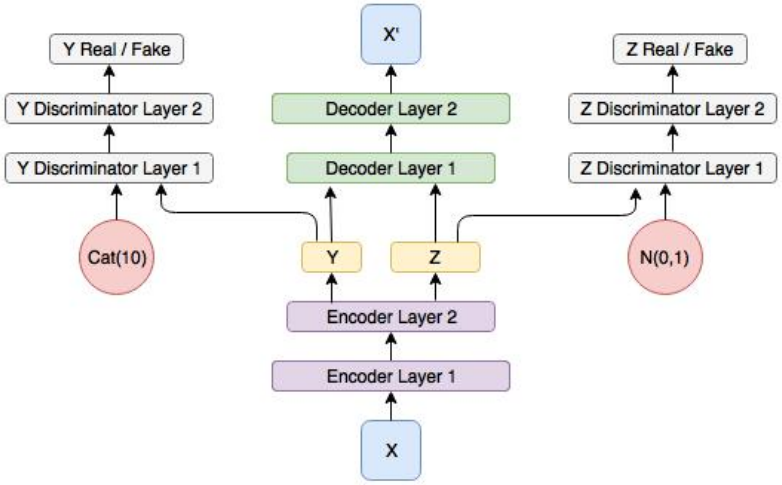
The ELBO objective from equation 2.18 is used to train the model. A Gaussian or Laplace distribution is assumed for the the likelihood $p_\theta(x|z)$ term, which is parametrized by the generative network. This encourages the model to reconstruct the images. The first KL term is a regularizer that encourage the inference network, $q_\phi(y|x)$, to become a categorical distribution, while the second KL divergence term is a regularizer that encourages the inference network, $q_\phi(z|x)$, to follow either a Gaussian or Laplace distribution.

The negative log likelihood objective from equation 2.1 is also used to train the model to classify the observed data by updating the inference network, $q_\phi(y|x)$. The model is optimized by alternating updates between the ELBO stage and the semi-supervised stage. The structure of the model forces it to keep the label information independent from other factors of variation. The inference network, $q_\phi(y|x)$, can then be used for classification.

2.4.5 Semi-supervised adversarial autoencoder

Again the semi-supervised adversarial autoencoder (ss-AAE) [37] uses the latent variable model in the autoencoder framework with a generator network and an inference network. However it does not require the Gumbel-softmax or PSGE to back-propagate the gradients of the categorical y and continuous z latent representations. This can be seen in figure 2.10.

Figure 2.10: Representation of a ss-AAE with two layers, a categorical auxiliary random variable, a Gaussian auxiliary random variable and two discriminator networks with two layers



In a similar to manner to the AAE, the KL divergences that regularize the latent representations in the ELBO equation 2.18 are approximated using the density ratio trick. Once for the categorical y latent variables and once for the continuous z latent variables. This can be seen in equation 2.24.

$$\log p(x) \geq \mathbb{E}_{q_\phi(y,z|x)} \left[\log p_\theta(x|y,z)\right] - \mathbb{E}_{q_\phi(y|x)}\left[\log \frac{D_\omega(y)}{1-D_\omega(y)}\right] - \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{D_\psi(z)}{1-D_\psi(z)}\right] \tag{2.24}$$

Similar to ss-GSVAE, the ELBO objective from equation 2.24 is used to train the model. Again a Gaussian or Laplace distribution is assumed for the likelihood $p_\theta(x|z)$ term, which is parametrized by the generative network. For the second and third terms, adversarial learning is used to impose the prior distributions, $p(y)$ and $p(z)$, on the inference networks, $q_\phi(y|x)$ and $q_\phi(z|x)$. Again this means that the inference network acts as both inference model and as an adversarial generator. The discriminator networks, D_ω and D_ψ, are then used to match the posterior distributions, $q_\phi(y|x)$ and $q_\phi(z|x)$, to $p(y)$ and $p(z)$.

The negative log likelihood objective from equation 2.1 is also used to train the model to classify the observed data by updating the inference network, $q_\phi(y|x)$.

The model is optimized by alternating updates between the reconstruction stage, the regularization stage and the semi-supervised stage. The observed data can then be classified using the inference network, $q_\phi(y|x)$.

2.4.6 Related work

In the past three years there has been significant progress in the task of semi-supervised learning with the application of deep generative models. This section classifies the models into ones with a prescribed likelihood and ones with an implicit likelihood that have been learned via adversarial training.

Prescribed likelihood

The VAE [29] described in section 2.3.3 was the first probabilistic autoencoder to obtain state-of-the-art results in the task of semi-supervised learning. However the model was difficult to train end-to-end and had to resort to pre-training a feature extractor as mentioned in section 2.4.3.

The ladder network [49] obtained superior results to the VAE and was able to be trained end-to-end. It works by connecting each layer in the generative network to its corresponding layer of the inference network. This enables the model to learn a manifold at each layer and forces the network to maintain a hierarchical structure to the type of information stored in each layer. It then uses the representations in the higher layers for semi-supervised classification.

The auxiliary deep generative model [35] is an extension of the VAE. It includes an auxiliary variable which enables the variational approximation to be more expressiveness. This allows the model to converge faster with a lower classification error than the previous two models.

Implicit likelihood

Most methods in the literature that apply GAN-like models to the task of semi-supervised learning use a modified discriminator as a classifier. Rather than return only a single output that represents the probability that a sample is real or fake, it now outputs n + 1 probabilities which represent the n classes and a fake class.

An example is ALI [12] which uses an objective that compares the divergences of the real and generated distribution with the prior distribution in the latent space. This means that no external discriminators are required in the process of learning, which makes them more architecturally and computationally efficient. Other successful semi-supervised (non-autoencoder) GAN based models include the SSGANS [46], the VAT [40] and the CATGAN. [58]

Chapter 3

Our contribution

The aim of this chapter is to define the contribution of this dissertation. It explains the motivation behind the stochastic adversarial autoencoder and how this is encoded into the model. The chapter also derives a lower bound of the evidence that underpins how the model works and is used to train the model.

3.1 Stochastic adversarial autoencoder

3.1.1 Motivation

The motivation and design of the stochastic adversarial autoencoder is driven by the prior belief that the outputs of most natural systems are smooth with respect to their inputs.

The stochastic adversarial autoencoder encodes this prior belief by injecting two extra sources of stochasticity into the adversarial autoencoder. This can be interpreted as a regularizer that smooths the mapping from input to output (hidden representation). This should theoretically increase the generalization and robustness of the latent representations, $q_\phi(z|x)$ and $q_\phi(y|x)$, the latter which is used for classification.

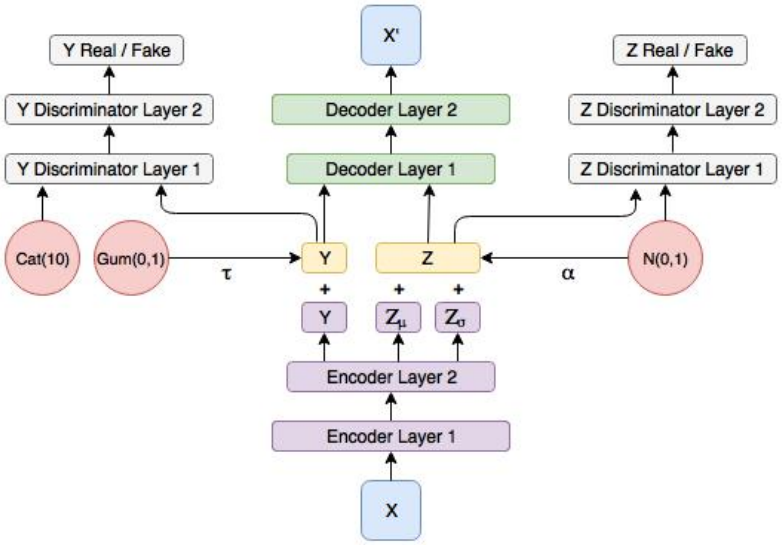
This model has parallels to other methods [7], [1], [31], [55], [64] that inject stochasticity into a model with the aim of making its latent representation more robust, particularly in the task of semi-supervised learning.

