Deep Generative Models for Semi-Supervised Machine Learning

Maaløe, Lars

Publication date:
2018

Document Version
Publisher's PDF, also known as Version of record

Deep Generative Models for Semi-Supervised Machine Learning

Lars Maaløe

Kongens Lyngby 2018
PhD-2018-472

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Richard Petersens Plads, building 324,
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Summary (English)

The reintroduction of deep neural networks has a large impact on the modeling capabilities of modern machine learning. This reignites the general public's dream of achieving artificial intelligence, and spawns rapid progress in large-scale industrial machine learning development, such as autonomous driving. However, the leaps in development are still confined to a rather limited learning domain, in which labeled data is required. Labeled data is hard and costly to acquire, due to the amount needed to efficiently learn a modern machine learning model, and that many data sources are not directly interpretable. Consequently, research in different learning paradigms that utilize vast amounts of unlabeled data is getting more and more attention. Albeit possessing intriguing theoretical properties, machine learning models that learn from unlabeled data are still an unsolved research topic.

The thesis comprises methods that utilize the power of deep neural networks to learn from both labeled and unlabeled data. A background for the theoretical foundation of the proposed methods are described and empirical results showing their capabilities within generation and classification tasks are presented. Finally, a real-life application within condition monitoring for sustainable energy is demonstrated, proving that the proposed methods have the expected impact and are applicable.

Summary (Danish)

Re-introduktionen af dybe neurale netværk har haft stor indflydelse på modelleringskapaciteten i den moderne maskinlæring. Det har aktualiseret offentlighedens efterspørgsel af kunstig intelligens, hvilket har medført signifikante fremskridt i den industrielle udvikling af maskinlæring, for eksempel inden for selvkørende biler. Udviklingen har dog været begrænset til et indskrænket læringsdomæne, hvori der kræves annoteret data. Store mængder af annoteret data er vanskelige og dyre at erhverve, og det er ikke muligt at fortolke mange datakilder for at skabe en moderne maskinlæringsmodel. Det resulterer i, at forskning inden for andre læringsparadigmer, der udnytter store mængder ikke-annoteret data, får stadig større opmærksomhed. På trods af at maskinlæringsmodeller der lærer fra ikke-annoteret data, har spændende teoretiske egenskaber, er det dog stadig et åbent forskningsfelt.

I denne afhandling præsenterer vi metoder, der udnytter mulighederne i neurale netværk til at lære fra både annoteret og ikke-annoteret data. Vi giver en baggrund for det teoretiske grundlag for de foreslåede metoder og præsenterer empiriske resultater, der viser metodernes evne inden for genererings- og klassifikationsopgaver. Endelig præsenterer vi en applikation inden for bæredygtig energiovervågning, hvori vi viser, at de foreslåede metoder virker.

Preface

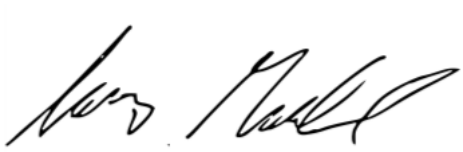
This thesis was prepared at the Department of Applied Mathematics and Computer Science, Technical University of Denmark, in fulfilment of the requirements for acquiring a PhD in Engineering.

During the PhD, a research stay was conducted at Apple Inc., Cupertino, California, USA. The research considered machine learning in autonomous systems, from which the findings are not disclosed in this thesis.

The thesis was funded by the Technical University of Denmark and Innovation Fund Denmark with guidance under Professor Ole Winther. The work was carried out between December 15, 2014 and March 17, 2018.

The thesis consists of 5 research papers.

17-March-2018



Lars Maaløe

Contributions

Papers included in thesis

A Maaløe, L., Sønderby, C. K., Sønderby, S. K., Winther, O. (2016). Auxiliary deep generative models. In *Proceedings of the International Conference on Machine Learning*, pages 1445–1454.

B Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., Winther, O. (2016). Ladder variational autoencoders. In *Advances in Neural Information Processing Systems*, pages 3738-3746.

C Maaløe, L., Fraccaro, M., Winther, O. (2017). CaGeM: A cluster aware deep generative model. In *Neural Information Processing Systems Workshop* on Approximate Bayesian Inference.

D Maaløe, L., Spataru, S. V., Sera, D., Winther, O. (2018). Condition monitoring in photovoltaic systems by semi-supervised machine learning. Submitted to *IEEE Transactions of Industrial Informatics*.

E Maaløe, L., Winther, O. (2018). Feature map variational auto-encoders. To be submitted.

Papers not included in thesis

(I) Maaløe, L., Arngren, M., Winther, O. (2014). Deep belief nets for topic modeling. *International Conference of Machine Learning Workshop* on Knowledge-Powered Deep Learning for Text Mining.

(II) Maaløe, L., Sønderby, C. K., Sønderby, S. K., Winther, O. (2015). Improving semi-supervised learning with auxiliary deep generative models. In *Neural Information Processing Systems Workshop* on Approximate Bayesian Inference workshop.

(III) Sønderby, S. K., Sønderby, C. K., Maaløe, L., Winther, O. (2015). Recurrent spatial transformer networks. arXiv preprint arXiv:1509.05329.

(IV) Spataru, S. V., Gavriluta, A., Sera, D., Maaløe, L., Winther, O. (2016). Development and implementation of a PV performance monitoring system based on inverter measurements. *IEEE Energy Conversion Congress and Exposition* (ECCE), 1-7.

(V) Tax, T. M. S., Antich, J. L. D., Purwins, H., Maaløe, L. (2017). Utilizing domain knowledge in end-to-end audio processing. In *Neural Information Processing Systems Workshop* on Machine Learning for Audio.

(VI) Parachiv, M., Borgholt, L., Tax, T. M. S., Singh, M., Maaløe, L. (2017). Exploiting nontrivial connectivity for automatic speech recognition. In *Neural Information Processing Systems Workshop* on Machine Learning for Audio.

Other contributions

Co-authored a large amount of teaching material throughout the PhD. They can all be found on the Github account named *DeepLearningDTU*:

(i) Exercises on neural networks, Bayesian neural networks, variational auto-encoders and ladder networks for course on *Advanced Topics in Machine Learning*, Technical University of Denmark, 2015.
`github.com/DeepLearningDTU/Summerschool_2015`

(ii) Exercises on semi-supervised learning for summerschool on *semi-supervised learning for image analysis and computer graphics*, Technical University of Denmark and University of Copenhagen, 2016.
`github.com/DeepLearningDTU/variational-autoencoders-summerschool-2016`

(iii) Various material for the the course on *Deep Learning*, Technical University of Denmark, 2016-2017.
`github.com/DeepLearningDTU/02456-deep-learning`

Developed Python libraries built upon Theano (Bastien et al., 2012), Lasagne (Dieleman et al., 2015), and Tensorflow (Abadi et al., 2015) as frameworks for implementing variational neural network models for semi-supervised and unsupervised learning:

(iv) Sønderby, C. K., Sønderby, S. K., Maaløe, L., *Parmesan.*
`github.com/casperkaae/parmesan`

(v) Maaløe, L., *Auxiliary Deep Generative Models.*
`github.com/larsmaaloee/auxiliary-deep-generative-models`

(vi) Maaløe, L., *Variational Tensorflow.*
`github.com/larsmaaloee/variational-tensorflow`

Acknowledgements

First and foremost I would like to thank my supervisor, Professor Ole Winther, for giving me the chance to embark on this journey, and for providing highly professional guidance and collaboration throughout the research. This has led to a PhD study that has been both fun and challenging.

I would also like to thank my research collaborators: Søren K. Sønderby, Casper K. Sønderby, Tapani Raiko, Marco Fraccaro, and my PhD colleagues: Rasmus B. Palm, Simon D. Kamronn, and Anders B. L. Larsen, for interesting discussions and learnings throughout the research.

Thanks to the Technical University of Denmark and Innovation Fund Denmark for funding the research.

Last but not least, I would like to thank Iben Thomsen, for her eternal support throughout these studies.

Contents

Chapter 1

Introduction

A problem domain can be characterized as one where the boundaries that systematically explain the domain are not understood. For some human beings, a problem domain could be the game of *tic-tac-toe* and for most it would be the game of *chess*, thus it is one where an analytical solution is intractable for the subject at hand. Explicitly programmed machine algorithms are great means of solving a problem domain, but they quickly become intractable when the dimensionality and possible outcomes get too large. In *machine learning* we define computational models that learn from data. The promise is that these models are able to solve, or at least approximate, a solution to very complex problem domains. However, in a large amount of domains, for which the human cognition naturally thrives, such as facial and speech recognition, the traditional modeling approaches come to a halt.

Deep learning has revived the hype of artificial intelligence by spawning an unprecedented increase in performance within machine learning. The fundamental basis of deep learning lies within the *neural network*, which is a model that allows for a *deep* stacking of computational layers. The hierarchy of layers enables learning of local as well as global representations of data (LeCun et al., 2015). The main breakthroughs in deep learning concern natural image and language modeling, where we have seen rapid improvements in the ability to capture patterns from very complex data distributions. Amongst the most impressive breakthroughs stand the improvements in computer vision (Krizhevsky et al.,

2012), natural language processing (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014), audio processing (Amodei et al., 2016; van den Oord et al., 2016a), and reinforcement learning (Mnih et al., 2015). A general conception is that it all spawned from Hinton et al. (2006) that illuminated how one could learn a *deep* stacking of neural network layers. However, the invention of a neural network is much older and can be attributed to research on the perceptron model[1] (Rosenblatt, 1958) on which Rumelhart et al. (1986) introduced a way of applying backpropagation to learn a hierarchy of internal representations[2]. Conversely to attributing the deep learning movement to a single event, we believe that it is a result of a plethora of inventions, introduced in order to solve the caveats that have impeded the performance of neural networks through time. Many of the inventions with the highest impact can be attributed as *tricks* that are easily accompanied by each other, e.g. non-linear functions (Glorot et al., 2011), simple regularization (Srivastava et al., 2014), and optimization schemes (Kingma and Ba, 2014). Other inventions have achieved to reinvent the architecture of the neural network in order to successfully embed temporal (Hochreiter and Schmidhuber, 1997) and spatial (LeCun et al., 1999) structure. In cohesion with the scientific research, rapid development in computer resources impact the size of the models that can be allocated in computer memory and the amount of time it takes to learn them.

Recent neural network research presents models with the ability to discriminate between images on par with the human visual cortex (He et al., 2016), transcribe conversational speech better than the human ear (Xiong et al., 2017), and outperform the world champion in the ancient game of Go (Silver et al., 2016). However, due to theoretical limitations in the applied machine learning frameworks, we are far from the final frontier in the development of useful machine learning models[3].

> *What I cannot create, I do not understand.*
>
> —Richard Feynman

Most of the groundbreaking results are still confined to the area of discriminative modeling, in which the neural networks learn a deterministic mapping between an input and a target, e.g. a natural image and its corresponding category. This paradigm is referred to as *supervised learning*. There exist two constraints in

[1] A neural network and multi-layer perceptron are often referred to interchangeably.

[2] Rumelhart et al. (1986) are not the first account of backpropagation. However, it is a publication that achieved tremendous traction since it presented the learning of internal representations by applying backpropagation. Cf. (Schmidhuber, 2014) for a detailed discussion.

[3] Inspiration for quote from Open AI blog: blog.openai.com/generative-models.

this formulation. One is that the models require vast amounts of labeled data, such as millions of labeled natural images or thousands of transcribed hours of audio from a highly varied data distribution in order to learn efficiently. One might say that the annotation process has become slightly less cumbersome by *crowd-sourced* solutions, however, these solutions still require large quantities of manual labor and that the type of data is directly interpretable, e.g. natural images or audio. Many interesting applications of machine learning lie within a problem domain where the data is not directly interpretable. This could be genome sequences, sensor data from wind turbines, or agricultural data of crop diseases. Another constraint when using neural networks as discriminative models, is that they have a tendency to be overly confident when presented to data that is significantly different than what was learned on (Nguyen et al., 2015; Gal, 2016; Carlini and Wagner, 2017). Many problem domains have a tolerance for faulty predictions, but what happens when we rely on image segmentation models for predicting cancer, or when we trust the next action made by the autonomous vehicle? It is simply infeasible to acquire enough labeled data in order to ascertain that one covers the entire problem domain, which is why we must learn more about where the data comes from. Gal (2016) refers to this as: *the importance of knowing what we don't know*. In order to achieve this, we tend to the *unsupervised learning* paradigm in which we learn from unlabeled data.

Conversely to discriminative models, generative models are typically defined as probabilistic models and have the ability to learn a distribution of the observed data, which in turn results in the ability to *create*, or more concisely, draw observations from a learned distribution. Generative models bear resemblance to how humans are learning, since both are able to learn from a single data distribution, e.g. the positive category of a binary classification problem, whereas its counterpart will have to learn from both positive and negative examples in order to discriminate (Xu and Tenenbaum, 2007; Murphy, 2012). The theoretical advantage of generative models boils down to the fact that they learn the data distribution and the adhering uncertainty. This results in the potential to discern overly confident predictions and opens for a wide variety of applications within machine learning, such as natural image generation and speech synthesis. The three most popular approaches to generative models are: *generative adversarial networks* (GAN) (Goodfellow et al., 2014), *autoregressive models* such as the PixelRNN (van den Oord et al., 2016b), and *probabilistic deep generative models* such as the *variational auto-encoder* (VAE) (Kingma and Welling, 2014; Rezende et al., 2014). The approaches are very different in their formulations, but all share an intriguing potential in the fact that they can learn from both labeled and unlabeled data. This learning paradigm is called *semi-supervised learning*, and that, in combination with the probabilistic deep generative modeling approach, are the main foci of the research presented in this thesis. In the remainder of the thesis a generative model refers to the probabilistic variant.

1.1 Probabilistic Generative Models

Defining a generative model may seem as the obvious choice from a theoretical perspective, but it is often difficult to formulate it in an adequately expressive way. At the core of a generative model lies the definition of a probability distribution $p(x)$, for which $p(x) \geq 0$ and $\int_{-\infty}^{\infty} p(x)dx = 1$. The rules of probability are defined as the *sum rule*: $p(x) = \int_y p(x, y)dy$, and the *product rule*: $p(x, y) = p(y|x)p(x)$, where x, y are two random variables, $p(x)$ is the probability distribution for the specific variable, $p(x, y)$ the joint probability, and $p(x|y)$ the conditional probability. We have defined the random variables as continuous, hence the integral, but the rules also apply to discrete variables. By utilizing the rules of probability we define Bayes theorem:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} = \frac{p(x|y)p(y)}{\int_y p(x|y)p(y)dy} \ , \tag{1.1}$$

which explains the relationship between the two conditional probabilities, $p(x|y)$ and $p(y|x)$ (Bishop, 2006). In this setting, $p(y|x)$ defines the *posterior*, $p(x|y)$ the *likelihood*, $p(y)$ the *prior*, and $p(x)$ the *evidence*. While omitting the model parameters, the above formulation of Bayes theorem, can be seen as the simplest formulation of a generative model for supervised classification, known as the *naive Bayes classifier*. Recall that in a supervised machine learning model we seek to learn a mapping between an input x and a category y. The naive Bayes classifier learns to map $p(y|x)$, where this mapping depends on the reverse conditional $p(x|y)$, from which the observed variable can be generated from a category. Given a dataset of N examples, $\mathbf{x} = x_1, ..., x_N$, $\mathbf{y} = y_1, ..., y_N$, we can thereby either infer a category for an example in $\mathbf{x}$ or an example from a category in $\mathbf{y}$.

In complex problem domains, it is not trivial to learn the mapping $p(x|y)$. E.g. if the input distribution is defined as natural images expressed by pixels of the dimension: $width \times height \times channels$, it quickly becomes intractable to learn the evidence w.r.t. the input distribution. In the discriminative models, introduced in the previous section, the model definition is *loosened*, by only learning the mapping $p(y|x)$. Empirical results show that this approach works significantly better for supervised classification tasks (Chapelle et al., 2010), but as mentioned earlier, they are also prone to overly confident predictions. It may be argued that the limitation of a generative model lies in the limited *expressive power* of the mapping function $f(x; y) = p(x|y)$. As we will see later, neural networks provide the ability for a much richer representation of the data, which provides the flexibility to model this complex mapping through non-linear functions.