3.1.2 Model

The architecture of the stochastic adversarial autoencoder can be seen in figure 3.1. It consists of four networks: a generator network, $p_\theta(x|y,z)$; an inference (adversarial generator) network, $q_\phi(y,z|x)$, (that make the autoencoder); and two discriminator

networks, $D_\omega(y)$ and $D_\psi(z)$.

Figure 3.1: Representation of a stochastic adversarial autoencoder with two layers, a Gumbel auxiliary random variable, a categorical random variable, a Gaussian auxiliary random variable and two discriminator networks with two layers



Observed data is passed to the inference network which then outputs three vectors. The first vector, y, is a linear layer that is then transformed using the Gumbel-softmax trick. The transformation injects stochasticity into the Gumbel-softmax distribution layer via an abstracted auxiliary variable. The parameter τ is used to control how much stochasticity passes into the latent y representation. This represents $q_\phi(y|x)$. As mentioned in 2.4.3, this transformation allows for gradients to be back propagated through a deterministic node.

The last two outputs of the encoder network are two linear layers that represent the the mean and variance of $q_\phi(z|x)$. The PSGE is used to sample from a continuous distribution (Gaussian or Laplace) with mean z_μ and variance z_σ. The transformation injects stochasticity into the layer via a second abstracted auxiliary variable. Here the parameter α is used to control how much stochasticity passes into the latent z representation. This represents $q_\phi(z|x)$. Again this allows the nodes to be deterministic so that gradients can be back-propagated.

The first (of two) transformed outputs of the encoder network is then fed into the discriminator network, $D_\omega(y)$, along with random samples taken from a categorical distribution. The discriminator, $D_\omega(y)$, then predicts whether the input was either a real or fake sample.

The second transformed output of the encoder network is passed to the discriminator network, $D_\psi(z)$, along with random samples from either a Gaussian or Laplace distribution. Similarly the discriminator, $D_\psi(z)$, predicts whether the input is real or fake.

Both of the transformed outputs of the encoder network, $q_\phi(y|x)$, and $q_\phi(z|x)$ are also passed to the decoder network. The decoder network then outputs the reconstructed observed data. The stochastic adversarial autoencoder has the same architecture for its autoencoder (not including discriminative networks) as the Gumbel-softmax variational autoencoder.

3.1.3 Derivation of ELBO from joint probability distribution p(x,y,z)

To train the SAAE, ss-AAE and ss-GSVAE a modified lower bound on the evidence that is derived from the joint probability distribution p(x,y,z) is required. An explicit derivation with explanations can be found below and is a contribution of this dissertation.

Marginal likelihood

$$p(x) = \int\int p_\theta(x, y, z)\ dydz$$

Conditional

$$p(x) = \int\int p_\theta(x|y, z)p(y, z)\ dydz$$

Importance samples

$$p(x) = \int\int p_\theta(x|y, z)p(y, z)\frac{q_\phi(y|x)q_\phi(z|x)}{q_\phi(y|x)q_\phi(z|x)}\ dydz$$

Rearrange

$$p(x) = \int\int q_\phi(y|x)q_\phi(z|x)p_\theta(x|y, z)\frac{p(y, z)}{q_\phi(y|x)q_\phi(z|x)}\ dydz$$

Independence p(y,z) = p(y)p(z)

$$\log p(x) = \log \int\int q_\phi(y|x)q_\phi(z|x)p_\theta(x|y, z)\frac{p(y)p(z)}{q_\phi(y|x)q_\phi(z|x)}dydz$$

Take logs

$$\log p(x) = \log \int \int q_\phi(y|x) q_\phi(z|x) p_\theta(x|y,z) \frac{p(y,z)}{q_\phi(y|x) q_\phi(z|x)} \, dydz$$

Using Jensen's Inequality that $\log \int p(x)dx \geq \int \log p(x)dx$

$$\log p(x) \geq \int \log \int q_\phi(y|x) q_\phi(z|x) \left(p_\theta(x|y,z) \frac{p(y)}{q_\phi(y|x)} \frac{p(z)}{q_\phi(z|x)} \right) dydz$$

Using Jensen's Inequality that $\log \int p(x)g(x)dx \geq \int p(x) \log g(x)dx$

$$\log p(x) \geq \int \int q_\phi(y|x) q_\phi(z|x) \log \left(p_\theta(x|y,z) \frac{p(y)}{q_\phi(y|x)} \frac{p(z)}{q_\phi(z|x)} \right) dydz$$

Rearrange

$$\log p(x) \geq \int \int q_\phi(y|x) q_\phi(z|x) \log p_\theta(x|y,z) \, dydz + \int \int q_\phi(y|x) q_\phi(z|x) \log \left(\frac{p(y)}{q_\phi(y|x)} \frac{p(z)}{q_\phi(z|x)} \right) dydz$$

Rearrange

$$\log p(x) \geq \int \int q_\phi(y|x) q_\phi(z|x) \log p_\theta(x|y,z) \, dydz + \int \int q_\phi(y|x) q_\phi(z|x) \log \left(\frac{p(y)}{q_\phi(y|x)} \right) dydz + \int \int q_\phi(y|x) q_\phi(z|x) \log \left(\frac{p(z)}{q_\phi(z|x)} \right) dydz$$

Integrate out y and z

$$\log p(x) \geq \int \int q_\phi(y|x) q_\phi(z|x) \log p_\theta(x|y,z) dydz + \int q_\phi(y|x) \log \left(\frac{p(y)}{q_\phi(y|x)} \right) dy + \int q_\phi(z|x) \log \left(\frac{p(z)}{q_\phi(z|x)} \right) dz$$

Express as log likelihood and KL divergences

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\mathbb{E}_{q_\phi(y|x)} \left[\log p_\theta(x|y,z) \right] \right] - \mathrm{KL}\left[q_\phi(y|x) \,||\, p(y) \right] - \mathrm{KL}\left[q_\phi(z|x) \,||\, p(z) \right]$$

Now armed with the derivation of the ELBO for the joint probability distribution, the model can be trained.

3.1.4 Training and regularization

Similar to ss-GSVAE and ss-AAE, the SAAE employs the ELBO objective derived in section 3.1.3 to train the model. Again a Gaussian or Laplace distribution is assumed for the likelihood $p_\theta(x|z)$ term, which is parametrized by the generative

network. For the second and third terms, adversarial learning is used to impose the prior distributions, $p(y)$ and $p(z)$, on the inference networks, $q_\phi(y|x)$ and $q_\phi(z|x)$. Again this means that the inference network acts as both inference model and as an adversarial generator. The discriminator networks, D_ω and D_ψ are then used to match the posterior distributions, $q_\phi(y|x)$ and $q_\phi(z|x)$, to $p(y)$ and $p(z)$.

The negative log likelihood objective from equation 2.1 is also used to train the model to classify the observed data by updating the inference network, $q_\phi(y|x)$.

The model is optimized in the same way as the AAE, by alternating updates between the reconstruction stage, the regularization stage and the semi-supervised stage. Again the structure of the model forces it to keep the label information independent from the other factors of variation. The observed data can then be classified using the inference network, $q_\phi(y|x)$.

Chapter 4

Experiments

In this chapter, the semi-supervised classification performance of the SAAE, AAE and GSVAE are evaluated on the simple MNIST dataset and the more complex SVHN dataset.

4.1 Data

The experiments are conducted on two dataset. The first is the MNIST [33] dataset which is a collection of handwritten digits. It contains 60k training points and 10k test points. The digits are size normalized and centred in 28 by 28 grey-scale images. A 5k subset of the training data is used as a validation set while conducting a hyper-parameter search. This is done in a random and stratified manner. The MNIST dataset is considered simple as the variation between images is low.

The second is a modified version of the SVHN dataset [43]. The original dataset is created from real world images of house numbers taken from Google Street View. It consists of 604k training points and 26k test points. Each image is 32 by 32 RGB and contains a single digit that is centred but has considerably more variation than an MNIST image. Due to computational constraints, the SVHN dataset had to be modified and reduced. Firstly the images are converted to grey-scale. The training set is limited from 604k to 60k data-points and the test from 26k to 10k. Again all of these processes are done in a random and stratified manner. The SVHN is a more complex dataset as it has lots of variation between images as they are real world images.

Table 4.1: Summary table of MNIST and SVHN datasets

Datasets	h / w /d	Train	Test
MNIST	28x28x1	60k	10k
SVHN	32x32x3	604k	26k
SVHN-mod	32x32x1	60k	10k

4.2 Model Architecture

The model architectures are summarized in table 4.2 below. All three models have the same generative network which consists of a 20 dimensional input for y and z. y consists of 10 dimensions that represent the 10 different classes. z also consists of 10 dimensions that represent 10 factors of variation. This passes through two layers, each containing 1000 nodes and a ReLu non-linearity. The final layer consists of 784 or 1024 dimensions (depending on the dataset) which are then passed through a sigmoid non-linearity to output $p_\theta(x|y,z)$.

Table 4.2: Architectures of the SAAE, AAE and GSVAE

		SAAE		**AAE**		**GSVAE**	
		Units	**Non-lin**	**Units**	**Non-lin**	**Units**	**Non-lin**
Generative network	$p_\theta(x\|y,z)$	784/1024	Sigmoid	784/1024	Sigmoid	784/1024	Sigmoid
	layer 2	1000	Relu	1000	Relu	1000	Relu
	layer 1	1000	Relu	1000	Relu	1000	Relu
	$y,\ z$	20	-	20	-	20	-
Inference network	$q_\phi(z\|x)$	10	-	-	-	10	-
	$q_\phi(z_\sigma\|x)$	10	Linear	-	-	10	Linear
	$q_\phi(z_\mu\|x)$	10	Linear	10	Linear	10	Linear
	$q_\phi(y\|x)$	10	G-S	10	Softmax	10	G-S
	layer 2	1000	Relu	1000	Relu	1000	Relu
	layer 1	1000	Relu	1000	Relu	1000	Relu
	x	784/1024	-	784/1024	-	784/1024	-
Discriminator network y	$D_\psi(y)$	1	Sigmoid	1	Sigmoid	-	-
	layer 2	1000	Relu	1000	Relu	-	-
	layer 1	1000	Relu	1000	Relu	-	-
	y	10	-	10	-	-	-
Discriminator network z	$D_\omega(z)$	1	Sigmoid	1	Sigmoid	-	-
	layer 2	1000	Relu	1000	Relu	-	-
	layer 1	1000	Relu	1000	Relu	-	-
	z	10	-	10	-	-	-

The SAAE and the GSVAE also have the same inference network. This has an input x which has 784 or 1024 dimensions. This is then passed through two lay-

ers, each containing 1000 nodes and a ReLu non-linearity. The inference network then returns $q_\phi(y|x)$ which is 10 dimensional output that has been passed through a Gumbel-softmax. The Gumbel-softmax injects stochasticity via an auxiliary variable. The inference network also returns $q_\phi(z_\mu|x)$ and $q_\phi(z_\sigma|x)$ which are both 10 dimensional outputs. These are then transformed into the 10 dimensional $q_\phi(z|x)$ via the PSGE. This transformation injects stochasticity via a second auxiliary variable. The AAE inference network has the same input and two 1000 node layers with Relu non-linearities. However it return $q_\phi(y|x)$ which is again a 10 dimensional output but this time has only been put through a softmax transformation. It also returns $q_\phi(z_\mu|x)$ with a linear transformation. There is no $q_\phi(z_\sigma|x)$ or transformation via the PSGE.

The SAAE and AAE also have two discriminator networks. The first takes a 10 classes which it passes these through two layers each containing 1000 nodes and a ReLu non-linearity. It then returns a single node that has been passed through a sigmoid non-linearity that represents $D_\psi(y)$. The second discriminator has the same architecture except that is takes 10 dimensional input z and its output represents $D_\omega(z)$.

It is worth highlighting again that the SSAE and GSVAE have exactly the same generative and inference networks. The SAAE and AAE have the same two discriminator networks that are used for regularizing.

4.3 Training procedure

The three models use the objectives described in chapters 2 and 3. The errors from these objective functions are then used to guide the learning process. The learning process happens via stochastic gradient descent with momentum defined in equation 2.5 with a momentum parameter of 0.9. A mini-batch of 100 is used and the weights are initialized using Xavier initialization [14].

The hyper-parameters are tuned using a subset (5k) of the training set as a validation set. The models are trained until convergence. To avoid the exponential cost of conducting a full hyper-parameter search, the process is split into three stages. This is less extensive than doing a full parameter search, however it is a reasonable compromise given the computational constraints. A summary of the hyper-parameter search can be found in table 4.3 below.

Table 4.3: Hyper-parameter search procedure for SAAE, AAE and GSVAE

		SAAE	**AAE**	**GSVAE**
Stage 1	Reconstruction learning rates	[0.03,0.01,0.003,0.001]	[0.03,0.01,0.003,0.001]	-
	Discrim / Generative learning rates	[0.03,0.01,0.003,0.001]	[0.03,0.01,0.003,0.001]	-
	Reconstruction + KL learning rates	-	-	[0.03,0.01,0.003,0.001]
	Semi-supervised learning rates	[0.1,0.03,0.01,0.003]	[0.1,0.03,0.01,0.003]	[0.1,0.03,0.01,0.003]
Stage 2	Likelihood assumption	Gaussian/Laplace	Gaussian/Laplace	Gaussian/Laplace
	Posterior assumption	Gaussian/Laplace	Gaussian/Laplace	Gaussian/Laplace
	G-S temperature, τ	[0.01,0.05,0.1,0.3,0.5]	-	[0.01,0.05,0.1,0.3,0.5]
	Gaussian noise, α	[0.01,0.05,0.1,0.3,0.5]	-	-
Stage 3	Random label sample	-	[1,...,100]	-

The first stage involves searching for the optimal learning rates for the models. For the AAE, a Gaussian likelihood and posterior are assumed. A full search over the reconstruction, discriminative & generative, and semi-supervised learning rates is conducted. A hyper-parameter search on the SAAE learning rates (assuming no Gumbel and Gaussian noise) is the same as one on the AAE. The learning rates that are found for the AAE are also used for the SAAE. For the GSVAE, a Gaussian likelihood and posterior is assumed along with a Gumbel-softmax temperature, τ, of 0.3. A full search over the reconstruction + KL divergence and semi-supervised learning rates is conducted.

In stage two, the learning rates obtained in stage one are used. A full search is then conducted over the likelihood $p_\theta(x|y,z)$ assumptions, the posterior $q_\phi(z|x)$ assumptions, the Gumbel-softmax temperature, τ (where applicable), and the Gaussian noise, α (where applicable).

The third stage is to randomly pick different combinations of labels. This is because the semi-supervised classification error is dependent on the combination of labels chosen. This is particularly noticeable on the MNIST experiments where the results are leveraged on only 100 labels. The effect is less in the SVHN experiments where 2000 labels are used. 100 random label sample experiments are conducted with the AAE. The optimal labels are then used for all subsequent experiments.

4.4 Motivation behind experiments

This section discusses the comparisons, experiments and analysis that are conducted and the motivation behind them.