Conversely to the discriminative model, a generative model can also be applied

in unsupervised learning. In unsupervised learning a common approach is to learn a latent representation directly from the input data distribution, thus no labeled data is needed. A good latent representation achieves to find global features of the input distribution that can be used for *clustering*, *anomaly detection*, *denoising*, and much more. In the context of probabilistic modeling for unsupervised learning problems, we can introduce a latent variable z, instead of the label y, to the generative modeling framework, such that:

$$p(z|x) = \frac{p(x|z)p(z)}{\int_z p(x|z)p(z)dz} \ . \tag{1.2}$$

The latent variable is often defined so that it is of a lower dimensionality than the input. Thereby, the data represented in the latent variable is less prone to the *curse of dimensionality*, which is a phenomenon in which data becomes sparse in high dimensions, so that the properties from low-dimensional data, such as measures of similarity, are intractable. There exist many approaches to estimate the latent variable, such that it explains the input distribution well. This will be explained further in Chapter 3. Learning a good latent representation of an input data distribution opens for a wide variety of applications.

1.2 Semi-Supervised Learning

The modeling of a *good* latent representation of the input distribution, leads us to the intriguing properties of generative models in semi-supervised machine learning. In semi-supervised machine learning we seek to alleviate the problem of annotating data, by learning from a small fraction of labeled data and a large fraction of unlabeled data. To accomplish this, we must acquire learnings on $p(x)$ from the unlabeled data that can support the mapping $p(y|x)$, and vice versa. If the labeled and unlabeled data is somehow disjoint, meaning that they come from different distributions, we cannot expect to gain value from a semi-supervised learning framework, and must resort to purely supervised or unsupervised learning. This does not necessarily mean that the unlabeled and labeled data must share direct similarity in the input dimension. Thus, for complex problem domains we cannot expect that the unlabeled data *lies* within the same clusters as the labeled data, due to the curse of dimensionality. Therefore, we propose the *manifold assumption* that states that the high-dimensional labeled and unlabeled data lies on a low-dimensional manifold (Chapelle et al., 2010), on which we will not be prone to the curse of dimensionality. This corresponds nicely to the latent variable z from the probabilistic generative model. If we can infer a latent variable that encapsulates the shared properties we can gain value from utilizing semi-supervised learning. From this prerequisite, we can easily formulate a generative model that encapsulates the variables for

semi-supervised learning by applying Bayes theorem to the joint probability, $p(x, y, z)$:

$$p(y, z|x) = \frac{p(x|y, z)p(y, z)}{\int_z \int_y p(x|y, z)p(y, z)dydz} \ . \tag{1.3}$$

However, as we shall see later, solving the above is not straightforward, due to the integral that quickly becomes intractable for complex problem domains.

1.3 Thesis outline

In the first part of this thesis we will present a brief background on neural networks in the context of deep learning and representation learning. Neural networks are used as parameterizations throughout the remainder of the modeling frameworks proposed in this thesis, but will not be further explored. For an in-depth review on deep learning and representation learning we refer to Bengio et al. (2013); LeCun et al. (2015) and Goodfellow et al. (2016).

In the second part of the thesis, we present the background on generative models based on deep learning, denoted deep generative models. The primary goal is to provide a foundation to the research undertaken in this study and therefore this should not be perceived as a comprehensive review.

The third and fourth part of the thesis concerns the proposed methodological research of the thesis within semi-supervised and unsupervised deep generative models. Here we will present the *auxiliary deep generative model* (Maaløe et al., 2016) (cf. **Appendix A**) for unsupervised and semi-supervised learning. We will also present the *ladder variational auto-encoder* (Sønderby et al., 2016) (cf. **Appendix B**) for unsupervised learning. Next we formulate a model for improving generative modeling by utilizing a fraction of labeled information, denoted *cluster-aware deep generative models* (Maaløe et al., 2017) (cf. **Appendix C**). We extrapolate over improving the unsupervised generative models, by presenting a research study, still in the process, in which we develop a combination of a deep generative model and an autoregressive model, denoted *feature map variational auto-encoders* (Maaløe and Winther, 2018) (cf. **Appendix E**).

Finally, we present an applied research study on the use of the proposed methods in condition monitoring for solar energy systems (Maaløe et al., 2018) (cf. **Appendix D**). We show that deep generative models can function for real-life fault detection, while also detecting outliers.

Within each chapter we clearly state the research contributions of this thesis.

We also refer to educational course material developed throughout the research, that will hopefully illuminate some of the more granular details. Throughout the thesis, we will not provide a comprehensive review of all results found in the articles. For this, we refer the reader to the original articles in the Appendices.

Chapter 2

Deep Neural Networks

In this chapter we give a brief introduction to neural networks with a special focus on the methods used throughout the research. We will treat neural networks as self-contained models, however, as we show in the following chapters, they can easily be deployed as learnable function approximations in other machine learning paradigms. First we explain the theory behind neural networks as discriminative models, in the supervised setting. Next we provide a perspective on the meaning of representation learning, followed by an introduction to deep learning models in unsupervised learning.

All models introduced are based on gradient descent optimization algorithms, for which we need a training set of N examples, $\mathbf{x} = x_1, ..., x_N$. In supervised learning, we also need the corresponding labels, $\mathbf{y} = y_1, ..., y_N$. We refer to a *mini-batch* of examples as a collection of B examples where $B \leq N$, denoted $\mathbf{x}_B$, $\mathbf{y}_B$. Model parameters are denoted θ and/or ϕ, and refers to the parameters that are learned during optimization in order to seek a better model performance.

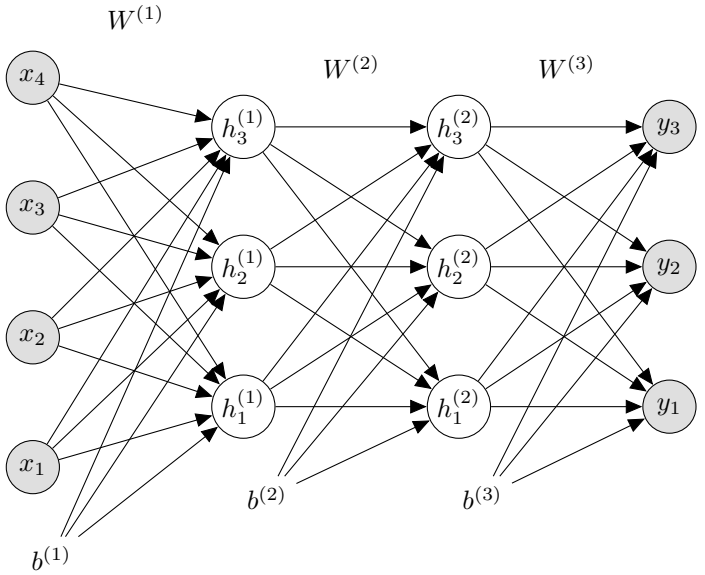


Figure 2.1: A graphical representation of depth $L = 3$ densely connected neural network with input variables $\mathbf{x}$, two hidden layers, $h^{(1)}$ and $h^{(2)}$, and the output variables $\mathbf{y}$. Subscripts denote the units. In this example the input dimensionality is reduced throughout the hidden layers and the neural network outputs 1-of-K classes, where $K = 3$.

2.1 Supervised Learning

A deep neural network consists of a number of non-linear activation functions, $l = 1, ..., L$, denoted hidden layers $h^{(l)}$, stacked on top of each other, so that one layer accepts the output of the layer below $h^{(l-1)}$ as input. Each layer consists of an activation function $f(\cdot)$, and learnable parameters, weight matrix $W^{(l)}$, and bias vector $b^{(l)}$, such that $h^{(l)} = f(W^{(l)}h^{(l-1)} + b^{(l)})$ (cf. Figure 2.1). The activation function of the last layer L is defined differently depending on the task at hand, $\hat{\mathbf{y}} = g(W^{(L)}h^{(L-1)} + b^{(L)})$. The stacking of the neural network layers follow:

$$h^{(0)} = \mathbf{x} \ , \tag{2.1}$$

$$h^{(l)} = f(W^{(l)}h^{(l-1)} + b^{(l)}) \ , \quad l = 1, ..., L-1 \ , \tag{2.2}$$

$$\hat{\mathbf{y}} = g(W^{(L)}h^{(L-1)} + b^{(L)}) \ . \tag{2.3}$$

We refer to the combined parameter space, $W^{(1)}, ..., W^{(L)}$ and $b^{(1)}, ..., b^{(L)}$ as θ. As mentioned in Chapter 1, there exist multiple variants beside the one depicted

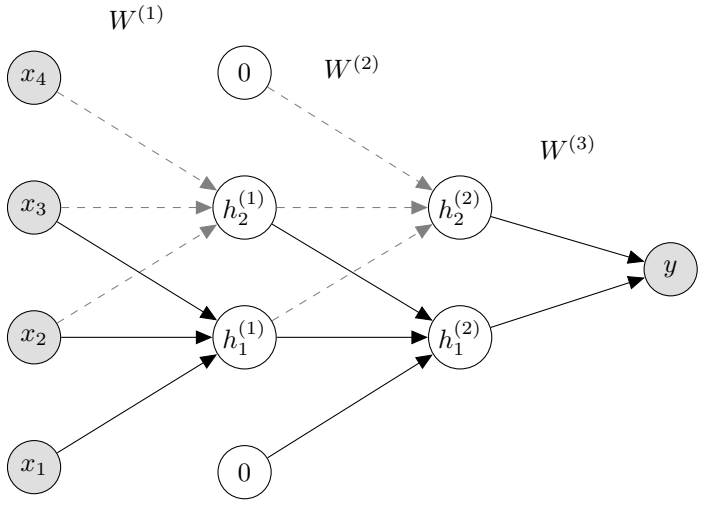


Figure 2.2: A graphical representation of a depth $L = 3$, 1D convolutional neural network with 1 channel/feature-map, input variables $\mathbf{x}$, two hidden layers, $h^{(1)}$ and $h^{(2)}$, and the output variable y, represented by a scalar. In this example the input dimensionality is reduced throughout the hidden layers and the neural network outputs a binary classification. The solid lines represent the kernel matrix W (we embed the bias) that is reused within the network layers. The *width* of the kernel is 3 for for layers $l = 1$ and $l = 2$, where the final layer is densely connected. In the first transformation we apply no padding and in the second we apply a padding of 1.

in Figure 2.1. The main variants are the recurrent neural network (RNN) and convolutional neural network (CNN). This work considers the densely connected neural network and the convolutional neural network. In Figure 2.2 we give an example of a 1-dimensional convolutional neural network with only 1 channel and 1 feature map throughout the layers. The architecture is easily scaled to a 2 or 3-dimensional example with multiple feature maps that intend to capture different representations of the data. The CNN defines much fewer weights in each layer that are then shared. While lowering the amount of parameters, this triggers an ability to capture spatial structure in data.

For the neural network to learn useful representations, we must apply hidden layer activation functions $f(\cdot)$ that are non-linear in nature. One of the most well-known activation functions is the sigmoid function:

$$\sigma(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{x}}} \ , \tag{2.4}$$

which lies in the range $[0, 1]$ and has a smooth gradient. Both of these prop-

erties are intriguing for optimization, as we will see later. Another well-known activation function is the tanh that is in the range $[-1, 1]$:

$$\tanh(\mathbf{x}) = \frac{\sinh(x)}{\cosh(x)} = 2\sigma(2\mathbf{x}) - 1 \ . \tag{2.5}$$

However, due to the plateaus near the limits of these activation functions, they are both prone to the phenomenon called *vanishing gradients* that saturates the learning process. This is where the ReLU activation function, $f(x) = \max(0, x)$, comes handy, since it does not saturate, $[0, \infty]$. However, it possesses the vulnerability of activations blowing up. This can be controlled by other tricks, such as gradient clipping or batch normalization (Ioffe and Szegedy, 2015). ReLU units also result in sparse activations, because all negative values become 0 (gradient of 0). This can come at a cost, in which many units in the neural network become passive, *dying ReLU problem*, which have spawn other variants, such as the leaky ReLU. The ReLU and its variants have gained tremendous traction as being the best choice for a hidden layer activation function. However, due to the properties of a bounded output interval in the case of the sigmoid and tanh functions, they are still popular as output activation functions, $g(\cdot)$. In the case of a discriminative binary classification task the activation function is a sigmoid, and a neural network with one hidden layer is given by:

$$\hat{\mathbf{y}}_\theta(\mathbf{x}) = \sigma\left(W^{(2)} ReLU\left(W^{(1)}\mathbf{x} + b^{(1)}\right) + b^{(2)}\right) \ . \tag{2.6}$$

Besides the binary classification task there exist two major applications within deep neural networks for supervised learning: *multi-class classification* and *regression*. In multi-class classification we seek to solve whether $\mathbf{x}$ belongs to 1-of-K classes (cf. Figure 2.1). In this case, the output activation function is denoted the softmax function and for the k^{th} class, with $k = 1, ..., K$ it is:

$$\text{softmax}(\hat{\mathbf{y}})_k = \frac{e^{\hat{\mathbf{y}}_k}}{\sum_j^K e^{\hat{\mathbf{y}}_j}} \ . \tag{2.7}$$

The neural network in Figure 2.1 is defined as:

$$\hat{\mathbf{y}}_\theta(\mathbf{x}) = \text{softmax}\left(W^{(3)} ReLU\left(W^{(2)} ReLU\left(W^{(1)}\mathbf{x} + b^{(1)}\right) + b^{(2)}\right) + b^{(3)}\right) \ . \tag{2.8}$$

For continuous regression problems the output is a linear transformation.

In the supervised learning problem, the aim of the learning task is to minimize the negative likelihood between the target $\mathbf{y}$ and the model $\hat{\mathbf{y}}_\theta(\mathbf{x})$ over the parameters θ:

$$\arg\min_\theta - \prod_n p_\theta(y_n | \hat{y}_\theta(x_n)) \ . \tag{2.9}$$

We often prefer to define the problem with respect to the log-likelihood, since it (i) has the same maximum value as the likelihood, (ii) is monotonically increasing, and (iii) avoids overflow and underflow since multiplication and division becomes addition and subtraction in the log-space. In order to solve the optimization problem in Equation 2.9 we seek to define a negative log-likelihood, denoted the objective function $\mathcal{F}(\hat{\mathbf{y}}_\theta(\mathbf{x}), \mathbf{y})$, that solves the task at hand. In the binary classification task, the objective function is defined as the Bernoulli cross-entropy:

$$\mathcal{F}(\hat{\mathbf{y}}_\theta(\mathbf{x}), \mathbf{y}) = -\sum_n \left(y_n \log \hat{y}_\theta(x_n) + (1 - y_n) \log(1 - \hat{y}_\theta(x_n))\right) \ , \tag{2.10}$$

where $\hat{\mathbf{y}}_\theta(\cdot)$ is a scalar in the range $[0, 1]$. For a multi-class classification task the objective function is the Categorical cross-entropy:

$$\mathcal{F}(\hat{\mathbf{y}}_\theta(\mathbf{x}), \mathbf{y}) = -\sum_n y_n \log \hat{y}_\theta(x_n) \ , \tag{2.11}$$

where $\hat{\mathbf{y}}_\theta(\cdot)$ is a vector representing 1-of-K outcomes. Since neural networks mostly apply to complex datasets, there is often not an analytical solution to find the maximum likelihood parameter setting θ_{ML}. The problems solved with deep neural networks are non-convex problems and the parameter space is too big to feasibly compute second-order derivatives. Therefore, we resort to a step-wise first-order derivative optimization scheme. Stochastic gradient descent (SGD) optimization proves a way to optimize the parameter w.r.t. to a mini-batch $\mathbf{x}_b$ and $\mathbf{y}_b$. This is done by first calculating the objective function $\mathcal{F}(\hat{\mathbf{y}}_\theta(\mathbf{x})$, and propagate this error back through each layer (backpropagation) in order to calculate the gradient $\nabla_\theta \mathcal{F}(\hat{\mathbf{y}}_\theta(\mathbf{x}_b), \mathbf{y_b})$. This process is denoted *backpropagation* (Rumelhart et al., 1986).

Optimization is performed for a predefined amount of iterations, τ, where each update is given by:

$$\theta_{\tau+1} = \theta_\tau - \alpha \nabla_{\theta_\tau} \mathcal{F}(\hat{\mathbf{y}}_{\theta_\tau}(\mathbf{x}_b), \mathbf{y_b}) \ , \tag{2.12}$$

and α is the learning rate. Compared to the gradient descent (GD) optimization scheme that updates the parameters θ for the entire dataset $\mathbf{x}$, $\mathbf{y}$, SGD has a regularizing effect. There exist multiple further advancements to the regular SGD optimization scheme that adapt the learning rate, α, by utilizing adaptive estimates of lower-order moments. Throughout the research we use ADAM (Kingma and Ba, 2014).

For an in-depth tutorial on neural networks, go to the lab exercises:

`github.com/DeepLearningDTU/02456-deep-learning`

Densely connected neural networks: Lab1
A thorough walk-through of the backpropagation algorithm and a guide to setup a neural network with an accompanying introduction to optimization algorithms. The tutorial also provides an introduction to neural network regularization and the reverse effect, *overfitting*.

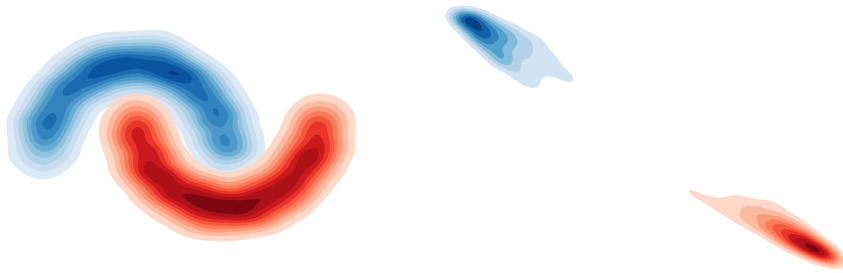
Convolutional neural networks: Lab2
Introduction to the concepts of convolutional neural networks, i.e. *stride*, *pooling*, and *padding*. The lab also provides a deep understanding of the arithmetics of both the densely connected and convolutional neural network with exercises on implementing the networks without libraries such as Tensorflow (Abadi et al., 2015). It proves practical learnings from posing examples on widely used machine learning datasets.

Recurrent neural networks: Lab3
A practical introduction to recurrent neural networks. The lab provides examples on how to implement different *encoder-decoder* structured networks with simple use-cases, such as implementing a network that maps from spelled numbers, i.e. *one, two, three*, to the numeric representation, i.e. *1, 2, 3*.

2.2 Representation learning

The learning process of each hidden layer throughout the neural network layers is referred to as *representation learning* (or *feature learning*), where a good representation is one that can untangle the input data distribution in order to improve on the training criterion at hand (Bengio et al., 2013). Let us consider a simple example hereof, in which we have a binary classification task consisting of two *half-moons*, where the upper half-moon belongs to the positive class and the lower half-moon to the negative class (cf. Figure 2.3). The two distributions are entangled in the input distribution and essentially impossible to discriminate between without a form of non-linearity.

Discriminating between the two half-moons is a simple task for a neural network, due to its ability to transform, i.e. rotate and resize, the data through its $W^{(l)}$ matrices and perform translations through its $b^{(l)}$ vectors. This means that,



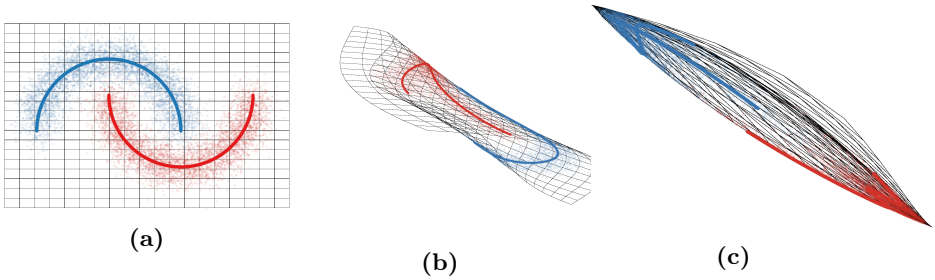
(a) (b)

Figure 2.3: (a) Density plot of a dataset consisting of two entangled half-moons in the input space. (b) A density plot of two transformations of the two entangled half-moons, in which they have been untangled. This can be considered as a representation of the input data distribution.

by stacking multiple layers on top of each other, the neural network gets more flexible in its ability to transform and translate, hence a higher expressive power. In order to show the effect, we draw a number of examples from the distributions and apply a mesh grid to the 2-dimensional half-moon example (cf. Figure 2.4a). To make the effect more clear, we also visualize two tendency lines, one for each class. We then apply a neural network with two hidden layers, using the tanh activation function, and an output layer with the sigmoid activation function with corresponding Bernoulli cross-entropy for the purpose of binary classification (cf. Equations 2.5, 2.4, 2.10). When training the neural network, we quickly see how the input data distribution transforms and translates, by visualizing the representation in one of the hidden layers (cf. Figure 2.4b). In the end of the learning phase, where this neural network achieves an accuracy of 100%, it is evident that it has accomplished to untangle the input space, such that it is possible, for the output layer, to linearly discriminate between the two half-moons (cf. Figure 2.4c).

2.3 Unsupervised Learning

In contrast to the clear definition of supervised learning, unsupervised learning covers an extensive amount of machine learning paradigms, e.g. dimensionality reduction, clustering, anomaly detection, and generative modeling. Amongst the famous machine learning models for unsupervised learning is: *principal component analysis* (PCA), *k-nearest neighbours* (KNN), *restricted Boltzmann machines* (RBM), and many more. In this thesis we will limit ourselves to unsupervised learning in the context of neural network auto-encoders.



(a) (b) (c)

Figure 2.4: Visualization of the half-moon dataset with a mesh grid in the input space (a), the beginning of training of a neural network in one of the hidden layer representations (b), and finally the visualization of the hidden representation at the end of training (c), in which the neural network is capable of discriminating between the half-moons linearly.

Auto means *self*, thus the meaning is to self-encode, which refers to having a model that can encode an input $\mathbf{x}$ to a latent variable $\mathbf{z}$ and decode this latent variable back into the input dimension, denoted a reconstruction $\hat{\mathbf{x}}$. In Figure 2.5a we show a simple visualization of a densely connected 1 hidden layer auto-encoder. In this context we denote the bottleneck, as a latent layer $\mathbf{z}$, and the remainder of the subsequent layers (not present in this visualization) hidden layers, $\mathbf{h}$. For ease of notation, we show a simpler graphical representation of the auto-encoder in Figure 2.5b, in which each edge corresponds to a stacking of hidden layers.

A breakthrough in deep auto-encoding was introduced in Hinton and Salakhutdinov (2006). They presented a model for dimensionality reduction that applied the pre-training and fine-tuning step, from Hinton et al. (2006), to avoid vanishing gradients when stacking layers. Within the pre-training step, they trained RBMs with the *contrastive divergence* update (Hinton, 2002), which can be viewed as a 2-layer bidirectional graphical model, for each hidden layer. This pre-training method was quite cumbersome, and scaled badly when stacking many layers on top of each other. Vincent et al. (2008) introduced a simpler process for the layer-wise pre-training, denoted the *denoising auto-encoder*, and later Vincent et al. (2010) complemented this with a stacked version. The introduction of these opened for new findings within areas such as semi-supervised learning (Ranzato and Szummer, 2008), natural image denoising (Vincent et al., 2010), topic modeling (Salakhutdinov and Hinton, 2009) and much more.

In broad terms, there exist two complementary tricks to make an efficient neural network auto-encoder. One is to learn to reconstruct the input data through a bottleneck of lower dimensionality. This *force* the model to embed the part of

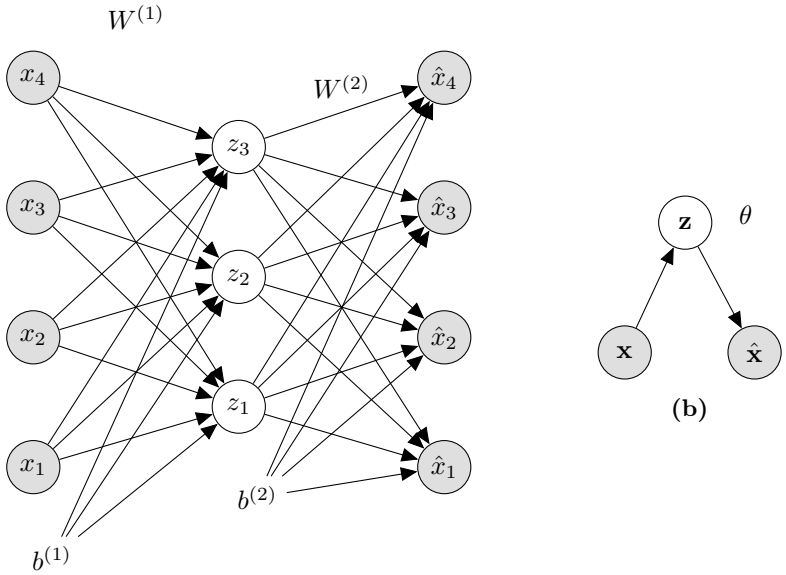


Figure 2.5: (a) A graphical representation of a densely connected auto-encoder with one hidden layer. (b) A simple graphical visualization of an auto-encoder, where the edges represent an undefined stacking of neural network layers.

the input data, from which it can reproduce it in the best way possible. The other trick, is to map from a noisy input $\tilde{\mathbf{x}}$ to the reconstruction $\hat{\mathbf{x}}$, but using the non-noisy $\mathbf{x}$ in the objective function, $\mathcal{F}(\hat{\mathbf{x}}_\theta(\tilde{\mathbf{x}}), \mathbf{x})$.

With the advent of activation functions, such as the ReLU, and non-trivial connectivity, e.g. residual connections (He et al., 2015), the auto-encoder is not prone to the vanishing gradient problem, thus there is no need for pre-training. This has let to much more complex and expressive auto-encoder architectures such as the *ladder network* for semi-supervised learning (Rasmus et al., 2015) and U-Nets for image segmentation (Ronneberger et al., 2015) that can be trained in an end-to-end fashion.

For an in-depth tutorial on auto-encoders, go to the lab exercises:

`github.com/DeepLearningDTU/02456-deep-learning`

Unsupervised representation learning: Lab5
An introduction to the concept of an auto-encoder with implementation details and a walk-through on how to validate these models. The lab also contains an exercise in which the task is to build an auto-encoder for semi-supervised learning.

For an implementation of the ladder network (Rasmus et al., 2015), go to the Parmesan library:

`github.com/casperkaae/parmesan`

Chapter 3

Deep Generative Models

In this chapter we will give an introduction to a general deterministic approximate inference approach called *variational inference* (Jordan et al., 1999). Next we will investigate variational inference in light of using an *inference model* (Hinton and Zemel, 1993; Salakhutdinov and Larochelle, 2010) and how this approach contains caveats when estimating the gradients. Finally, we will introduce the variational auto-encoder (VAE) (Kingma and Welling, 2014; Rezende et al., 2014) that can be viewed as a variational inference version of the deep neural network auto-encoder, introduced in the previous chapter.

3.1 Variational Inference (VI)

As explained in Section 1.1 we can derive Bayes theorem from the *likelihood* $p(\mathbf{x}|\mathbf{z})$, the *prior* $p(\mathbf{z})$, and the joint likelihood $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$:

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})} = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{\int_{\mathbf{z}} p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}} \ , \tag{3.1}$$

for which $p(\mathbf{z}|\mathbf{x})$ and $p(\mathbf{x})$ is denoted the *posterior* and the *evidence*. Solving the above integral is often intractable, hence a closed form solution can not be derived. A common approach in order to solve this problem is to apply Markov Chain Monte Carlo (MCMC) techniques, in which we draw samples from the posterior, $p(\mathbf{z}|\mathbf{x})$, without knowing the normalization constant. Increasing the number of samples drawn will ensure that the expectation converges towards the true posterior. We will not go into the different MCMC techniques here, but common features are, that they will eventually converge to the true $p(\mathbf{z}|\mathbf{x})$, but have a tendency to be cumbersome in regards to training and convergence time for a high-dimensional $\mathbf{z}$-space (latent space). In many applications the use of a too simple low dimensional latent space introduces a lack of expressiveness when modeling complex input data distributions. This can be related to the bottleneck of the neural network auto-encoder. If this is too low-dimensional, it is impossible to reconstruct the data sufficiently.

Variational inference (VI) is an approach in which we optimize the parameters, ϕ, of a *variational approximation*, $q_\phi(\mathbf{z})$, that approximates the posterior, so that $q_\phi(\mathbf{z}) \approx p(\mathbf{z}|\mathbf{x})$. In the following paragraphs, for the sake of simplicity, we will refer to the posterior and the variational approximation as discrete distributions, $P(\mathbf{z}|\mathbf{x})$ and $Q_\phi(\mathbf{z})$.

In order to explain VI in more details, we first introduce the notion of *entropy*. If we draw a sample from the discrete probability distribution, $P(\mathbf{z}|\mathbf{x})$, the entropy of this specific draw will be high as a function of how improbable it is. Conversely, if a specific draw from $P(\mathbf{z}|\mathbf{x})$ is the same every time, then the entropy is 0. From an information theoretical view, the entropy describes the average amount of information that is transfered from a *sender* to a *receiver* w.r.t. to $P(\mathbf{z}|\mathbf{x})$. The entropy of a random variable $\mathbf{z}$ can be seen as the *degree of surprise* for an event to happen (Bishop, 2006) and is defined as:

$$\mathcal{H}(P(\mathbf{z}|\mathbf{x})) = \mathbb{E}_{P(\mathbf{z}|\mathbf{x})}\left[-\log P(\mathbf{z}|\mathbf{x})\right] = -\sum_{\mathbf{z}} P(\mathbf{z}|\mathbf{x}) \log P(\mathbf{z}|\mathbf{x}) \ . \tag{3.2}$$

Now that we have a notion of the entropy, we are interested in a measure that describes a similarity between our variational approximation, $Q_\phi(\mathbf{z})$, and the true posterior $P(\mathbf{z}|\mathbf{x})$. The *Kullback-Leibler divergence* (or *relative entropy*),

does that by describing the amount of additional information needed if a sender transmits $\mathbf{z}$ to a receiver through $Q_\phi(\mathbf{z})$ rather than $P(\mathbf{z}|\mathbf{x})$. In other words it provides a measure to determine the difference between a distribution $Q_\phi(\mathbf{z})$ and $P(\mathbf{z}|\mathbf{x})$. It is defined as:

$$D_{KL}[P(\mathbf{z}|\mathbf{x})||Q_\phi(\mathbf{z})] = -\sum_{\mathbf{z}} P(\mathbf{z}|\mathbf{x}) \log \frac{Q_\phi(\mathbf{z})}{P(\mathbf{z}|\mathbf{x})} = \sum_{\mathbf{z}} P(\mathbf{z}|\mathbf{x}) \log \frac{P(\mathbf{z}|\mathbf{x})}{Q_\phi(\mathbf{z})} \ . \tag{3.3}$$

It is important to note that $D_{KL}(\cdot)$ is not symmetrical, $D_{KL}[P(\mathbf{z}|\mathbf{x})||Q_\phi(\mathbf{z})] \neq D_{KL}[Q_\phi(\mathbf{z})||\ P(\mathbf{z}|\mathbf{x})]$. It is also easily shown that the cross-entropy loss (introduced in Equation 2.11) derives from exactly $\mathcal{H}(P(\mathbf{z}|\mathbf{x}))$ and $D_{KL}[P(\mathbf{z}|\mathbf{x})||Q_\phi(\mathbf{z})]$:

$$\begin{aligned}
\mathcal{H}(P(\mathbf{z}|\mathbf{x}), Q_\phi(\mathbf{z})) &= \mathcal{H}(P(\mathbf{z}|\mathbf{x})) + D_{KL}[P(\mathbf{z}|\mathbf{x})||Q_\phi(\mathbf{z})] \\
&= -\sum_{\mathbf{z}} P(\mathbf{z}|\mathbf{x}) \log P(\mathbf{z}|\mathbf{x}) - \sum_{\mathbf{z}} P(\mathbf{z}|\mathbf{x}) \log \frac{Q_\phi(\mathbf{z})}{P(\mathbf{z}|\mathbf{x})} \\
&= -\sum_{\mathbf{z}} P(\mathbf{z}|\mathbf{x}) \left[\log \frac{P(\mathbf{z}|\mathbf{x}) Q_\phi(\mathbf{z})}{P(\mathbf{z}|\mathbf{x})} \right] \\
&= -\sum_{\mathbf{z}} P(\mathbf{z}|\mathbf{x}) \log Q_\phi(\mathbf{z}) \ .
\end{aligned} \tag{3.4}$$

$D_{KL}(\cdot)$ is easily extended to continuous distributions $p(\mathbf{z}|\mathbf{x})$ and $q_\phi(\mathbf{z})$:

$$D_{KL}[p(\mathbf{z}|\mathbf{x})||q_\phi(\mathbf{z})] = -\int_{\mathbf{z}} p(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})} dz \ . \tag{3.5}$$

Furthermore a property, is that $D_{KL}(\cdot) \geq 0$, which can be shown by utilizing *Jensen's inequality*[1]:

$$\begin{aligned}
D_{KL}[q_\phi(\mathbf{z})||p_\theta(\mathbf{z}|\mathbf{x})] &= -\int_z q_\phi(\mathbf{z}) \log \frac{p_\theta(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z})} dz \\
&\geq -\log \int_x q_\phi(\mathbf{z}) \frac{p_\theta(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z})} dx \\
&= -\log \int_x p_\theta(\mathbf{z}|\mathbf{x}) dx = -\log 1 = 0 \ .
\end{aligned} \tag{3.6}$$

In order to solve the problem from Equation 3.1, we introduce parameters θ for the joint distribution $p_\theta(\mathbf{x}, \mathbf{z})$. We seek to find an approximation to the intractable part of the theorem, $p(\mathbf{x}) = \int_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}) d\mathbf{z}$, such that:

$$\arg\max_\theta \prod_i^N p_\theta(x_i) \ . \tag{3.7}$$

[1] If $f(\cdot)$ is a convex function, Jensen's inequality states that $f(\mathbb{E}(x)) \leq \mathbb{E}(f(x))$.