Table 4.4 shows that there is only one factor of difference between the SAAE and AAE and only one factor of difference between the SAAE and GSVAE. The experiments focus on these direct comparisons so that the effect of the individual technique can be isolated. Direct comparison between the AAE and GSVAE is avoided given that there are two factors of difference.

Table 4.4: Table showing the latent architecture and regularization of the SAAE, AAE and GSVAE

		Regularization	
		Adversarial	KL divergence
Latent architecture	Gumbel-softmax and PSGE	SAAE	GSVAE
	No Gumbel-softmax or PSGE	AAE	

Table 4.4 shows that the SAAE has the same regularization as the AAE. However the SAAE injects two sources of extra stochasticity into its latent representation during training. It does this to smooth the relationship between the input and the latent representation, whereas the AAE does not. Comparison of these two models indicates whether the injected stochasticity is beneficial for the task of semi-supervised learning.

Table 4.4 also shows that the SAAE and GSVAE share the same latent architecture i.e they both include a Gumbel-softmax and a PSGE to transform their y and z representations. The difference is that the SAAE regularizes its latent representations implicitly via adversarial learning, whereas the GSVAE regularizes explicitly using the KL divergence. Comparison of the two models suggests if one method of regularization is advantageous over another for the task of semi-supervised learning.

The three models are first compared on the full MNIST dataset with only 100 labels. The hyper-parameters of the three models are tuned using a single-validation set rather than cross-validation to avoid computational overheads. The hyper-parameters are tuned in three stages as described in section 4.3. The experiments are repeated 10 times using the tuned hyper-parameters and the same 100 labels, to obtain a more accurate estimate of each model's performance on the task of semi-supervised learning. A comparison is then done on whether the latent architecture (SSAE vs AAE) or regularization (SAAE vs GSVAE) has a meaningful effect on performance.

Given this dissertation is concerned with the task of semi-supervised learning, it is appropriate to analyse the latent representation for any insights and to check the models are working correctly. First, the statistics of the latent representation are checked to make sure that they correspond to those of the distribution they are being regularized to. A visual inspection is also conducted using a t-distributed stochastic neighbour embedding (t-SNE) [36]. The embedding allows for high dimensional data to be visualized in a lower dimension. It works by creating joint probabilities from the similarity between observations. It then minimizes the KL divergence between the joint probabilities of the lower dimensional embedding and the higher dimensional observations [57]. The inspection is only heuristic, however it gives a strong indication if the model is correctly disentangling the observations into class labels and other factors of variation. This is done to check that the models are working as expected.

The three models are then trained on a number of reduced training set: 60k, 50k, 40k, 30k, 20k, 10k and 5k. For each model the same hyper-parameters that are obtained from the validation on the full training set are used. The reason for this is that these hyper-parameters are learned using a larger number of observations and are therefore less likely to have over-fit than a model that has been trained on less observations. The model and hyper-parameters therefore generalize better and are more representative of the true underlying data generating distribution [19]. Again, to be consistent, the same 100 labels are used in each experiment. Each experiment is repeated 10 times and the averages are reported. This is done to obtain a robust estimate of each model's performance on the task of semi-supervised learning for each size of training set.

A comparison is made as to whether the latent architecture (SAAE vs AAE) has an effect on performance. This comparison is designed to test how robust latent representations are when the amount of stochasticity available via the training set is reduced and whether this can be offset and smoothed by injecting stochasticity into the latent representations via the Gumbel-softmax and PSGE. A comparison is also done to test whether the type of regularization (SAAE vs GSVAE) has a relative effect on performance as the training set is reduced.

These experiments are then repeated using a more complex dataset, SVHN, that contains more variation between each observation. 2000 labels are used. The purpose of this is to compare how the complexity of the dataset affects the relative performance of the models i.e whether a more complex distribution increases or decreases the advantage or disadvantage of a particular model over another.

4.5 Simple data distribution

In this section, the three models are trained on the MNIST dataset which has a smaller variation across observation than the SVHN dataset. Therefore it is a simple data distribution. Throughout the MNIST experiments, the same 100 labels are used. They are equally spread across the 10 classes.

4.5.1 Full dataset

Hyper-parameters

As mentioned in section 4.3, the hyper-parameters are tuned in three stages using a single validation set also described in section 4.3. The results of the hyper-parameter tuning for the three models on the MNIST dataset can be found in table 4.5.

Table 4.5: Tuned hyper-parameters for SAAE, AAE and GSVAE on MNIST dataset

	MNIST	**SAAE**	**AAE**	**GSVAE**
Stage 1	Reconstruction learning rates	0.01	0.01	-
	Discrim / Generative learning rates	0.1	0.1	-
	Reconstruction + KL learning rates	-	-	0.01
	Semi-supervised learning rates	0.1	0.1	0.1
Stage 2	Likelihood assumption	Gaussian	Gaussian	Gaussian
	Posterior assumption	Gaussian	Gaussian	Gaussian
	G-S temperature, τ	0.3	-	0.3
	Gaussian noise, α	0.1	-	-
Stage 3	Random label sample	57	57	57

Table 4.5 shows that the reconstruction and semi-supervised learning rates are the same. It is optimal to assume a Gaussian for both the likelihood and posterior . However it is worth noting that there is actually little difference in results between the Gaussian and Laplace assumptions. The optimal Gumbel-softmax temperature, τ, is found to be 0.3 for both the SAAE and GSVAE. The optimal amount of Gaussian noise, α, is found to be 0.1.

After a long time spent, attempting to replicate the results of the AAE [37], it was discovered that the stratified random sample of 100 labels is a very important hyper-parameter. The difference in classification error ranges from 12% to 3% i.e results are very leveraged on only a few labels. The 57th random label sample of

100 stratified labels gives the lowest classification error and is therefore used for the rest of the MNIST experiments.

Results

The models are first compared against each other in the task of semi-supervised classification using the relatively simple dataset MNIST. The models are trained on the full dataset using the tuned hyper-parameters obtained in the previous section. The experiments are repeated 10 times. The average results are reported in table 4.6.

Table 4.6: Average semi-supervised classification errors of the SAAE, AAE and GSVAE on the MNIST dataset

MNIST	**SAAE**	**AAE**	**GSVAE**
Average classification error (%)	3.64	3.78	6.18

The results in table 4.6 show that the SAAE marginally outperforms the AAE, 3.64% vs 3.78% average classification error. This indicates that the injected stochasticity offers little benefit (in a low variation dataset) when there exists ample observations to train the models.

However the SAAE significantly outperformed the GSVAE, 3.64% vs 6.18% average classification error. This indicates that regularizing by adversarial learning is advantageous over regularizing by explicit KL divergence (in a low variance dataset) when there exists a large number of observations to train the models.

Analysis of the latent representations

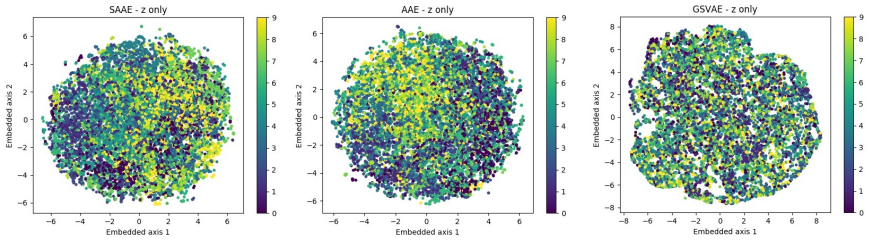
This sections looks at the latent representations for evidence that the models are working correctly. Firstly the statistics of the latent representations are checked to see if they corresponded to the distributions that they are being regularized to. It can be seen from table 4.7 that they did for all models with means and variances of 0.1 and 0.3 respectively for the categorical distribution, and means and variances of $\sim$0 and $\sim$1 respectively for the Gaussian distribution.

Table 4.7: Average regularization statistics of the SAAE, AAE and GSVAE on the MNIST dataset

MNIST	**SAAE**	**AAE**	**GSVAE**
y_μ	0.10	0.10	0.10
y_σ	0.30	0.30	0.30
z_μ	0.00	-0.01	-0.01
z_σ	0.99	0.98	1.00

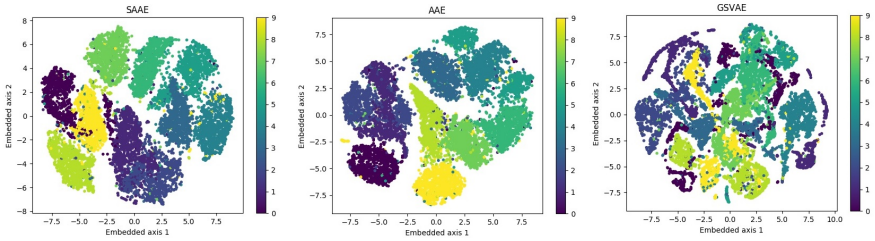
The latent representations of the models are also checked visually to see if they are behaving as expected. t-SNE is used to reduce dimension of the z representation from 10 to two dimensions. From figure 4.1 it can be seen that the labels are randomly spread over the plot in a circle with no apparent class clusters, indicating that class labels have been successfully disentangled from other factors of variation.

Figure 4.1: Visualization of two-dimensional t-SNE embedding of the joint z latent representation of the MNIST test set from the SAAE (left). Visualization of two-dimensional t-SNE embedding of the joint z latent representation of the MNIST test set from the AAE (middle). Visualization of two-dimensional t-SNE embedding of the joint z latent representation of the MNIST test set from the GSVAE (right)



This is in contrast to applying t-SNE on the combined y and z representations. As expected, figure 4.2 clearly shows clusters corresponding to the class labels. This corresponds back to the semi-supervised learning cluster assumption stated in section 2.4.1.

Figure 4.2: Visualization of two-dimensional t-SNE embedding of the joint y and z latent representation of the MNIST test set from the SAAE (left). Visualization of two-dimensional t-SNE embedding of the joint y and z latent representation of the MNIST test set from the AAE (middle). Visualization of two-dimensional t-SNE embedding of the joint y and z latent representation of the MNIST test set from the GSVAE (right).

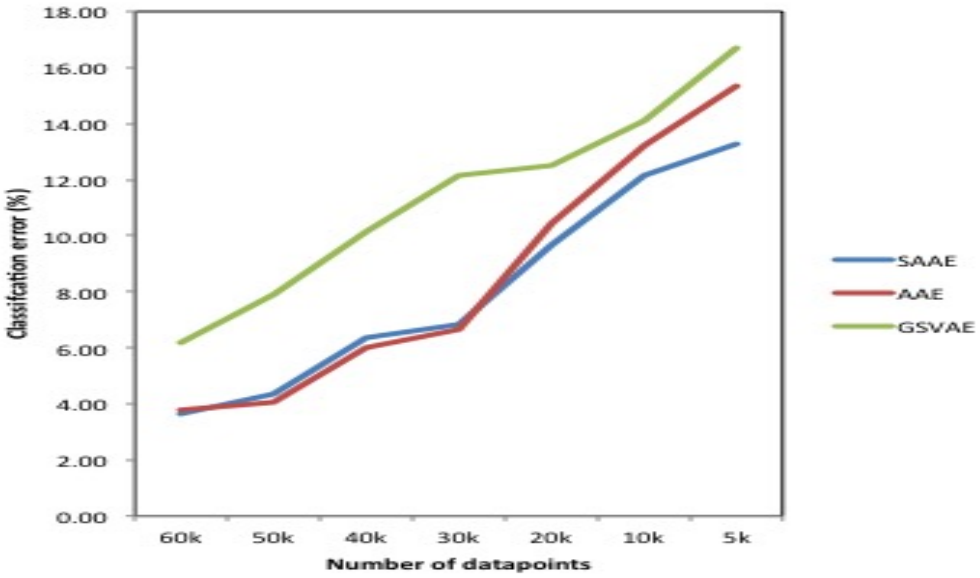


The statistics and visualizations indicate that the models are working as expected.

4.5.2 Reduced dataset

In the reduced dataset experiments, the models are trained on a reduced training set, down from 60k to 5k. To be consistent, the same 100 labels are used in each experiment. For each model, the same hyper-parameters that are found during cross-validation on the full training set are used. The reason for doing this is described in section 4.3. Each experiment is repeated 10 times and the average results are plotted in figure 4.3. The numerical results can be found in appendix B

Figure 4.3: Plot of average semi-supervised classification error of the SAAE, AAE and GSVAE on the MNIST dataset as the number of data-points is reduced



It can be seen in figure 4.3 that the SAAE increasingly outperformed the AAE as the training set is reduced. This is evidence that as the available variation and stochasticity in the training set is reduced, the increase in the classification error can partly be offset by injecting stochasticity in the latent representation. It aids the inference networks, $q_\phi(y|x)$ and $q_\phi(z|x)$, in creating smoother and more robust latent representations.

Against the GSVAE, as the training set is reduced, the out-performance of the SAAE remains largely constant. This is evidence that as the number of training points reduces in a low variance dataset, the adversarial regularization of the SAAE appears to consistently create a more robust latent representation than the GSVAE which regularizes with an explicit KL divergence.

4.6 Complex data distribution

In this section, the three models are trained on the SVHN dataset which has a larger variation across observations than the MNIST dataset. Therefore it is a more complex dataset. Throughout the SVHN experiments the same 2000 labels will be used. They are equally spread across the 10 classes. As mentioned in section 4.4, the purpose of this is to compare how the complexity of the dataset affects the relative performance of the models.

4.6.1 Full dataset

Hyper-parameters

Again as mentioned in section 4.3, the hyper-parameters are tuned in three stages, using a single validation set. The results of the hyper-parameter tuning for the three models on the SVHN dataset can be found in table 4.8.

Table 4.8: Tuned hyper-parameters for SAAE, AAE and GSVAE on SVHN dataset

	SVHN	**SAAE**	**AAE**	**GSVAE**
Stage 1	Reconstruction learning rates	0.001	0.001	-
	Discrim / Generative learning rates	0.001	0.01	-
	Reconstruction + KL learning rates	-	-	0.001
	Semi-supervised learning rates	0.003	0.003	0.003
Stage 2	Likelihood assumption	Gaussian	Gaussian	Gaussian
	Posterior assumption	Gaussian	Gaussian	Gaussian
	G-S temperature, τ	0.3	-	0.3
	Gaussian noise, α	0.1	-	-
Stage 3	Random label sample	11	11	11

Table 4.8 shows that the reconstruction and semi-supervised learning rates are the same. It is optimal to assume a Gaussian for both the likelihood and posterior. The optimal Gumbel-softmax temperature, τ, is again found to be 0.3 for both the SAAE and GSVAE. The optimal amount of Gaussian noise, α, is also found to be 0.1.

As the SVHN experiments use a larger number of labels, the effect of different random label samples matters less. The 11th subset of 2000 stratified random labels gives the best results which are used for the rest of the SVHN experiments.

Results

The models are now compared against each other in the task of semi-supervised classification using the more complex SVHN dataset. The models are trained on the full dataset using the tuned hyper-parameters obtained in the previous section. The experiments are repeated 10 times. The average results are reported in table 4.9.

Table 4.9: Average semi-supervised classification errors of the SAAE, AAE and GSVAE on the SVHN dataset

SVHN	**SAAE**	**AAE**	**GSVAE**
Average classification error (%)	37.41	37.42	37.57

The results in table 4.9 show that the SAAE has virtually the same performance as the AAE, 37.41% vs 37.42% average classification error. This indicates that the

injected stochasticity offers little benefit (in a high variation datasets) when there exists ample observations to train the models.