The objective function for the optimization problem is normally derived directly from the evidence, however we intend to derive it slightly differently. As stated above, we introduce the variational approximation with parameters ϕ, and require that it must be similar to the posterior. In order to define this relationship, we can apply the KL-divergence to the two distributions. Hence, when the KL-divergence is small, we ensure that $q_\phi(\mathbf{z}) \approx p(\mathbf{z}|\mathbf{x})$. However, due to the asymmetry, we must define whether we deploy the *forward* KL-divergence $D_{KL}[p_\theta(\mathbf{z}|\mathbf{x})||q_\phi(\mathbf{z})]$ or the *reverse* KL-divergence $D_{KL}[q_\phi(\mathbf{z})||p_\theta(\mathbf{z}|\mathbf{x})]$. Murphy (2012) amongst others, notice that there is properties that may affect an optimization problem when defining one over the other. If $q_\phi(\mathbf{z}) = 0$ and $p_\theta(\mathbf{z}|\mathbf{x}) > 0$ the forward KL-divergence is infinite, referred to as *zero avoiding*. The result of this is that $q_\phi(\mathbf{z})$ has a tendency to overestimate the support of $p_\theta(\mathbf{z}|\mathbf{x})$. The reverse KL-divergence on the other hand is infinite for $p_\theta(\mathbf{z}|\mathbf{x}) = 0$ and $q_\phi(\mathbf{z}) > 0$ (*zero forcing*), which results in a tendency where $q_\phi(\mathbf{z})$ underestimates $p_\theta(\mathbf{z}|\mathbf{x})$. In order to exemplify the result of using one over the other, it is helpful to imagine $p_\theta(\mathbf{z}|\mathbf{x})$ as a bimodal distribution and $q_\phi(\mathbf{z})$ as a unimodal distribution. Optimizing w.r.t. the forward KL-divergence will result in the unimodal distribution covering the full support of the bimodal distribution. However, this comes at a cost, since there will be many regions covered by $q_\phi(\mathbf{z})$ with no density for $p_\theta(\mathbf{z}|\mathbf{x})$. Conversely, the reverse KL-divergence would fit the unimodal distribution to one of the modes of the bimodal distribution and would thereby be an underestimation of the true posterior. In order to avoid that the distribution $q_\phi(\mathbf{z})$ covers too many no-density regions we derive the objective of the optimization problem from the reverse KL-divergence:

$$D_{KL}[q_\phi(\mathbf{z})||p_\theta(\mathbf{z}|\mathbf{x})] = -\int_{\mathbf{z}} q_\phi(\mathbf{z}) \log \frac{p_\theta(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z})} d\mathbf{z} \ . \tag{3.8}$$

Another reason for starting with the reverse KL-divergence is that taking the expectation, $\int_{\mathbf{z}} q_\phi(\mathbf{z}) = \mathbb{E}_{q_\phi(\mathbf{z})}$ is tractable compared to the opposite. However, the above does not suffice, since it is only defined w.r.t. the posterior. Referring back to Bayes theorem, we need to optimize for the joint probability $p_\theta(\mathbf{x}, \mathbf{z})$. Introducing the evidence to the above can easily embed this:

$$\begin{aligned} D_{KL}[q_\phi(\mathbf{z})||p_\theta(\mathbf{z}|\mathbf{x})] &= -\int_{\mathbf{z}} q_\phi(\mathbf{z}) \log \frac{p_\theta(\mathbf{x})}{p_\theta(\mathbf{x})} \frac{p_\theta(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z})} d\mathbf{z} \\ &= -\int_{\mathbf{z}} q_\phi(\mathbf{z}) \log \frac{p_\theta(\mathbf{z}, \mathbf{x})}{q_\phi(\mathbf{z}) p_\theta(\mathbf{x})} d\mathbf{z} \\ &= -\int_{\mathbf{z}} q_\phi(\mathbf{z}) \log \frac{p_\theta(\mathbf{z}, \mathbf{x})}{q_\phi(\mathbf{z})} d\mathbf{z} + \log p_\theta(\mathbf{x}) \ . \end{aligned} \tag{3.9}$$

As explained earlier, it is intractable to optimize w.r.t. the evidence, so Equation 3.9 can be rearranged:

$$\log p_\theta(\mathbf{x}) = D_{KL}[q_\phi(\mathbf{z})||p(\mathbf{z}|\mathbf{x})] + \int_{\mathbf{z}} q_\phi(\mathbf{z}) \log \frac{p_\theta(\mathbf{z}, \mathbf{x})}{q_\phi(\mathbf{z})} d\mathbf{z} \ . \tag{3.10}$$

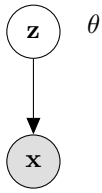
From the above we have achieved to isolate the evidence (normalization constant). However, we are still dependent on the reverse KL-divergence. Since we know that $D_{KL} \geq 0$, we can remove this term from Equation 3.10 resulting in a lower bound on the evidence, denoted *evidence lower bound* (ELBO) for which there is no dependency on the true posterior:

$$\begin{aligned}\log p_\theta(\mathbf{x}) &\geq \int_{\mathbf{z}} q_\phi(\mathbf{z}) \log \frac{p_\theta(\mathbf{x}|\mathbf{z}) p_\theta(\mathbf{z})}{q_\phi(\mathbf{z})} d\mathbf{z} \\ &= \mathbb{E}_{q_\phi(\mathbf{z})} \left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - D_{KL}[q_\phi(\mathbf{z})||p_\theta(\mathbf{z})] \equiv \mathcal{L}_{\phi,\theta}(\mathbf{x}) \ . \end{aligned} \tag{3.11}$$

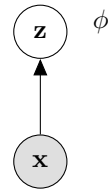
We now have an optimization problem and we can optimize the bound w.r.t. ϕ and θ where at the maximum value, the variational approximation equals the posterior, $q_\phi(\mathbf{z}) = p(\mathbf{z}|\mathbf{x})$, and the bound equals the evidence, $\mathcal{L}_{\phi,\theta}(\mathbf{x}) = \log p_\theta(\mathbf{x})$. The integral in the ELBO does not naturally introduce less complexity. However, since we have replaced the dependency on $p(\mathbf{z}|\mathbf{x})$ with one on $q_\phi(\mathbf{z})$, we now have the option to choose from a tractable family, for which we can achieve a good approximation to the posterior and one where we can optimize the parameters ϕ efficiently.

3.2 VI with Deep Neural Networks

For high-dimensional complex datasets, e.g. images our audio, it will not suffice to introduce $q_\phi(\mathbf{z})$ as a simple tractable distribution. We need to somehow approximate the posterior through an expressive model. This is denoted an inference model for which we have seen many different parameterizations through time (Hinton and Zemel, 1993; Dayan and Hinton, 1996; Salakhutdinov and Larochelle, 2010). In Figure 3.1 we present a graphical representation of a probabilistic model with a latent variable $\mathbf{z}$ and observed variable $\mathbf{x}$. The generative model is factorized as $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})$ and the corresponding inference model is $q_\phi(\mathbf{z}|\mathbf{x})$. Parameterizing the inference and generative models with neural networks provide a more expressive model, since the deep neural network can find a non-linear representation that lies on a manifold within a tractable distribution. However, as we shall see, using a deep neural network inference model raises problems during optimization, due to high variance of the gradient estimates.



(a) Generative model p_θ



(b) Inference model q_ϕ

Figure 3.1: Graphical representation of the generative model parameterized by θ (a) and the variational approximation/inference model parameterized by ϕ (b). $\mathbf{z}$ represents a latent variable that can either be expressed by a discrete or continuous distribution and $\mathbf{x}$ represents the observed variable.

3.2.1 A High-Variance Gradient Estimator

In order to formulate a gradient estimator for these models, we must first derive the ELBO from Equation 3.11 w.r.t. θ:

$$\begin{aligned}\nabla_\theta \mathcal{L}(\mathbf{x}) &= \nabla_\theta \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log p_\theta(\mathbf{x},\mathbf{z}) d\mathbf{z} - \nabla_\theta \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log q_\phi(\mathbf{z}|\mathbf{x}) d\mathbf{z} \\ &= \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \nabla_\theta \log p_\theta(\mathbf{x},\mathbf{z}) d\mathbf{z} \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\nabla_\theta \log p_\theta(\mathbf{x},\mathbf{z})\right] \ . \end{aligned} \tag{3.12}$$

The $\mathbf{z}$ for calculating the gradient w.r.t. θ is thereby dependent on a sample from the variational approximation. Since the ELBO is dependent on the expectation over the variational approximation, which is a function of ϕ, the gradients w.r.t. to ϕ is calculated in a slightly different way:

$$\nabla_\phi \mathcal{L}(\mathbf{x}) = \underbrace{\nabla_\phi \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log p_\theta(\mathbf{x},\mathbf{z}) d\mathbf{z}}_{\nabla_\phi \mathcal{L}^A(\mathbf{x})} - \underbrace{\nabla_\phi \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log q_\phi(\mathbf{z}|\mathbf{x}) d\mathbf{z}}_{\nabla_\phi \mathcal{L}^B(\mathbf{x})} \ . \tag{3.13}$$

We split the gradient w.r.t. ϕ into two terms for convenience, for which the first term is given by:

$$\begin{aligned}\nabla_\phi \mathcal{L}^A(\mathbf{x}) &= \nabla_\phi \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log p_\theta(\mathbf{x},\mathbf{z}) d\mathbf{z} \\ &= \int_{\mathbf{z}} \nabla_\phi q_\phi(\mathbf{z}|\mathbf{x}) \log p_\theta(\mathbf{x},\mathbf{z}) d\mathbf{z} \ . \end{aligned} \tag{3.14}$$

The derivation of the second term is a little more complex, since we must deploy the product rule for gradients[2], and the chain rule on the log-likelihood[3]:

$$\begin{aligned}
\nabla_\phi \mathcal{L}^B(\mathbf{x}) &= \nabla_\phi \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log q_\phi(\mathbf{z}|\mathbf{x}) d\mathbf{z} \\
&= \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \nabla_\phi \log q_\phi(\mathbf{z}|\mathbf{x}) d\mathbf{z} + \int_{\mathbf{z}} \nabla_\phi q_\phi(\mathbf{z}|\mathbf{x}) \log q_\phi(\mathbf{z}|\mathbf{x}) d\mathbf{z} \\
&= \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \left(\frac{\nabla_\phi q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} \right) d\mathbf{z} + \int_{\mathbf{z}} \nabla_\phi q_\phi(\mathbf{z}|\mathbf{x}) \log q_\phi(\mathbf{z}|\mathbf{x}) d\mathbf{z} \\
&= \nabla_\phi \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) d\mathbf{z} + \int_{\mathbf{z}} \nabla_\phi q_\phi(\mathbf{z}|\mathbf{x}) \log q_\phi(\mathbf{z}|\mathbf{x}) d\mathbf{z} \\
&= \nabla_\phi 1 + \int_{\mathbf{z}} \nabla_\phi q_\phi(\mathbf{z}|\mathbf{x}) \log q_\phi(\mathbf{z}|\mathbf{x}) d\mathbf{z} \\
&= \int_{\mathbf{z}} \nabla_\phi q_\phi(\mathbf{z}|\mathbf{x}) \log q_\phi(\mathbf{z}|\mathbf{x}) d\mathbf{z} \ . \qquad (3.15)
\end{aligned}$$

From knowing that $\nabla_\phi q_\phi(\mathbf{z}|\mathbf{x}) \log q_\phi(\mathbf{z}|\mathbf{x}) = q_\phi(\mathbf{z}|\mathbf{x}) \nabla_\phi \log q_\phi(\mathbf{z}|\mathbf{x})$ we can combine the two terms by:

$$\begin{aligned}
\nabla_\phi \mathcal{L}(\mathbf{x}) &= \nabla_\phi \mathcal{L}^A(\mathbf{x}) - \nabla_\phi \mathcal{L}^B(\mathbf{x}) \\
&= \int_{\mathbf{z}} \nabla_\phi q_\phi(\mathbf{z}|\mathbf{x}) \log p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} - \int_{\mathbf{z}} \nabla_\phi q_\phi(\mathbf{z}|\mathbf{x}) \log q_\phi(\mathbf{z}|\mathbf{x}) d\mathbf{z} \\
&= \int_{\mathbf{z}} \nabla_\phi q_\phi(\mathbf{z}|\mathbf{x}) \left[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}) \right] d\mathbf{z} \\
&= \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \nabla_\phi \log q_\phi(\mathbf{z}|\mathbf{x}) \left[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}) \right] d\mathbf{z} \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \nabla_\phi \log q_\phi(\mathbf{z}|\mathbf{x}) \right] \ . \qquad (3.16)
\end{aligned}$$

In order to calculate the gradients in Equation 3.12 and 3.16 we apply Monte Carlo integration over a number of samples s to estimate:

$$\nabla_\theta \mathcal{L}(\mathbf{x}) \approx \frac{1}{s} \sum_{i=1}^{s} \mathbb{E}_{q_\phi(\mathbf{z}^{(i)}|\mathbf{x})} \left[\nabla_\theta \log p_\theta(\mathbf{x}, \mathbf{z}^{(i)}) \right] \ , \qquad (3.17)$$

and

$$\nabla_\phi \mathcal{L}(\mathbf{x}) \approx \frac{1}{s} \sum_{i=1}^{s} \mathbb{E}_{q_\phi(\mathbf{z}^{(i)}|\mathbf{x})} \left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z}^{(i)})}{q_\phi(\mathbf{z}^{(i)}|\mathbf{x})} \nabla_\phi \log q_\phi(\mathbf{z}^{(i)}|\mathbf{x}) \right] \ . \qquad (3.18)$$

[2] $\nabla(f \cdot g) = \nabla f \cdot g + f \cdot \nabla g$
[3] $\nabla \log p(\mathbf{x}) = \nabla p(\mathbf{x}) / p(\mathbf{x})$

By utilizing the MCMC gradient estimators it is straightforward to apply a SGD algorithm to optimize the parameters ϕ and θ. However, due to the *scaling-term* of the gradients w.r.t. ϕ inside the expectation, $\mathbb{E}_{q_\phi(\mathbf{z}^{(i)}|\mathbf{x})}[\cdot]$, the estimator in Equation 3.18 has proven close to useless (Mnih and Gregor, 2014; Hinton and Zemel, 1993; Dayan and Hinton, 1996). To circumvent the issue of high-variance gradient estimates, Mnih and Gregor (2014) introduced a technique to reduce variance, denoted *neural variational inference and learning* (NVIL), by applying control variates. Another less general, but more popular, approach to estimate the gradients of a generative model and inference model is to apply the *reparameterization trick* (Kingma and Welling, 2014; Rezende et al., 2014).

3.2.2 Variational Auto-Encoder

Instead of assuming that $\mathbf{z}^{(i)} \approx q_\phi(\mathbf{z}^{(i)}|\mathbf{x})$, the reparameterization trick changes the variable $\mathbf{z}^{(i)}$ to come from a deterministic function:

$$\mathbf{z}^{(i)} = g_\phi(\epsilon^{(i)}, \mathbf{x}), \tag{3.19}$$

where $\epsilon^{(i)} \approx p(\epsilon^{(i)})$. The gradients w.r.t. ϕ are hereby not dependent on the expectation and this results in a much simpler formulation of the gradient estimator:

$$\nabla_{\phi,\theta}\mathcal{L}(\mathbf{x}) \approx \frac{1}{s}\sum_{i=1}^{s}\mathbb{E}_{p(\epsilon^{(i)})}\left[\nabla_{\phi,\theta}\log\frac{p_\theta(\mathbf{x}, g_\phi(\epsilon^{(i)}, \mathbf{x}))}{q_\phi(g_\phi(\epsilon^{(i)}, \mathbf{x})|\mathbf{x})}\right] . \tag{3.20}$$

With the above formulation, the gradient is not dependent on stochasticity in the generative and inference model, thus the entire model θ and ϕ is differential w.r.t. to the ELBO. This means that we can now backpropagate the ELBO through the generative model θ followed by the inference model ϕ, which is prone to much less variance, due to the elimination of the disjoint update. Furthermore, it is easily implemented in modern frameworks, e.g. Tensorflow (Abadi et al., 2015) that utilize automatic differentiation. A model that applies the reparameterization trick, while being parameterized by neural networks in both generative and inference models, is often referred to as a *variational auto-encoder* (VAE) (cf. Figure 3.2) (Kingma and Welling, 2014; Rezende et al., 2014), due to its obvious similarity with the neural network auto-encoder.