The SAAE also marginally outperforms the GSVAE, 37.41% vs 37.57% average classification error. This indicates that regularizing by adversarial learning gives little benefit over regularizing by explicit KL divergence (in a high variance dataset) when there exists a large number of observations to train the models.

Analysis of the latent representations

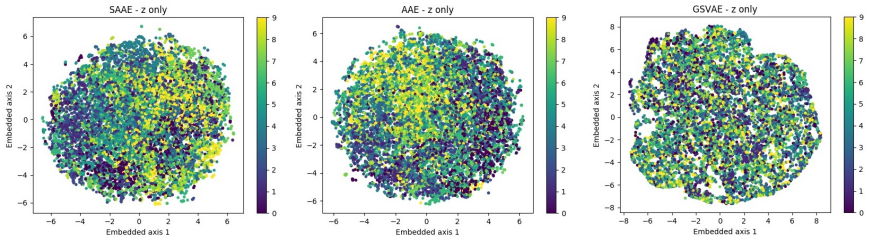
The statistics of the latent representations are checked to see if they correspond to the distributions that they are being regularized to. It is evident from table 4.10 that the three models manage to regularize $q_\phi(y|x)$ well with means and variances of 0.1 and 0.27 respectively. However the SAAE and AAE struggle to maintain the regularization for $q_\phi(z|x)$. They have means close to 0 but larger variances $\sim$1.15. This is not an issue for the GSVAE.

Table 4.10: Average regularization statistics of the SAAE, AAE and GSVAE on the SVHN dataset

SVHN	**SAAE**	**AAE**	**GSVAE**
y_μ	0.10	0.10	0.10
y_σ	0.27	0.27	0.27
z_μ	0.01	-0.01	0.00
z_σ	1.15	1.16	1.02

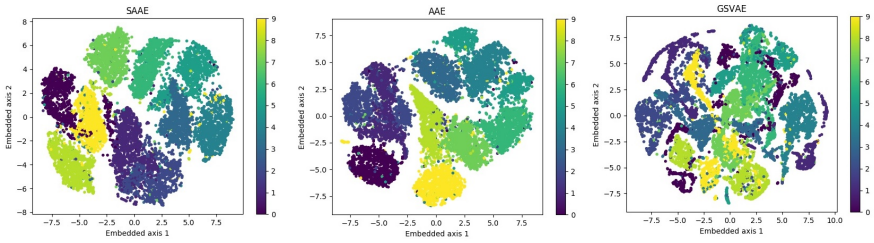
The latent representations of the models are also checked visually to see if they are behaving as expected. t-SNE is used to reduce dimension of the z representation from 10 to 2 dimensions. Figure 4.4, depicts the labels randomly spread over the plot in a circle with no apparent class clusters. This indicates that there is no apparent class label information.

Figure 4.4: Visualization of two-dimensional t-SNE embedding of the joint z latent representation of the SVHN test set from the SAAE (left). Visualization of two-dimensional t-SNE embedding of the joint z latent representation of the SVHN test set from the AAE (middle). Visualization of two-dimensional t-SNE embedding of the joint z latent representation of the SVHN test set from the GSVAE (right)



This is in contrast to using t-SNE on the combined y and z representation. As expected figure 4.5 clearly shows clusters corresponding to the class labels. This reflects the cluster assumption stated previously in section 4.5.

Figure 4.5: Visualization of two-dimensional t-SNE embedding of the joint y and z latent representation of the SVHN test set from the SAAE (left). Visualization of two-dimensional t-SNE embedding of the joint y and z latent representation of the SVHN test set from the AAE (middle). Visualization of two-dimensional t-SNE embedding of the joint y and z latent representation of the SVHN test set from the GSVAE (right)

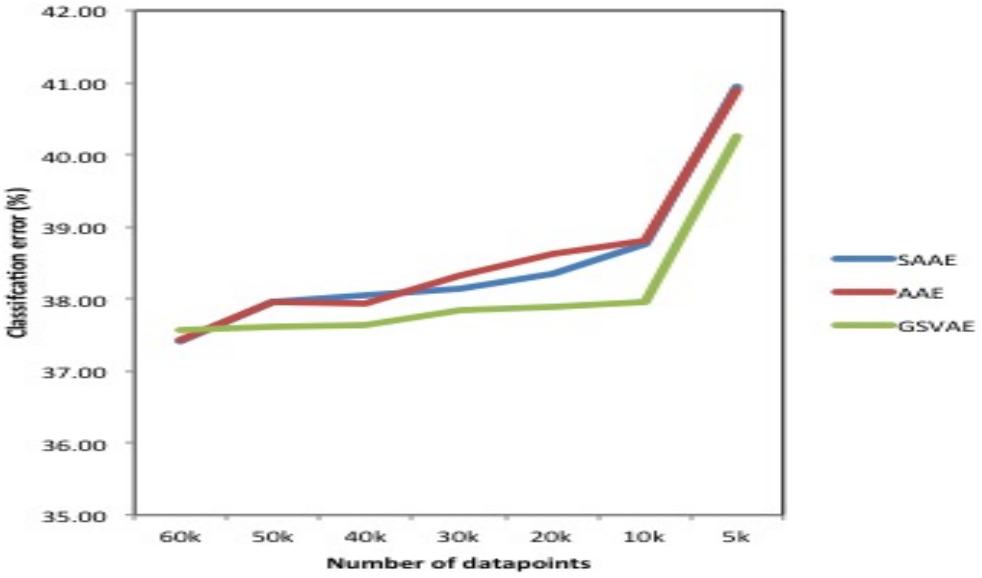


It appears from the statistics and visualizations that the models are working as expected.

4.6.2 Reduced dataset

In the reduced dataset experiments, as before the models are trained on a reduced training set, down from 60k to 5k. To be consistent, the same 2000 labels are used in each experiment. Again, for each model, the same hyper-parameters that are found during cross-validation on the full training set are used. Each experiment is repeated 10 times and the average results are plotted in figure 4.6. The numerical results can be found in appendix B.

Figure 4.6: Plot of average semi-supervised classification error of the SAAE, AAE and GSVAE on the SVHN dataset as the number of data-points is reduced



It can be seen in figure 4.6 that the SAAE performs in-line with the AAE as the training set is reduced. This indicates that the injected stochasticity has little or no effect on the robustness of the latent representations when there are large variations within a dataset, such as SVHN.

Against the GSVAE, as the training set is reduced, the under-performance of the SAAE remains largely constant. This is evidence that as the number of training points reduces in a higher variance dataset, the adversarial regularization of the SAAE appears to consistently create less robust latent representations than the GSVAE, which regularizes with an explicit KL divergence.

Chapter 5

Conclusion

5.1 Summary

This dissertation aimed to introduce the reader to various models in the probabilistic autoencoder literature. This was done by first introducing the reader to the fundamental statistical and deep learning techniques that are exploited. The techniques were then used to build the AAE and GSVAE from the ground up, along with their semi-supervised counterparts. From these building blocks, a novel model coined the stochastic adversarial autoencoder was developed. The design of which was driven by the prior belief that the outputs of most natural systems are smooth with respect to their inputs.

The SAAE has the same regularization as the AAE however the SAAE injects two sources of extra stochasticity into its latent representations during training to smooth the relationship between the input and the latent representations, whereas the AAE does not. The SAAE is also similar to the GSVAE as they both share the same latent architecture i.e they both include the Gumbel-softmax trick and a PSGE to transform their y and z representations. The difference is that the SAAE regularizes its latent representations implicitly via adversarial learning whereas the GSVAE regularizes explicitly using the KL divergence.

The semi-supervised experiments showed that the SAAE obtains a superior latent representation than either the AAE or the GSVAE when training with a distribution that is simple. The SAAE increasingly outperformed the AAE when the training set is reduced. Evidence that injecting stochasticity in the latent representation can create smoother and more robust latent representation that can be used for classification.

However the benefit becomes less clear when the data distributions is more complex, such as the SVHN. For instance the SAAE and AAE perform equally well, even when the training set was reduced. It was also interesting to note that GSVAE actually outperformed the SAAE in the more complex distribution, indicating that explicit KL regularization was beneficial when considering more complex data distributions.

5.2 Critique and Further work

The architecture in the models was purposefully kept simple to limit the potential hyper-parameters and therefore computation constraints of the experiments. However the architecture did appear to limit the models ability to perform on the more complex SVHN dataset. Therefore further work would including repeat the experiment with convolutional layers rather than fully connected. The SVHN results were also based on a smaller modified dataset. It would be worth conducted further experiments on the full training set.

Another point to note about the SAAE (and AAE) is that they have four networks to train which is required more computation along with more hyper-parameter tuning. One method to reduce this cost would be to use only one discriminator to discriminate between the joint y and z samples, however it would be interesting to develop other ways to reduce the complexity of the models.

Bibliography

[1] Philip Bachman, Ouais Alsharif, and Doina Precup. "Learning with pseudo-ensembles". In: *Advances in Neural Information Processing Systems*. 2014, pp. 3365–3373.

[2] Pierre Baldi and Kurt Hornik. "Neural networks and principal component analysis: Learning from examples without local minima". In: *Neural networks* 2.1 (1989), pp. 53–58.

[3] David Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

[4] Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation learning: A review and new perspectives". In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1798–1828.

[5] Yoshua Bengio et al. "Deep generative stochastic networks trainable by backprop". In: *International Conference on Machine Learning*. 2014, pp. 226–234.

[6] David Berthelot, Tom Schumm, and Luke Metz. "Began: Boundary equilibrium generative adversarial networks". In: *arXiv preprint arXiv:1703.10717* (2017).

[7] Chris M Bishop. "Training with noise is equivalent to Tikhonov regularization". In: *Neural computation* 7.1 (1995), pp. 108–116.

[8] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. "Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]". In: *IEEE Transactions on Neural Networks* 20.3 (2009), pp. 542–542.

[9] Balázs Csanád Csáji. "Approximation with artificial neural networks". In: *Faculty of Sciences, Etvs Lornd University, Hungary* 24 (2001), p. 48.

[10] George Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals, and Systems (MCSS)* 2.4 (1989), pp. 303–314.

[11] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. "Adversarial feature learning". In: *arXiv preprint arXiv:1605.09782* (2016).

[12] Vincent Dumoulin et al. "Adversarially learned inference". In: *arXiv preprint arXiv:1606.00704* (2016).

[13] Yarin Gal and Zoubin Ghahramani. "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning". In: *international conference on machine learning*. 2016, pp. 1050–1059.

[14] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 2010, pp. 249–256.

[15] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[16] EJ Gumbel. "The maxima of the mean largest value and of the range". In: *The Annals of Mathematical Statistics* (1954), pp. 76–84.

[17] Emil Julius Gumbel. "Les valeurs extrêmes des distributions statistiques". In: *Ann. Inst. Henri Poincaré* 5.2 (1935), pp. 115–158.

[18] Michael U Gutmann and Aapo Hyvärinen. "Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics". In: *Journal of Machine Learning Research* 13.Feb (2012), pp. 307–361.

[19] Mark Herbster. *Lecture notes Supervised Learning*. Oct. 2016.

[20] Geoffrey E Hinton and Ruslan R Salakhutdinov. "Reducing the dimensionality of data with neural networks". In: *science* 313.5786 (2006), pp. 504–507.

[21] Geoffrey E Hinton et al. "Improving neural networks by preventing co-adaptation of feature detectors". In: *arXiv preprint arXiv:1207.0580* (2012).

[22] Sergey Ioffe and Christian Szegedy. "Batch normzation: Accelerating deep network training by reducing internal covariate shift". In: *International Conference on Machine Learning*. 2015, pp. 448–456.

[23] Eric Jang, Shixiang Gu, and Ben Poole. "Categorical reparameterization with gumbel-softmax". In: *arXiv preprint arXiv:1611.01144* (2016).

[24] Thorsten Joachims. "Transductive inference for text classification using support vector machines". In: *ICML*. Vol. 99. 1999, pp. 200–209.

[25] Theofanis Karaletsos. "Adversarial message passing for graphical models". In: *arXiv preprint arXiv:1612.05048* (2016).

[26] Diederik Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[27] Diederik P Kingma. "Fast gradient-based inference with continuous latent variable models in auxiliary form". In: *arXiv preprint arXiv:1306.0733* (2013).

[28] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[29] Diederik P Kingma et al. "Semi-supervised learning with deep generative models". In: *Advances in Neural Information Processing Systems*. 2014, pp. 3581–3589.

[30] Anders Krogh and John A Hertz. "A simple weight decay can improve generalization". In: *Advances in neural information processing systems*. 1992, pp. 950–957.

[31] Samuli Laine and Timo Aila. "Temporal Ensembling for Semi-Supervised Learning". In: *arXiv preprint arXiv:1610.02242* (2016).

[32] Anders Boesen Lindbo Larsen et al. "Autoencoding beyond pixels using a learned similarity metric". In: *arXiv preprint arXiv:1512.09300* (2015).

[33] Yann LeCun. "The MNIST database of handwritten digits". In: *http://yann. lecun. com/exdb/mnist/* (1998).

[34] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[35] Lars Maaløe et al. "Auxiliary deep generative models". In: *arXiv preprint arXiv:1602.05473* (2016).

[36] Laurens van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE". In: *Journal of Machine Learning Research* 9.Nov (2008), pp. 2579–2605.

[37] Alireza Makhzani et al. "Adversarial autoencoders". In: *arXiv preprint arXiv: 1511.05644* (2015).

[38] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[39] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. "Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks". In: *arXiv preprint arXiv:1701.04722* (2017).

[40] Takeru Miyato et al. "Virtual Adversarial Training: a Regularization Method for Supervised and Semi-supervised Learning". In: *arXiv preprint arXiv:1704.03976* (2017).

[41] Shakir Mohamed. *Lecture notes Advanced Topic in Machine Learning*. Mar. 2017.

[42] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.

[43] Yuval Netzer et al. "Reading digits in natural images with unsupervised feature learning". In: *NIPS workshop on deep learning and unsupervised feature learning*. Vol. 2011. 2. 2011, p. 5.

[44] Anh Nguyen et al. "Plug & play generative networks: Conditional iterative generation of images in latent space". In: *arXiv preprint arXiv:1612.00005* (2016).

[45] Frank Nielsen and Vincent Garcia. "Statistical exponential families: A digest with flash cards". In: *arXiv preprint arXiv:0911.4863* (2009).

[46] Augustus Odena. "Semi-supervised learning with generative adversarial networks". In: *arXiv preprint arXiv:1606.01583* (2016).

[47] Ning Qian. "On the momentum term in gradient descent learning algorithms". In: *Neural networks* 12.1 (1999), pp. 145–151.

[48] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015).

[49] Antti Rasmus et al. "Semi-supervised learning with ladder networks". In: *Advances in Neural Information Processing Systems*. 2015, pp. 3546–3554.

[50] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. "Stochastic backpropagation and approximate inference in deep generative models". In: *arXiv preprint arXiv:1401.4082* (2014).

[51] Salah Rifai et al. "Contractive auto-encoders: Explicit invariance during feature extraction". In: *Proceedings of the 28th international conference on machine learning (ICML-11)*. 2011, pp. 833–840.

[52] Mihaela Rosca et al. "Variational Approaches for Auto-Encoding Generative Adversarial Networks". In: *arXiv preprint arXiv:1706.04987* (2017).

[53] Frank Rosenblatt. "The perceptron: A probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[54] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. "Learning representations by back-propagating errors". In: *Cognitive modeling* 5.3 (1988), p. 1.