The reparameterization trick applies to a wide family of continuous latent variables, but the Gaussian is the most common, for which the trick is given by:

$$\mathbf{z} = g_\phi(\epsilon \approx \mathcal{N}(0, I), \mathbf{x}) = \mu_\phi(\mathbf{x}) + \sigma_\phi(\mathbf{x}) \odot \epsilon \ . \tag{3.21}$$

The generative model of the most commonly used VAE is thereby decomposed by:

$$p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z}) \ , \tag{3.22}$$

with

$$p_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{z}|0, I) \tag{3.23}$$

$$p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mu_\theta(\mathbf{z}), \sigma_\theta(\mathbf{z})) \quad \text{or} \quad P_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{B}(\mathbf{x}|\mu_\theta(\mathbf{z})) \ , \tag{3.24}$$

where $\mathcal{N}(\cdot)$ and $\mathcal{B}(\cdot)$ denote the Gaussian and Bernoulli distribution. The corresponding inference model is given by:

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mu_\phi(\mathbf{x}), \sigma_\phi(\mathbf{x})) \ . \tag{3.25}$$

$\mu_\phi(\mathbf{x})$, $\sigma_\phi(\mathbf{x})$, $\mu_\theta(\mathbf{z})$, $\sigma_\theta(\mathbf{z})$ are all parameterized by deep neural networks. The inference model is thereby defined as:

$$h^{(0)} = \mathbf{x}, \tag{3.26}$$

$$h^{(m)} = f(W^{(m)}h^{(m-1)} + b^{(m)}) \ , \quad m = 1, ..., M-1, \tag{3.27}$$

$$\mu_\phi = W^{(M)}_{\phi,\mu}h^{(M-1)} + b^{(M)}_{\phi,\mu}, \tag{3.28}$$

$$\sigma_\phi = \texttt{softplus}(W^{(M)}_{\phi,\sigma}h^{(M-1)} + b^{(M)}_{\phi,\sigma}) \ , \tag{3.29}$$

where *softplus* denotes an activation function with similar characteristics as the ReLU, while being differentiable in 0. After defining each element of the VAE it becomes obvious that the two terms in Equation 3.11 can be viewed as a reconstruction regularized by a term that keeps the variational approximation close to the uninformative prior:

$$\mathcal{L}_{\phi,\theta}(x) = \underbrace{\mathbb{E}_{q_\phi(\mathbf{z})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right]}_{\text{reconstruction term}} - \underbrace{D_{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})]}_{\text{regularization term}} \ . \tag{3.30}$$

When $D_{KL}[q_\phi(\mathbf{z})||p_\theta(\mathbf{z})] = 0$ there will be no information going from the *encoder*. This will be referred to as the latent variable $\mathbf{z}$ is *collapsing* and will be discussed further in Section 3.3.

From the above formulation of the typical VAE it is obvious that rather limiting assumptions are made on the true posterior distribution $p(\mathbf{z}|\mathbf{x})$. First of, it is assumed that the posterior distribution is factorial, so that all variables are statistically independent, $p(x_1, x_2) = p(x_1)p(x_2)$. Another assumption is that the neural networks, used in the inference and generative models, are *flexible* enough to map complex input distributions $p(\mathbf{x})$ into a rather simple $q_\phi(\mathbf{z}|\mathbf{x})$ that is only defined by a unimodal distribution.

Burda et al. (2015) was among the first to identify this problem. They proposed the *importance-weighted auto-encoder* (IWAE), which is a way to perform multiple samples in the variational approximation, $q_\phi(\mathbf{z}|\mathbf{x})$, so that the ELBO becomes *tighter* and the model performs a better approximation towards the true posterior. In their work an unbiased estimator of $p(\mathbf{x})$ is introduced that applies importance weights inside the log-term of Equation 3.30:

$$\mathcal{L}_{\phi,\theta}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{1}{k}\sum_{j=1}^{k} \frac{p_\theta(\mathbf{x}|\mathbf{z}^{(j)})p_\theta(\mathbf{z}^{(j)})}{q_\phi(\mathbf{z}^{(j)}|\mathbf{x})}\right] . \tag{3.31}$$

The intuition behind Equation 3.31 is that the normal VAE emphasizes high-probability regions of the approximate posterior too much, leaving little to low-probability regions. In the IWAE we loosen the restrictions such that the variational approximation may fit a true posterior that does not align with the limiting assumptions of the VAE. Note that the 1-sample IWAE is equivalent to the standard VAE. In this thesis we will refer to the ELBO of a VAE as formulated in Equation 3.31, where we omit the parameters ϕ and θ for short and $\mathcal{L}(\mathbf{x})$ denotes 1 IW sample and $\mathcal{L}^{10}(\mathbf{x})$ denotes 10 IW samples.

Despite the significant improvement from using the tighter bound in equation 3.31 compared to Equation 3.11 the single latent layer with a relatively simple distribution still pose a limiting constraint to approximate the true posterior. There are two main research paths that intend to solve this problem, by (i) adding more latent variables, and (ii) modeling a more flexible distribution for each latent variable. The research throughout this thesis mainly concerns (i), where it is important to note that most of the findings in (i) and (ii) are not mutually exclusive.

For an in-depth tutorial on variational auto-encoders, go to the lab exercises:

`github.com/DeepLearningDTU/02456-deep-learning`

Variational auto-encoders: Lab5
An introduction to the derivation and theoretical background of the variational auto-encoder in cohesion with a Tensorflow (Abadi et al., 2015) implementation on how to implement and train a one latent layer model.

For an implementation of the importance weighted auto-encoder (Burda et al., 2015), go to the Parmesan library:

`github.com/casperkaae/parmesan`

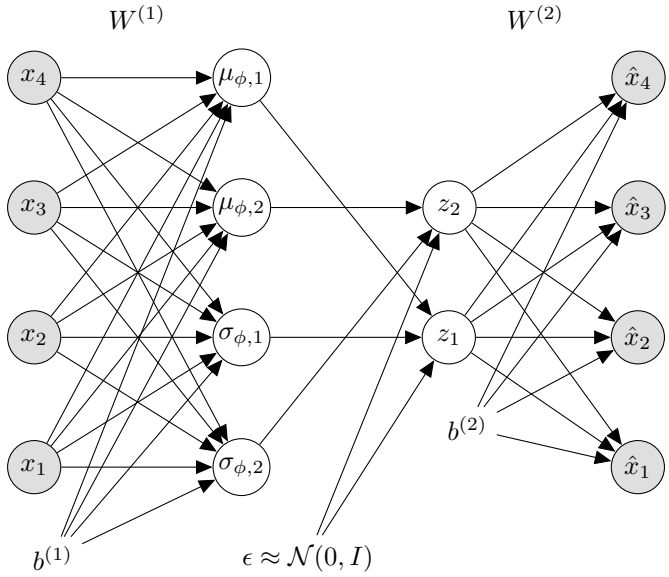


Figure 3.2: Visualization of a densely connected variational auto-encoder (VAE) with one latent layer $\mathbf{z}$ consisting of two units and no hidden deterministic layers. From this visualization it is evident that the VAE enables backpropagation through all variables, since the stochastic variable can be circumvented when applying the chain-rule.

3.3 Towards a Richer Posterior

This section cites two of the contributions in this thesis:

B Sønderby, C.K., Raiko, T., Maaløe, L., Sønderby, S. K., Winther, O.. Ladder variational autoencoders. In *Advances in Neural Information Processing Systems*, pages 3738-3746.

C Maaløe, L., Fraccaro, M., Winther, O. (2017). CaGeM: A cluster aware deep generative model. In *Neural Information Processing Systems Workshop* on Approximate Bayesian Inference.

The general assumption is to formulate the VAE with a diagonal covariance, which leaves limitations to the expressiveness of the model, since $q_\phi(\mathbf{z}|\mathbf{x})$ will be restricted in modeling more complex posteriors. Hence, if the data lies on a complex posterior, the variational approximation could be a *bad fit*. Adding more latent layers may improve flexibility, since the model is thereby able to learn different representations of the posterior throughout the latent variables.

(a) Generative model p_θ **(b)** Inference model q_ϕ

Figure 3.3: Graphical representation of the generative model parameterized by θ (a) and the variational approximation/inference model parameterized by ϕ (b) for a two latent layer hierarchical VAE. $\mathbf{z_1}$ and $\mathbf{z_2}$ represent the latent variables expressed by continuous distributions and $\mathbf{x}$ represents the observed variable.

This leads to the more general formulation of the VAE with L latent layers (cf. Figure 3.3):

$$p_\theta(\mathbf{x}, \mathbf{Z}) = p_\theta(\mathbf{x}|\mathbf{z}_1)p_\theta(\mathbf{z}_L) \prod_{i=1}^{L-1} p_\theta(\mathbf{z}_i|\mathbf{z}_{i+1}) \ , \tag{3.32}$$

where $\mathbf{Z} = \mathbf{z}_i, ..., \mathbf{z}_L$ and:

$$p_\theta(\mathbf{z}_i|\mathbf{z}_{i+1}) = \mathcal{N}(\mathbf{z}_i|\mu_{\theta,i}(\mathbf{z}_{i+1}), \sigma_{\theta,i}(\mathbf{z}_{i+1})), \quad p_\theta(\mathbf{z}_L) = \mathcal{N}(\mathbf{z}_L|0, I) \tag{3.33}$$

$$p_\theta(\mathbf{x}|\mathbf{z}_1) = \mathcal{N}(\mathbf{x}|\mu_{\theta,0}(\mathbf{z}_1), \sigma_{\theta,0}(\mathbf{z}_1)) \quad \text{or} \quad P_\theta(\mathbf{x}|\mathbf{z}_1) = \mathcal{B}(\mathbf{x}|\mu_{\theta,0}(\mathbf{z}_1)) \ . \tag{3.34}$$

The corresponding *vanilla* inference model is decomposed by:

$$q_\phi(\mathbf{Z}|\mathbf{x}) = q_\phi(\mathbf{z}_1|\mathbf{x}) \prod_{i=2}^{L} q_\phi(\mathbf{z}_i|\mathbf{z}_{i-1}) \ , \tag{3.35}$$

with

$$q_\phi(\mathbf{z}_1|\mathbf{x}) = \mathcal{N}(\mathbf{z}_1|\mu_{\phi,0}(\mathbf{x}), \sigma_{\phi,0}(\mathbf{x})), \quad q_\phi(\mathbf{z}_i|\mathbf{z}_{i-1}) = \mathcal{N}(\mathbf{z}_i|\mu_{\phi,i}(\mathbf{z}_{i-1}), \sigma_{\phi,i}(\mathbf{z}_{i-1})) \ . \tag{3.36}$$

The reason why a model with more latent layers may introduce more flexibility is simply that it has more expressive power through the ability to learn multiple representations of the approximation to the posterior. The view on representation learning introduced in Section 2.2 is especially useful here. Thus, in many

settings of hierarchical VAEs we seek to introduce dimensionality reduction between each latent layer $\mathbf{z}_1, ..., \mathbf{z}_L$, such that we force the network to learn more and more global information of the data distribution $p(\mathbf{x})$. Another intriguing property of the above formulation is that the latent layers $\mathbf{z}_1, ..., \mathbf{z}_{L-1}$ are more flexible, since the prior is learned as opposed to that of $p_\theta(\mathbf{z}_L) = \mathcal{N}(\mathbf{z}_L|0, I)$.

Finally, the hierarchical VAE introduces an intuitively compelling feature during inference (training), which bears resemblance to findings in the purely deterministic neural networks. Throughout the recent research within deep neural networks, we have seen many research contributions that successfully apply very deep architectures without the network being prone to vanishing gradients. These contributions mainly concern *skip*-connectivity, such as *residual connections* (He et al., 2015), *highway networks* (Srivastava et al., 2015), and *densely connected networks* (Huang et al., 2017). These inventions have been successful in very deep auto-encoder architectures, such as the PixelCNN++ (Salimans et al., 2017) and U-Net (Ronneberger et al., 2015). The intention of the skip-connectivity is to keep the path from the input to the output of the neural network as short as possible, while still having a deep stacking of layers that can learn highly non-linear representations. The skip-connections ensure that the inference-signal can skip the subsequent layers. During inference of the VAE, each latent variable can be viewed as a skip-connection between the inference model and the generative model. This is due to the fact that the sample $\mathbf{z}_1 \approx q_\phi(\mathbf{z}_1|\mathbf{x})$ serves as input to $p_\theta(\mathbf{x}|\mathbf{z}_1)$ and $\mathbf{z}_2 \approx q_\phi(\mathbf{z}_2|\mathbf{z}_1)$ serves as input to $p_\theta(\mathbf{z}_1|\mathbf{z}_2)$ and so forth. For a very deep hierarchical inference and generative model we can thereby utilize the *natural skip-connectivity* in order to learn very deep architectures.

However, as presented in Sønderby et al. (2016) (see **Appendix B**) among others, the latent layers have a tendency to collapse when $L > 2$. Maaløe et al. (2017) (see **Appendix C**) gives an explanation to this problem in which we first consider the asymptotic average properties by taking the expectation over the true evidence of the data distribution $p_d(\mathbf{x})$. This is applied to the log-likelihood of model $p_\theta(\mathbf{x})$ (cf. Equation 3.2, 3.3, 3.4):

$$\begin{aligned}
\mathbb{E}_{p_d(\mathbf{x})}[\log p_\theta(\mathbf{x})] &= \int_{\mathbf{x}} p_d(\mathbf{x}) \log p_\theta(\mathbf{x}) d\mathbf{x} \\
&= \int_{\mathbf{x}} p_d(\mathbf{x}) \log p_d(\mathbf{x}) \frac{p_\theta(\mathbf{x})}{p_d(\mathbf{x})} d\mathbf{x} \\
&= \int_{\mathbf{x}} p_d(\mathbf{x}) \log p_d(\mathbf{x}) + \int_{\mathbf{x}} p_d(\mathbf{x}) \log \frac{p_\theta(\mathbf{x})}{p_d(\mathbf{x})} d\mathbf{x} \\
&= -\mathcal{H}\left(p_d(\mathbf{x})\right) - D_{KL}\left[p_d(\mathbf{x})||p_\theta(\mathbf{x})\right] \ . \qquad (3.37)
\end{aligned}$$

Thus, the expected log-likelihood is the difference between the negative entropy of the data generating distribution $p_{data}(\mathbf{x})$, and the deviation between the

data generating distribution $p_d(\mathbf{x})$ and the model distribution $p_\theta(\mathbf{x})$. When we introduce a latent variable to the model, such that $p_\theta(\mathbf{x}) = \int_{\mathbf{z}} p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})d\mathbf{z}$ we can derive the ELBO by:

$$\begin{aligned}
\log p_\theta(\mathbf{x}) &= \log \int_{\mathbf{z}} p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} \\
&= \log \int_{\mathbf{z}} \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} \\
&\geq \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} d\mathbf{z} \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right] . \qquad (3.38)
\end{aligned}$$

We will now take the expectation over the true evidence of the data distribution on our new model:

$$\begin{aligned}
\mathbb{E}_{p_d(\mathbf{x})}[\log p_\theta(\mathbf{x})] &\geq \mathbb{E}_{p_d(\mathbf{x})} \left[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right] \right] \\
&= \int_{\mathbf{x}} p_d(\mathbf{x}) \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} d\mathbf{z} d\mathbf{x} \\
&= \int_{\mathbf{x}} p_d(\mathbf{x}) \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \left[\log p_d(\mathbf{x}) \frac{p_\theta(\mathbf{x})}{p_d(\mathbf{x})} - \log \frac{p_\theta(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} \right] d\mathbf{z} d\mathbf{x} \\
&= \int_{\mathbf{x}} p_d(\mathbf{x}) \log p_d(\mathbf{x}) \frac{p_\theta(\mathbf{x})}{p_d(\mathbf{x})} d\mathbf{x} - \int_{\mathbf{x}} p_d(\mathbf{x}) \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} d\mathbf{z} d\mathbf{x} \\
&= -\mathcal{H}\left(p_d(\mathbf{x})\right) - D_{KL}\left[p_d(\mathbf{x}) || p_\theta(\mathbf{x})\right] - \mathbb{E}_{p_d(\mathbf{x})}\left[D_{KL}\left[q_\phi(\mathbf{z}|\mathbf{x}) || p_\theta(\mathbf{z}|\mathbf{x})\right]\right] . \qquad (3.39)
\end{aligned}$$

When comparing Equation 3.37 with Equation 3.39 it is obvious that adding a latent variable $\mathbf{z}$ will add additional cost. Hence, the model has an incentive to omit the latent variable during optimization, which may eventually result in a collapse. Conversely, if the introduced latent variable proofs beneficial in maximizing the log-likelihood beyond the additional cost it will not collapse. It is trivial to show that the cost of adding more variables is even higher. Following the same derivation as in 3.39 for a model, $p_\theta(\mathbf{x}) = \int_{\mathbf{z}_1, \mathbf{z}_2} p_\theta(\mathbf{x}|\mathbf{z}_1)p_\theta(\mathbf{z}_1|\mathbf{z}_2)p_\theta(\mathbf{z}_2)d\mathbf{z}_1 d\mathbf{z}_2$ we get:

$$\begin{aligned}
\mathbb{E}_{p_d(\mathbf{x})}[\log p_\theta(\mathbf{x})] \geq -\mathcal{H}\left(p_d(\mathbf{x})\right) &- D_{KL}\left[p_d(\mathbf{x}) || p_\theta(\mathbf{x})\right] \\
&- \mathbb{E}_{p_d(\mathbf{x})}\left[D_{KL}\left[q_\phi(\mathbf{z}_1|\mathbf{x}) || p_\theta(\mathbf{z}_1|\mathbf{x})\right]\right] \\
&- \mathbb{E}_{p_d(\mathbf{x})}\left[D_{KL}\left[q_\phi(\mathbf{z}_2|\mathbf{z}_1) || p_\theta(\mathbf{z}_2|\mathbf{z}_1)\right]\right] . \qquad (3.40)
\end{aligned}$$

Since we initialize the variational approximation with deep neural networks, the samples from the latent variables will be very noisy in the beginning of a

SGD optimization. Therefore, the model will have a larger incentive to collapse latent variables in the beginning of training, resulting in a poor local maxima. As stated in Bowman et al. (2016); Fraccaro et al. (2016); Chen et al. (2017) the collapse of latent variable can also happen when the decoder neural network, parameterized by θ, is very powerful. In the following chapters we will elaborate more on our findings concerning how to avoid latent variables to collapse in deep generative models.

Concluding remarks There still remains a lot of recent research, that is not included in this thesis, investigating a more expressive VAE framework, such as *inverse autoregressive flows* (IAF) (Kingma et al., 2016), variational inference with *normalizing flows* (Rezende and Mohamed, 2015), and variational Gaussian processes (Tran et al., 2016). There are also variants of using the reparameterization in a supervised learning paradigm by applying priors to the weights of a neural network (Blundell et al., 2015).

Chapter 4

Deep Generative Models for Semi-supervised Learning

In the previous chapter we introduced a framework that utilizes deep neural networks for probabilistic generative modeling. As we briefly introduced in Section 1.2, such a modeling framework can prove useful for semi-supervised learning. In this chapter we first give an introduction to the methodological foundation of our work on semi-supervised learning. Throughout the chapter we make a clear distinction between (i) semi-supervised classification and (ii) semi-supervised generation. (i) relates to models with the objective of learning from labeled as well as unlabeled data in order to improve on a classification task. (ii) relates to models that are utilizing labeled and unlabeled data in order to improve on a generation task. We introduce the *auxiliary deep generative model* (Maaløe et al., 2016) (cf. **Appendix A**) that improves significantly on semi-supervised classification by utilizing a rich variational approximation. Then we introduce the *cluster-aware generative model* (Maaløe et al., 2017) (cf. **Appendix C**) that achieves a good generative performance by utilizing labeled information in both the variational approximation and generative model. In order to make a proper comparison, the models are benchmarked on image datasets that are widely used across the machine learning community. The datasets are MNIST (LeCun et al., 1998) (handwritten digits), OMNIGLOT (Lake et al., 2013) (handwritten characters), SVHN (Netzer et al., 2011) (natural images of house numbers), and NORB (LeCun et al., 2004) (images of toys).

4.1 Defining a Semi-Supervised VAE

This section cites one of the contributions in this thesis:

A Maaløe, L., Sønderby, C. K., Sønderby, S. K., Winther, O. (2016). Auxiliary deep generative models. In *Proceedings of the International Conference on Machine Learning*, pages 1445–1454.

Kingma et al. (2014) provide a rather general probabilistic model for semi-supervised classification, denoted M2, by introducing a new latent variable $\mathbf{y}$ to the framework of the single latent layer VAE. Conversely, to the hierarchical VAE introduced in Section 3.3, the two latent variables $\mathbf{z}$ and $\mathbf{y}$ are assumed statistically independent, such that $p(\mathbf{z}, \mathbf{y}) = p(\mathbf{z})p(\mathbf{y})$. By applying Bayes theorem we get following expression for the posterior:

$$p(\mathbf{z}, \mathbf{y}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z}, \mathbf{y})p(\mathbf{z})p(\mathbf{y})}{\int_{\mathbf{z},\mathbf{y}} p(\mathbf{x}|\mathbf{z}, \mathbf{y})p(\mathbf{z})p(\mathbf{y})d\mathbf{z}d\mathbf{y}} \ . \tag{4.1}$$

The reason for assuming that $\mathbf{z}$ and $\mathbf{y}$ are independent is that the model can be explicitly forced to learn different representations of the data distribution, $p(\mathbf{x})$, beneficial in solving semi-supervised classification. The generative model is defined as:

$$p_\theta(\mathbf{x}, \mathbf{z}, \mathbf{y}) = p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{y})p_\theta(\mathbf{z})p_\theta(\mathbf{y}) \ , \tag{4.2}$$

and the adhering distributions are:

$$p_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{z}|0, I) \ , \quad p_\theta(\mathbf{y}) = \mathrm{Cat}(\mathbf{y}|\pi) \ , \tag{4.3}$$

$$p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{y}) = \mathcal{N}(\mathbf{x}|\mu_\theta(\mathbf{z}, \mathbf{y}), \sigma_\theta(\mathbf{z}, \mathbf{y})) \quad \text{or} \quad P_\theta(\mathbf{x}|\mathbf{z}, \mathbf{y}) = \mathcal{B}(\mathbf{x}|\mu_\theta(\mathbf{z}, \mathbf{y})) \ , \tag{4.4}$$

where $\mathrm{Cat}(\cdot)$ is a multinomial distribution and π is a probability vector. We have hereby specified a different purpose of the two latent variables, where $\mathbf{z}$ remains a Gaussian. By defining $\mathbf{y}$ as a multinomial, the property is that it is perfectly aligned with a classifier, since it represents 1-of-K possible categories. In Figure 4.1a we present the graphical representation of the generative model.

From a learned generative model (Figure 4.1a) we can sample from $\mathbf{z}$ and generate an $\mathbf{x}$ corresponding to the 1-of-K classes that we choose from $\mathbf{y}$. The big difference from the VAE (cf. Section 3.2) is that we no longer learn all latent information in $\mathbf{z}$, since the class-dependent information is not embedded within. In Figure 4.2 we present an example of this generative process from the model introduced in Maaløe et al. (2016) (cf. **Appendix A**) that will be further elaborated in Section 4.2. In the Figure we see generated samples from a model learned on the MNIST dataset, which is a dataset comprising $70,000$

(a) Generative model p_θ **(b)** Inference model q_ϕ

Figure 4.1: Graphical representation of the M2 model from Kingma et al. (2014) with the generative model parameterized by θ (a) and the variational approximation/inference model parameterized by ϕ (b) for a two latent variable VAE for semi-supervised classification. $\mathbf{z}$ represents the continuous latent variable, $\mathbf{y}$ a partially observed latent variable, and $\mathbf{x}$ represents the observed variable.

28×28 gray-scale images of handwritten digits in the range from 0 through 9 (LeCun et al., 1998). Each row represents a specific sample from the posterior and each column represents 1-of-K categories. Since the latent variables are defined as independent in the model, the model embeds non-category specific global information in $\mathbf{z}$, which for this dataset is the *style* of the handwriting. Some of the rows are expressed by bold and vertical handwriting; others are expressed by thin and cursive.

Similarly to the VAE we introduce a variational approximation, $q_\phi(\mathbf{z}, \mathbf{y}|\mathbf{x})$, with parameters ϕ to learn an approximation to the true posterior. We decompose the variational approximation such that:

$$q_\phi(\mathbf{z}, \mathbf{y}|\mathbf{x}) = q_\phi(\mathbf{z}|\mathbf{y}, \mathbf{x}) q_\phi(\mathbf{y}|\mathbf{x}) \ , \tag{4.5}$$

where

$$q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y}) = \mathcal{N}(\mathbf{z}|\mu_\phi(\mathbf{x}, \mathbf{y}), \sigma_\phi(\mathbf{x}, \mathbf{y})) \ , \quad q_\phi(\mathbf{y}|\mathbf{x}) = \mathrm{Cat}(\mathbf{y}|\pi_\phi(\mathbf{x})) \ . \tag{4.6}$$

$q_\phi(\mathbf{y}|\mathbf{x})$ is defined as a discriminative classifier similarly to a neural network with a softmax output function (cf. Equation 2.7). Intuitively, one could suggest that the inference model should be defined similarly to the generative model by treating the latent variables as independent. However, this would pose a problem during inference, since we are interested in introducing a *weighting* term that weights the representation of $\mathbf{z}$ depending on the category. This leads to the inference model in Figure 4.1b. To explain this mathematically we tend to the derivation of the ELBO, which is derived in a similar fashion as in the

Figure 4.2: An example of generated samples from a generative model described in Maaløe et al. (2016) (cf. **Appendix A**) learned on the MNIST dataset (LeCun et al., 1998). Each row represents a different sample from the continuous distribution of the latent variable $\mathbf{z}$ and each column represents a different 1-of-K where K is 10; the number of categories for the handwritten digits 0 through 9.

previous chapter. Note that $\mathbf{y}$ is discrete:

$$\begin{aligned}\log p_\theta(\mathbf{x}) &= \log \sum_{\mathbf{y}} \int_{\mathbf{z}} p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z}) p_\theta(\mathbf{y}) p_\theta(\mathbf{z}) d\mathbf{z} \\ &= \log \sum_{\mathbf{y}} q_\phi(\mathbf{y}|\mathbf{x}) \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y}) \frac{p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z}) p_\theta(\mathbf{y}) p_\theta(\mathbf{z})}{q_\phi(\mathbf{y}|\mathbf{x}) q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})} d\mathbf{z} \\ &\geq \sum_{\mathbf{y}} q_\phi(\mathbf{y}|\mathbf{x}) \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y}) \log \frac{p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z}) p_\theta(\mathbf{y}) p_\theta(\mathbf{z})}{q_\phi(\mathbf{y}|\mathbf{x}) q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})} d\mathbf{z} \,. \end{aligned} \tag{4.7}$$

By rearranging the ELBO similarly to Equation 3.30 we can write up the bound for an observed $\mathbf{x}$ and unobserved latent variables $\mathbf{z}$ and $\mathbf{y}$, also denoted the unsupervised ELBO:

$$\begin{aligned}\mathcal{U}_{\phi,\theta}(\mathbf{x}) = \underbrace{\mathcal{H}\left[q_\phi(\mathbf{y}|\mathbf{x})\right]}_{\text{entropy term}} + \sum_{\mathbf{y}} \underbrace{q_\phi(\mathbf{y}|\mathbf{x})}_{\text{weighting term}} \Big[&\underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})}[\log p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z})]}_{\text{reconstruction term}} \\ &- \underbrace{D_{KL}[q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})||p_\theta(\mathbf{z})]}_{\text{regularization term}} \Big] \,. \end{aligned} \tag{4.8}$$

We are seeking to find the parameter setting for ϕ and θ that maximizes $\mathcal{U}_{\phi,\theta}(\mathbf{x})$, meaning that (i) the reconstruction error will decrease, (ii) that the deviation

between $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})$ and $p_\theta(\mathbf{z})$ is not too big, and (iii) that the entropy is maximized. It may seem counterintuitive that it is beneficial to increase the entropy for semi-supervised classification. However, this has a positive effect on precisely that, since it acts as a regularization term by pushing the multinomial towards being uniform. When introducing the labeled information during training, this ensures that the model will not *overfit* towards the small fraction of labeled data.

The unsupervised ELBO, $\mathcal{U}_{\phi,\theta}(\mathbf{x})$, provides a way to introduce the discrete latent variable $\mathbf{y}$, however, for semi-supervised classification it does not utilize the information from the fraction of labeled data. Therefore we introduce another ELBO for the labeled data for which the variable $\mathbf{y}$ is observed:

$$\log p_\theta(\mathbf{x},\mathbf{y}) \geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})}\left[\log p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z})\right] - D_{KL}\left[q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})||p_\theta(\mathbf{z})\right] \ . \tag{4.9}$$

In order to incorporate an error term for the classifier, $q_\phi(\mathbf{y}|\mathbf{x})$, we add the cross-entropy loss (cf. Equation 2.11) to the above:

$$\begin{aligned}\mathcal{J}_{\phi,\theta}(\mathbf{x},\mathbf{y}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})}\left[\log p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z})\right] - D_{KL}\left[q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})||p_\theta(\mathbf{z})\right] \\ - \alpha \cdot \underbrace{\mathcal{H}(\mathbf{y}, q_\phi(\mathbf{y}|\mathbf{x}))}_{\text{cross-entropy}} \ ,\end{aligned} \tag{4.10}$$

where α denotes a scaling hyper-parameter. The model for semi-supervised learning is optimized by SGD and the unsupervised and labeled bounds are estimated with mini-batches for the labeled data $\mathbf{x}_l, \mathbf{y}_l$ and the unlabeled data $\mathbf{x}_u$:

$$\mathcal{L}_{\phi,\theta}(\mathbf{x}_l,\mathbf{y}_l,\mathbf{x}_u) = \mathcal{J}_{\phi,\theta}(\mathbf{x}_l,\mathbf{y}_l) + \mathcal{U}_{\phi,\theta}(\mathbf{x}_u) \ . \tag{4.11}$$

We can deploy the reparameterization trick to calculate the gradients $\nabla_{\phi,\theta}$ similarly to Equation 3.12 and Equation 3.20.

From Kingma et al. (2014) the M2 model did not prove to perform very well[1], which is why they introduced the M1+M2 model, where M1 is a regular VAE. First the M1 model is trained to learn a good approximation to the posterior. Next the M2 model is trained by using samples from the latent layer of M1. Note that the M1 model parameters are fixed when the M2 model is trained. This combination of models proved a significant improvement[2] over the M2 model for semi-supervised classification.

[1] From experiments we have improved on the 11.97%(±1.71) for 100 labeled samples from MNIST reported in Kingma et al. (2014) to ≈ 9.5% (Maaløe et al., 2016).

[2] For the semi-supervised MNIST experiments, the M1+M2 model achieves a classification error of 3.33%(±0.14) compared to 11.97%(±1.71) for the M2 model.

Concluding remarks When comparing the performance gains from using M1+M2 over M2, it is clear that learning a latent variable that holds both the *style* and class information is beneficial towards semi-supervised classification. However, learning the M1 model and then the M2 model has some serious drawbacks, since the parameter space for the latent variable in M1 is not adaptable towards the semi-supervised classification accuracy. Therefore an intriguing adaption to the formulation in Kingma et al. (2014) would be to learn the M1+M2 model in cohesion. We have performed multiple attempts in learning this model, but the results never compared to the reported M1+M2 model and the topmost latent variable had a tendency to collapse (cf. Section 3.3). This was the initial key driver to our studies on the auxiliary deep generative model (Maaløe et al., 2016) (cf. **Appendix A**).

For an in-depth tutorial on variational auto-encoders for semi-supervised classification, go to the lab exercises:

`github.com/DeepLearningDTU/variational-autoencoders-summerschool-2016`

VAEs for semi-supervised learning: Lab4
A walk-through of the implementation details of a variational auto-encoder for semi-supervised learning.