[55] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. "Regularization with stochastic transformations and perturbations for deep semi-supervised learning". In: *Advances in Neural Information Processing Systems*. 2016, pp. 1163–1171.

[56] Claude E Shannon. "A mathematical theory of communication". In: *ACM SIGMOBILE Mobile Computing and Communications Review* 5.1 (2001), pp. 3–55.

[57] sklearn. *t-distributed Stochastic Neighbor Embedding.* 2017. URL: `http://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html` (visited on 09/30/2010).

[58] Jost Tobias Springenberg. "Unsupervised and semi-supervised learning with categorical generative adversarial networks". In: *arXiv preprint arXiv:1511.06390* (2015).

[59] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting." In: *Journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[60] Richard S Sutton. "Two problems with backpropagation and other steepest-descent learning procedures for networks". In: *Proc. 8th annual conf. cognitive science society*. Erlbaum. 1986, pp. 823–831.

[61] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. "It Takes (Only) Two: Adversarial Generator-Encoder Networks". In: (2017). eprint: `arXiv:1704.02304`.

[62] Pascal Vincent et al. "Extracting and composing robust features with denoising autoencoders". In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 1096–1103.

[63] Pascal Vincent et al. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion". In: *Journal of Machine Learning Research* 11.Dec (2010), pp. 3371–3408.

[64] Stefan Wager, Sida Wang, and Percy S Liang. "Dropout training as adaptive regularization". In: *Advances in neural information processing systems*. 2013, pp. 351–359.

[65] Yair Weiss, Antonio Torralba, and Rob Fergus. "Spectral hashing". In: *Advances in neural information processing systems*. 2009, pp. 1753–1760.

[66] Jun-Yan Zhu et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks". In: *arXiv preprint arXiv:1703.10593* (2017).

[67] Xiaojin Zhu and Zoubin Ghahramani. “Learning from labeled and unlabeled data with label propagation”. In: (2002).

A Derivation

This derivation has been taken from [41].

Marginal likelihood

$$p(x) = \int p_\theta(x, z)\ dz$$

Conditional

$$p(x) = \int p_\theta(x|z)p(z)\ dz$$

Importance samples

$$p(x) = \int p_\theta(x|z)p(z)\frac{q_\phi(z|x)}{q_\phi(z|x)}\ dz$$

Rearrange

$$p(x) = \int q_\phi(z|x)p_\theta(x|z)\frac{p(z)}{q_\phi(z|x)}\ dz$$

Take logs

$$\log\ p(x) = \log q_\phi(z|x) \int p_\theta(x|z)\frac{p(z)}{q_\phi(z|x)}\ dz$$

Using Jensen's inequality that $\log \int p(x)g(x)dx \geq \int p(x)\log\ g(x)dx$

$$\log\ p(x) \geq \int q_\phi(z|x)\ \log\left(p_\theta(x|z)\frac{p(z)}{q_\phi(z|x)}\right)\ dz$$

Rearrange

$$\log\ p(x) \geq \int q_\phi(z|x)\ \log\ p_\theta(x|z) + \int q_\phi(z|x)\ \log\left(\frac{p(z)}{q_\phi(z|x)}\right)\ dz$$

Express and likelihood and KL diverence

$$\log\ p(x) \geq \mathbb{E}_{q_\phi(z|x)}\ \big[\log\ p_\theta(x|z)\big] - \mathrm{KL}\big[q_\phi(z|x)\ ||\ p(z)\big]$$

B Experiment results

Table B.1: Average semi-supervised classification errors and regularization statistics of the SAAE on the MNIST dataset as the size of the dataset is reduced

		Latent y stats		**Latent z stats**	
SAAE	**Error (%)**	y_μ	y_σ	z_μ	z_σ
Idea stats	Bayes [19]	0.10	0.30	0.00	1.00
60k	3.64	0.10	0.30	0.00	0.99
50k	4.33	0.10	0.30	0.00	0.99
40k	6.35	0.10	0.30	0.00	0.99
30k	6.86	0.10	0.30	0.00	0.99
20k	9.66	0.10	0.30	0.00	0.99
10k	12.17	0.10	0.29	0.00	0.97
5k	13.28	0.10	0.29	0.00	0.98

Table B.2: Average semi-supervised classification errors and regularization statistics of the AAE on the MNIST dataset as the size of the dataset is reduced

		Latent y stats		**Latent z stats**	
AAE	**Error (%)**	y_μ	y_σ	z_μ	z_σ
Idea stats	Bayes	0.10	0.30	0.00	1.00
60k	3.78	0.10	0.30	-0.01	0.98
50k	4.05	0.10	0.30	0.00	0.98
40k	6.04	0.10	0.30	0.00	0.99
30k	6.63	0.10	0.30	0.00	0.98
20k	10.45	0.10	0.30	0.00	0.97
10k	13.20	0.10	0.29	0.00	0.97
5k	15.37	0.10	0.29	0.00	0.97

Table B.3: Average semi-supervised classification errors and regularization statistics of the GSVAE on the MNIST dataset as the size of the dataset is reduced

		Latent y stats		**Latent z stats**	
GSVAE	**Error (%)**	y_μ	y_σ	z_μ	z_σ
Idea stats	Bayes	0.10	0.30	0.00	1.00
60k	6.18	0.10	0.30	-0.01	1.00
50k	7.89	0.10	0.30	0.00	0.99
40k	10.12	0.10	0.30	0.00	0.99
30k	12.18	0.10	0.30	0.00	1.00
20k	12.50	0.10	0.30	0.00	0.99
10k	14.09	0.10	0.29	0.00	0.99
5k	16.68	0.10	0.29	0.00	1.01

Table B.4: Average semi-supervised classification errors and regularization statistics of the SAAE on the SVHN dataset as the size of the dataset is reduced

		Latent y stats		**Latent z stats**	
SAAE	**Error (%)**	y_μ	y_σ	z_μ	z_σ
Idea stats	Bayes	0.10	0.30	0.00	1.00
60k	37.41	0.10	0.27	0.01	1.15
50k	37.95	0.10	0.27	0.00	1.16
40k	38.04	0.10	0.27	0.01	1.13
30k	38.13	0.10	0.27	-0.01	1.09
20k	38.36	0.10	0.26	0.00	1.13
10k	38.76	0.10	0.25	0.02	1.51
5k	40.94	0.10	0.21	-0.01	1.93

Table B.5: Average semi-supervised classification errors and regularization statistics of the AAE on the SVHN dataset as the size of the dataset is reduced

		Latent y stats		**Latent z stats**	
AAE	**Error (%)**	y_μ	y_σ	z_μ	z_σ
Idea stats	Bayes	0.10	0.30	0.00	1.00
60k	37.42	0.10	0.27	-0.01	1.16
50k	37.96	0.10	0.27	-0.01	1.17
40k	37.92	0.10	0.27	-0.01	1.13
30k	38.33	0.10	0.27	0.00	1.07
20k	38.62	0.10	0.26	-0.01	1.08
10k	38.81	0.10	0.25	0.00	1.41
5k	40.90	0.10	0.21	0.03	1.96

Table B.6: Average semi-supervised classification errors and regularization statistics of the GSVAE on the SVHN dataset as the size of the dataset is reduced

		Latent y stats		**Latent z stats**	
GSVAE	**Error (%)**	y_μ	y_σ	z_μ	z_σ
Idea stats	Bayes	0.10	0.30	0.00	1.00
60k	37.57	0.10	0.27	0.00	1.02
50k	37.60	0.10	0.27	0.00	1.02
40k	37.63	0.10	0.27	0.00	1.03
30k	37.85	0.10	0.26	0.00	1.03
20k	37.88	0.10	0.26	0.00	1.03
10k	37.96	0.10	0.25	0.00	1.02
5k	40.24	0.10	0.21	0.00	1.96