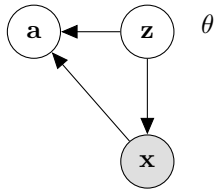
4.2 Auxiliary Deep Generative Models

This section cites one of the contributions in this thesis:

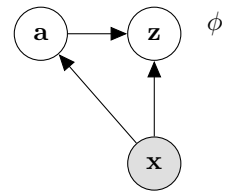
A Maaløe, L., Sønderby, C. K., Sønderby, S. K., Winther, O. (2016). Auxiliary deep generative models. In *Proceedings of the International Conference on Machine Learning*, pages 1445–1454.

As the name implies, an auxiliary variable is used to assist the task at hand and is widely used in the EM algorithm and Gibbs sampling. It was also previously introduced by Agakov and Barber (2004) in regards to variational inference. In Maaløe et al. (2015b; 2016) (cf. **Appendix A**) we introduce auxiliary variables in the VAE, denoted auxiliary variational auto-encoders (AVAE)[3], and the M2 model, denoted auxiliary deep generative models (ADGM), in order to (i) improve the flexibility of the variational approximation, and (ii) improve on semi-supervised classification. For the remainder of this section we will mainly consider the auxiliary variables for semi-supervised classification, ADGM, and in Chapter 5 we will analyze its contribution to the generative process.

[3]Concurrent with this work, Ranganath et al. (2016) developed a similar approach.



(a) Generative model p_θ



(b) Inference model q_ϕ

Figure 4.3: Graphical representation of the auxiliary VAE from Maaløe et al. (2016) (cf. **Appendix A**) with the generative model parameterized by θ (a) and the variational approximation/inference model parameterized by ϕ (b). $\mathbf{z}$ represents the continuous latent variable, $\mathbf{a}$ the continuous auxiliary variable, and $\mathbf{x}$ the observed variable.

A property of the auxiliary variable, $\mathbf{a}$, is that it complements the distribution $q_\phi(\mathbf{z}|\mathbf{x})$, so that it can fit a more complicated posterior, while maintaining the generative model $p_\theta(\mathbf{x}, \mathbf{z})$ when marginalizing over $\mathbf{a}$ (cf. Figure 4.3a):

$$p_\theta(\mathbf{x}, \mathbf{z}, \mathbf{a}) = p_\theta(\mathbf{a}|\mathbf{x}, \mathbf{z}) p_\theta(\mathbf{x}, \mathbf{z}) \ . \tag{4.12}$$

Similarly to the latent variable $\mathbf{z}$ the auxiliary variable $\mathbf{a}$ is defined by:

$$p_\theta(\mathbf{a}|\mathbf{z}, \mathbf{x}) = \mathcal{N}(\mathbf{a}|\mu_{a,\theta}(\mathbf{z}, \mathbf{x}), \sigma_{a,\theta}(\mathbf{z}, \mathbf{x})) \ . \tag{4.13}$$

The complementary inference model is:

$$q_\phi(\mathbf{a}, \mathbf{z}|\mathbf{x}) = q_\phi(\mathbf{a}|\mathbf{x}) q_\phi(\mathbf{z}|\mathbf{a}, \mathbf{x}) \ , \tag{4.14}$$

with

$$q_\phi(\mathbf{a}|\mathbf{x}) = \mathcal{N}(\mathbf{a}|\mu_{a,\phi}(\mathbf{x}), \sigma_{a,\phi}(\mathbf{x})) \ , \quad q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{a}) \quad = \mathcal{N}(\mathbf{z}|\mu_{z,\phi}(\mathbf{x}, \mathbf{a}), \sigma_{z,\phi}(\mathbf{x}, \mathbf{a})) \ . \tag{4.15}$$

If $p_\theta(\mathbf{a}|\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{a})$, the auxiliary variable will not be used in the variational approximation, thus the AVAE model would collapse into a normal VAE. In order to show this, we consider a model where we omit $\mathbf{x}$ so that $p_\theta(\mathbf{a}, \mathbf{z}) = p_\theta(\mathbf{a}|\mathbf{z}) p_\theta(\mathbf{z})$. In this case we consider the lower bound on the log-partition function $p_\theta(\mathbf{z}) = f_\theta(\mathbf{z})/Z$ and introduce the variational approximation $q_\phi(\mathbf{a}, \mathbf{z}) = q_\phi(\mathbf{z}|\mathbf{a}) q_\phi(\mathbf{a})$:

$$\begin{aligned} \log Z &= \log \int_{\mathbf{a}} \int_{\mathbf{z}} f_\theta(\mathbf{z}) p_\theta(\mathbf{a}|\mathbf{z}) d\mathbf{z} d\mathbf{a} \\ &= \log \int_{\mathbf{a}} q_\phi(\mathbf{a}) \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{a}) \frac{f_\theta(\mathbf{z}) p_\theta(\mathbf{a}|\mathbf{z})}{q_\phi(\mathbf{a}) q_\phi(\mathbf{z}|\mathbf{a})} d\mathbf{z} d\mathbf{a} \\ &\geq \int_{\mathbf{a}} q_\phi(\mathbf{a}) \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{a}) \log \frac{f_\theta(\mathbf{z}) p_\theta(\mathbf{a}|\mathbf{z})}{q_\phi(\mathbf{a}) q_\phi(\mathbf{z}|\mathbf{a})} d\mathbf{z} d\mathbf{a} \ . \end{aligned} \tag{4.16}$$

If the two latent variables are assumed independent in the generative model, $p_\theta(\mathbf{z}, \mathbf{a}) = p_\theta(\mathbf{z})p_\theta(\mathbf{a})$, the optimization falls back to the normal VAE:

$$\log Z \geq \int_{\mathbf{a}} q_\phi(\mathbf{a}) \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{a}) \log \frac{f_\theta(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{a})} d\mathbf{z} d\mathbf{a} \ . \tag{4.17}$$

This concludes that $p_\theta(\mathbf{a}|\mathbf{x}, \mathbf{z}) \neq p_\theta(\mathbf{a})$ must hold for the AVAE. One way to ensure this is to use a temperature on the $D_{KL}[q_\phi(\mathbf{a}|\mathbf{x})||p_\theta(\mathbf{a}|\mathbf{z}, \mathbf{x})]$ (cf. Chapter 5). To show that the additional auxiliary variable enables a more expressive variational approximation, we consider an experiment on a 2D potential model $p(\mathbf{z}) = f_\theta(\mathbf{z})/Z = e^{U(\mathbf{z})}/Z$. Similarly to Rezende and Mohamed (2015) we define a complicated posterior, so that $U(\mathbf{z})$ denotes:

$$U(\mathbf{z}) = \frac{1}{2}\left(\frac{||z_1|| - 2}{0.4}\right)^2 - \log\left(e^{-\frac{1}{2}[\frac{z_2-2}{0.6}]^2} + e^{-\frac{1}{2}[\frac{z_2+2}{0.6}]^2}\right) \ . \tag{4.18}$$

Again we decompose the variational approximation as $q_\phi(\mathbf{a}, \mathbf{z}) = q_\phi(\mathbf{z}|\mathbf{a})q_\phi(\mathbf{a})$. The bound is:

$$\log Z \geq \mathbb{E}_{q_\phi(\mathbf{a},\mathbf{z})}\left[\log \frac{e^{U(\mathbf{z})}p_\theta(\mathbf{a}|\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{a})q_\phi(\mathbf{a})}\right] \ . \tag{4.19}$$

In Figure 4.4 we see how the auxiliary variable model is able to fit the two modes of the complicated posterior, which makes them an intriguing complement to a regular one layered VAE.

Figure 4.4: The approximation $q_\phi(\mathbf{a}, \mathbf{z}) = q_\phi(\mathbf{z}|\mathbf{a})q_\phi(\mathbf{a})$ to a complex posterior $U(\mathbf{z})$. We can see how the additional auxiliary variable is able to fit the two modes.

On the basis of the above, adding an auxiliary variable to a semi-supervised model is straightforward. Similarly to the derivation in Equation 4.7 we can

(a) Generative model p_θ **(b)** Inference model q_ϕ

Figure 4.5: Graphical representation of ADGM from Maaløe et al. (2015b; 2016) (cf. **Appendix A**) with the generative model parameterized by θ (a) and the variational approximation/inference model parameterized by ϕ (b) for a three latent variable model solving the semi-supervised classification task. $\mathbf{z}$ represents the continuous latent variable, $\mathbf{a}$ the continuous auxiliary variable, $\mathbf{y}$ a partially observed latent variable, and $\mathbf{x}$ represents the observed variable.

derive the slightly more complex unsupervised bound (cf. Figure 4.5):

$$\begin{aligned}\log p_\theta(\mathbf{x}) &= \log \int_a \sum_{\mathbf{y}} \int_{\mathbf{z}} p_\theta(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{a}) d\mathbf{z} d\mathbf{a} \\ &\geq \mathbb{E}_{q_\phi(\mathbf{a}|\mathbf{x})}\left[\sum_{\mathbf{y}} q_\phi(\mathbf{y}|\mathbf{a},\mathbf{x}) \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{a},\mathbf{x},\mathbf{y})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{a})}{q_\phi(\mathbf{a},\mathbf{y},\mathbf{z}|\mathbf{x})}\right]\right] \equiv \mathcal{U}_{\phi,\theta}(\mathbf{x}) \ ,\end{aligned} \tag{4.20}$$

where

$$\frac{p_\theta(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{a})}{q_\phi(\mathbf{a},\mathbf{y},\mathbf{z}|\mathbf{x})} = \frac{p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z}) p_\theta(\mathbf{a}|\mathbf{y},\mathbf{z},\mathbf{x}) p_\theta(\mathbf{y}) p_\theta(\mathbf{z})}{q_\phi(\mathbf{a}|\mathbf{x}) q_\phi(\mathbf{y}|\mathbf{a},\mathbf{x}) q_\phi(\mathbf{z}|\mathbf{a},\mathbf{x},\mathbf{y})} \ . \tag{4.21}$$

The supervised bound is similarly derived by:

$$\begin{aligned}\log p_\theta(\mathbf{x},\mathbf{y}) &= \log \int_a \int_{\mathbf{z}} p_\theta(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{a}) d\mathbf{z} d\mathbf{a} \\ &\geq \mathbb{E}_{q_\phi(\mathbf{a}|\mathbf{x})}\left[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{a},\mathbf{x},\mathbf{y})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{a})}{q_\phi(\mathbf{a},\mathbf{z}|\mathbf{x},\mathbf{y})}\right]\right] \ ,\end{aligned} \tag{4.22}$$

where we add the cross-entropy loss as in Equation 4.10:

$$\mathcal{J}_{\phi,\theta}(\mathbf{x},\mathbf{y}) = \mathbb{E}_{q_\phi(\mathbf{a}|\mathbf{x})}\left[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{a},\mathbf{x},\mathbf{y})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{a})}{q_\phi(\mathbf{a},\mathbf{z}|\mathbf{x},\mathbf{y})}\right]\right] - \alpha \cdot \mathcal{H}(\mathbf{y}, q_\phi(\mathbf{y}|\mathbf{x},\mathbf{a})) \ . \tag{4.23}$$

The supervised and unsupervised bounds for the ADGM can be collected in $\mathcal{L}_{\phi,\theta}(\cdot)$ (cf. Equation 4.11). We now have a formulation of a semi-supervised model with an auxiliary variable that can learn a global latent representation

(a) Generative model p_θ **(b)** Inference model q_ϕ

Figure 4.6: Graphical representation of SDGM from Maaløe et al. (2016) (cf. **Appendix A**) with the generative model parameterized by θ (a) and the variational approximation/inference model parameterized by ϕ (b) for a three latent variable model solving the semi-supervised classification task. $\mathbf{z}_1$ and $\mathbf{z}_2$ represents the continuous latent variables, $\mathbf{y}$ a partially observed latent variable, and $\mathbf{x}$ the observed variable.

of the input distribution, similarly to the M1+M2 model. Compared to the M2 model, ADGM is much stronger due to the addition of the extra auxiliary variable. Furthermore, the ADGM can be trained in an end-to-end fashion, conversely to the M1+M2 model. Thereby, the ADGM can adjust the latent representation in the auxiliary variable in accordance to the semi-supervised classification task. In Maaløe et al. (2016) we also introduce a slight variant of the ADGM, where we change the auxiliary variable to be yet another latent variable. This model is named *skip deep generative model* (SDGM) for which we show a graphical representation in Figure 4.6. The difference between the ADGM and SDGM, is that we now include the additional latent variable in the generative model, thus the variable is not auxiliary anymore.

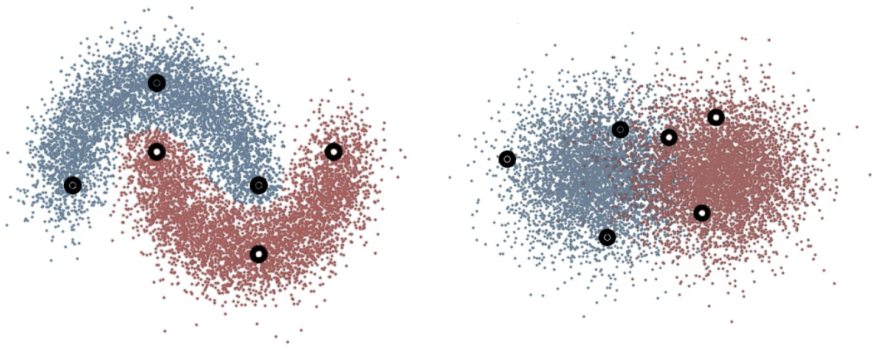
Both the ADGM and SDGM[4] significantly outperform the directly comparable M2 and M1+M2 models (cf. Table 4.1). On the more complex datasets, SVHN and NORB, the SDGM model outperforms the classification of ADGM, which is expected, since it is a stronger model. However, interestingly the ADGM performs significantly better than the SDGM on the MNIST benchmark. In order to visualize what the auxiliary variable contributes with in the semi-supervised classification, we revisit the half-moon example from earlier. Again the task is a binary classification problem, where the upper half-moon is the positive class and the lower half-moon is the negative class. However, to make this problem a semi-supervised task, we only provide 3 labeled data points from each half-moon (cf. Figure 4.7). The ADGM is able to learn this problem rather fast to a classification accuracy close to 100%, with slight variations due to the data points that are lying in the intersection between the two classes. If we visualize the latent space of the auxiliary variable, we see how it is able to distinguish

[4]We evaluate the performance of the fully unsupervised AVAE in Section 5.

	MNIST 100 LABELS	SVHN 1000 LABELS	NORB 1000 LABELS
M1+TSVM	11.82% (±0.25)	55.33% (±0.11)	18.79% (±0.05)
M2	11.97% (±1.71)	-	-
M1+M2	3.33% (±0.14)	36.02%(±0.10)	-
VAT	2.12%	24.63%	9.88%
LADDER NETWORK	1.06% (±0.37)	-	-
ADGM	**0.96**% (±0.02)	22.86%	10.06% (±0.05)
SDGM	1.32% (±0.07)	**16.61**% (±0.24)	**9.40**% (±0.04)

Table 4.1: Semi-supervised test error benchmarks from Maaløe et al. (2016) (cf. **Appendix A**) on MNIST (LeCun et al., 1998), SVHN (Netzer et al., 2011), and NORB (LeCun et al., 2004) for randomly labeled and evenly distributed data points. The lower section demonstrates the benchmarks for the Auxiliary Deep Generative Model (ADGM) and the Skip Deep Generative Model (SDGM). M1+TSVM, M2, M1+M2 are reported in Kingma et al. (2014), VAT in Miyato et al. (2015), and Ladder Network in Rasmus et al. (2015).

the clusters of the half-moons that in turn will aid the classifier.



(a) **(b)**

Figure 4.7: Visualization of the ADGM trained on a semi-supervised classification task in the half-moon example. In this example there is only provided 6 labeled data points, 3 from each half-moon. Large scalar points visualize the labeled data. (a) shows the classification prediction of the ADGM in the input space, where the blue and red color corresponds to the prediction of the upper and lower half-moon respectively. (b) is a PCA plot on the first two principal components, showing the latent space of the auxiliary variable.

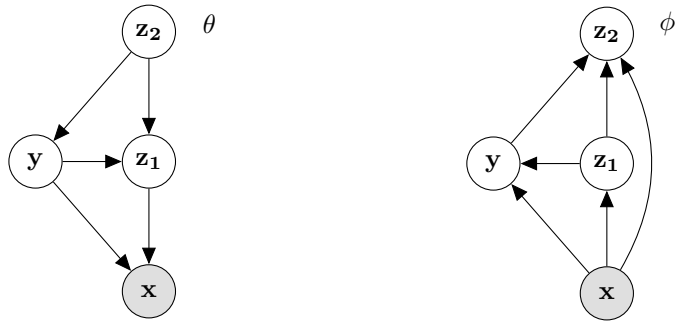
Concluding remarks Both the ADGM and SDGM have proven extremely efficient for the semi-supervised classification task, but there are still many further developments that could be explored. One of the limiting caveats of the models is the summation $\sum_{\mathbf{y}}^{K}$ which scales poorly for many categories K. In order to perform the summation as efficiently as possible, one can reorder the unsupervised input mini-batch, so that it becomes K times larger. This is much more efficient towards the runtime complexity than propagating the unsupervised mini-batch through the deep neural networks K times. However, it comes at a cost in regards to the space complexity for a large model, located in the limited amount of memory available on modern graphical processing units. Gumbel (1954); Jang et al. (2016) provides a good way to circumvent the summation, by sampling from the discrete latent variable $\mathbf{y}$.

Another interesting research area would be to utilize the bound of the ADGM and SDGM models similarly to a Bayes classifier $p(\mathbf{y}|\mathbf{x}) \propto p(\mathbf{y}, \mathbf{x})$. The way that this works is intuitively derived from the definition of the supervised bound, $\mathcal{J}_{\phi,\theta}(\mathbf{x}, \mathbf{y})$, so that $\mathbf{x}$ belongs to the class $\mathbf{y}$ for which the bound is highest. We have performed some experiments on this procedure, but did not achieve to outperform the classifier $q_\phi(\mathbf{y}|\mathbf{a}, \mathbf{x})$. However, a stronger generative model could prove to increase performance. Another approach is to utilize the two classifiers in unison. Finally, a way to utilize the bound in practice is for anomaly detection as we will see in Section 6. This could in turn prove useful for active learning.

Recently, we have seen several improvements to the semi-supervised classification performance, where GANs have proven very efficient (Salimans et al., 2016). Many of the new additions to the benchmarks on semi-supervised classification utilize much more complicated deep neural network structures and architectures, i.e. convolutions. An interesting study to the ADGM and SDGM models would be to parameterize them with deeper neural networks and convolutions in order to compare them with recent state-of-the-art.

For an implementation of the auxiliary deep generative model, go to:

`github.com/larsmaaloee/auxiliary-deep-generative-models`



(a) Generative model p_θ **(b)** Inference model q_ϕ

Figure 4.8: Graphical representation of CaGeM from Maaløe et al. (2017) (cf. **Appendix C**) with the generative model parameterized by θ (a) and the variational approximation/inference model parameterized by ϕ (b) for a three latent variable model solving the semi-supervised generation task. $\mathbf{z}_1$ and $\mathbf{z}_2$ represents the continuous latent variables, $\mathbf{y}$ a partially observed latent variable, and $\mathbf{x}$ the observed variable.

4.3 Cluster-Aware Deep Generative Models

This section cites one of the contributions in this thesis:

C Maaløe, L., Fraccaro, M., Winther, O. (2017). CaGeM: A cluster aware deep generative model. In *Neural Information Processing Systems Workshop* on Approximate Bayesian Inference.

As we briefly introduced in the beginning of the previous chapter, we formulate a clear distinction between semi-supervised classification and semi-supervised generation, where this section will elaborate on the latter. Now we have seen that deep generative models can be utilized for semi-supervised classification, the apparent question arise, to whether a small amount of labeled data can improve on the generative performance. This view on a generative process can be directly translated to cognitive science, in the sense that humans can efficiently infer a causal model from very few labeled examples (Tenenbaum et al., 2006). This pose a problem in the typical generative frameworks, i.e. GANs and VAEs, since they generally perceive the generative process as fully unsupervised. In Maaløe et al. (2017) (cf. **Appendix C**) we propose an approach, denoted *cluster-aware generative models* (CaGeM), that incorporates labeled information in order to improve on the generative performance. Similarly to the SDGM, the CaGeM is defined from three latent variables, $p_\theta(\mathbf{x}, \mathbf{y}, \mathbf{z}_1, \mathbf{z}_2)$, with the big difference being that the discrete variable $\mathbf{y}$ and the latent variable $\mathbf{z}_2$ are

no longer assumed statistically independent, $p_\theta(\mathbf{y}, \mathbf{z}_2) \neq p_\theta(\mathbf{y})p_\theta(\mathbf{z}_2)$. Instead the purpose is to improve on the generative performance by explicitly defining the natural clustering of the data in the higher latent representations (Bengio et al., 2013). This leads to the generative model (cf. Figure 4.8a):

$$p_\theta(\mathbf{x}, \mathbf{y}, \mathbf{z}_1, \mathbf{z}_2) = p_\theta(\mathbf{x}|\mathbf{y}, \mathbf{z}_1)p_\theta(\mathbf{z}_1|\mathbf{y}, \mathbf{z}_2)p_\theta(\mathbf{y}|\mathbf{z}_2)p_\theta(\mathbf{z}_2) \ , \tag{4.24}$$

where

$$\begin{aligned} &p_\theta(\mathbf{z}_2) = \mathcal{N}(\mathbf{z}_2|0, I) \ , \quad p_\theta(\mathbf{y}|\mathbf{z}_2) = \mathrm{Cat}(\mathbf{y}|\pi_\theta(\mathbf{z}_2) \\ &p_\theta(\mathbf{z}_1|\mathbf{y}, \mathbf{z}_2) = \mathcal{N}(\mathbf{z}_1|\mu_\theta(\mathbf{y}, \mathbf{z}_2), \sigma_\theta(\mathbf{y}, \mathbf{z}_2)) \\ &p_\theta(\mathbf{x}|\mathbf{z}_1, \mathbf{y}) = \mathcal{N}(\mathbf{x}|\mu_\theta(\mathbf{z}_1, \mathbf{y}), \sigma_\theta(\mathbf{z}_1, \mathbf{y})) \quad \text{or} \quad P_\theta(\mathbf{x}|\mathbf{z}_1, \mathbf{y}) = \mathcal{B}(\mathbf{x}|\mu_\theta(\mathbf{z}_1, \mathbf{y})) \ . \end{aligned} \tag{4.25}$$

As we will see below, the discrete variable $p_\theta(\mathbf{y}|\mathbf{z}_2)$, will ensure that cluster information is preserved in the highest latent representation $\mathbf{z}_2$, which also has an effect on the fact that the higher latent representation will stay active (cf. Section 3.3 for a discussion on this topic). The corresponding inference model is decomposed by (cf. Figure 4.8b):

$$q_\phi(\mathbf{y}, \mathbf{z}_1, \mathbf{z}_2|\mathbf{x}) = q_\phi(\mathbf{z}_1|\mathbf{x})q_\phi(\mathbf{y}|\mathbf{z}_1, \mathbf{x})q_\phi(\mathbf{z}_2|\mathbf{z}_1, \mathbf{y}, \mathbf{x}) \ . \tag{4.26}$$

The ELBO can be derived in a similar fashion as earlier:

$$\mathcal{U}_{\phi,\theta}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z_1}|\mathbf{x})}\left[\sum_{\mathbf{y}} q_\phi(\mathbf{y}|\mathbf{z}_1, \mathbf{x})\mathbb{E}_{q_\phi(\mathbf{z_2}|\mathbf{x},\mathbf{y},\mathbf{z_1})}\left[\log \frac{p_\theta(\mathbf{x}, \mathbf{y}, \mathbf{z}_1, \mathbf{z}_2)}{q_\phi(\mathbf{y}, \mathbf{z}_1, \mathbf{z}_2|\mathbf{x})}\right]\right] \ . \tag{4.27}$$

Conversely to the semi-supervised classification models, we specify the Categorical distributions $p_\theta(\mathbf{y}|\mathbf{z}_2)$ and $q_\phi(\mathbf{y}|\mathbf{z}_1, \mathbf{x})$, as representing the number of clusters to be modeled from the dataset, thus the representations learned in the highest latent variable $\mathbf{z}_2$ could be perceived as a form of semi-supervised clustering (Basu et al., 2002). In this work we perceive *clusters* as being a more general term than *classes*, so that the amount of classes K of a dataset would be directly translatable to the amount of clusters C but not the other way around. To exploit cluster information in the CaGeM we can define a classifier in the inference model as:

$$q_\phi(\mathbf{y}|\mathbf{x}) = \int_{\mathbf{z_1}} q_\phi(\mathbf{y}, \mathbf{z}_1|\mathbf{x})d\mathbf{z}_1 \tag{4.28}$$

$$= \int_{\mathbf{z_1}} q_\phi(\mathbf{y}|\mathbf{z}_1, \mathbf{x})q_\phi(\mathbf{z}_1|\mathbf{x})d\mathbf{z}_1 \ , \tag{4.29}$$

and from the posterior $p_\theta(\mathbf{z}_2|\mathbf{x})$ we have another classifier in the generative model given by:

$$p_\theta(\mathbf{y}|\mathbf{x}) = \int_{\mathbf{z_2}} p_\theta(\mathbf{y}|\mathbf{z}_2)p_\theta(\mathbf{z}_2|\mathbf{x})d\mathbf{z}_2 \ . \tag{4.30}$$

Model	Clusters	$\leq \log p(x)$
CaGeM-0	20	-81.92^a
CaGeM-0	5	-81.86^a
CaGeM-0	10	-81.60
CaGeM-20	10	-81.47
CaGeM-50	10	-80.49
CaGeM-100	10	-79.38

Table 4.2: Log-likelihood scores for CaGeM (Maaløe et al., 2017) (cf. **Appendix C**) trained on MNIST (LeCun et al., 1998) with 0, 20, 50 and 100 labels. The more labeled data points, the better the performance.

[a]These results are updated compared to the **Appendix C** version of the article.

The posterior in the classifier given in Equation 4.30 is intractable, which is why we need to utilize the variational approximation to estimate the classifier:

$$p_\theta(\mathbf{y}|\mathbf{x}) \approx \int_{\mathbf{z_2}} p_\theta(\mathbf{y}|\mathbf{z}_2) q_\phi(\mathbf{z}_2|\mathbf{x}) d\mathbf{z}_2 \tag{4.31}$$

$$= \int_{\mathbf{z}_2} p_\theta(\mathbf{y}|\mathbf{z}_2) \left[\int_{\mathbf{z}_1} \sum_{\tilde{y}} q_\phi(\mathbf{z}_2|\mathbf{x}, \tilde{\mathbf{y}}, \mathbf{z}_1) q_\phi(\tilde{\mathbf{y}}|\mathbf{x}, \mathbf{z}_1) q_\phi(\mathbf{z}_1|\mathbf{x}) d\mathbf{z}_1 \right] d\mathbf{z}_2 \ . \tag{4.32}$$

In Maaløe et al. (2017) we refer to the above as a *cascade* of classifiers, since the classifier $p_\theta(\mathbf{y}|\mathbf{x})$ indirectly depends on the weighting term $q_\phi(\tilde{\mathbf{y}}|\mathbf{x}, \mathbf{z}_1)$.

As was done for the M2, ADGM, and SDGM models, we add a cross-entropy term to the ELBO given in Equation 4.27. However, in order for the classification term not to have too big of an impact on the generative part of the model, we update the parameters independently. Hence, we now define the parameters of the model, such that ϕ_y and θ_y refers to the parameters of $q_{\phi_y}(\mathbf{y}|\mathbf{x})$ and $p_{\theta_y}(\mathbf{y}|\mathbf{x})$ respectively. The remainder of the parameters for $q_\phi(\mathbf{z}_1, \mathbf{z}_2|\mathbf{x})$ and $p_\theta(\mathbf{x}, \mathbf{z}_1, \mathbf{z}_2)$ are still denoted ϕ and θ. The ELBO for the semi-supervised generation task is then defined as:

$$\mathcal{L}(\mathbf{x}_l, \mathbf{y}_l, \mathbf{x}_u) = \sum_{\mathbf{x}_u} \mathcal{U}_{\phi,\theta}(\mathbf{x}_u) - \alpha \left[\sum_{\mathbf{x}_l, \mathbf{y}_l} \left(\mathcal{H}_{\theta_y}(\mathbf{y}_l, p_{\theta_y}(\mathbf{y}|\mathbf{x})) + \mathcal{H}_{\phi_y}(\mathbf{y}_l, q_{\phi_y}(\mathbf{y}|\mathbf{x})) \right) \right] , \tag{4.33}$$

where α is a scaling term, defined as a hyper-parameter during optimization and $\mathcal{H}(\cdot)$ denotes the cross-entropy. This expression means that we only consider the cross-entropy loss w.r.t. ϕ_y and θ_y, meaning that the gradient will be 0 for the remainder of the model parameters. The approach ensures that we do not *overfit* the generative model towards our labeled dataset. However, as we will

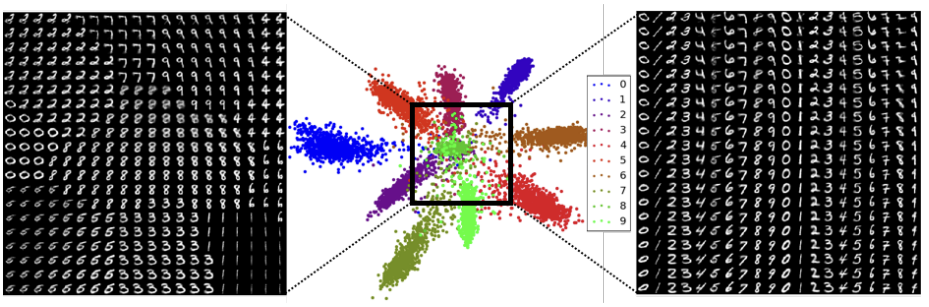


Figure 4.9: Visualization of the $\mathbf{z}_2$ variable of CaGeM (Maaløe et al., 2017) (cf. **Appendix C**) trained with 100 labeled examples from the MNIST dataset (LeCun et al., 1998). The middle plot shows on the latent space, where we generate samples by inferring the class label through $p_\theta(\mathbf{y}|\mathbf{x})$ (left) and generate samples by predefining the class (right) from a mesh grid (black bounding box).

see below, the CaGeM also proves as a very good semi-supervised classification model.

The evaluation criterion for the generative performance of CaGeM is the 5000 importance weighted ELBO given in Equation 4.27 measured in nats. In Table 4.2 we show the generative performance of CaGeM-0, CaGeM-20, CaGeM-50, and CaGeM-100, denoting 0, 20, 50, and 100 labeled data points. When we provide labeled data that is evenly distributed across categories, the amount of clusters C equals the number of classes, i.e. for the MNIST dataset $C = 10$. For the fully unsupervised example, CaGeM-0, on the other hand we show results for different numbers of clusters, 5, 10, and 20. It is interesting to see how the fully unsupervised log-likelihood achieves the best score when defining the *natural* amount of clusters. Furthermore, one can see how the performance increases significantly as a function of adding a relatively small fraction of labeled data points. In Section 5 we will compare the performance of CaGeM to other fully unsupervised models. When analyzing the latent space of variable $\mathbf{z}_2$ it becomes clear how the model utilize the class-conditional clustering. When we sample from different regions of $\mathbf{z}_2$, the classifier $p_\theta(\mathbf{y}|\mathbf{x})$ infers different classes corresponding to the region, hence different digits will be generated for the MNIST case (cf. Figure 4.9).

Besides being optimized towards semi-supervised generation, CaGeM proves similar performance to ADGM, SDGM, and other state-of-the-art semi-supervised classification models, 1.16% for 100 labeled MNIST. This comes as a surprise when comparing the log-likelihood score of a ADGM-100 to CaGeM-100. Here ADGM achieves a log-likelihood of -86.06 nats, which is significantly worse than the CaGeM (cf. Table 4.2). This indicates that the CaGeM optimize to-

wards both a semi-supervised classification and semi-supervised generation task much better than above models.

Concluding remarks CaGeM proves to be a compelling model for semi-supervised learning as a whole. The argument of using these kinds of deep generative models similarly to a Bayes classifier, brought forward in Section 4.2, is an even more interesting path to follow for the CaGeM, as compared to ADGM and SDGM, taking into account the significant generative improvement. As we will see in the next section, CaGeM is also a very efficient generative model compared to other permutation invariant models[5]. There are still many other parameterizations to explore, such as convolutional neural networks for the image generation task.

[5]When a model is *permutation invariant* it means that it is not parameterized with any spatial or temporal relationship, i.e. a densely connected neural network vs. convolutional and recurrent neural networks.

Chapter 5

Deep Generative Models for Unsupervised Learning

In the previous chapter we introduced models for semi-supervised learning that take advantage of the information in unlabeled data. This leads us to the research on purely unsupervised learning, for which models like the AVAE and CaGeM also applies. In addition we will introduce the *ladder variational auto-encoder* (Sønderby et al., 2016) (cf. **Appendix B**) that formulates yet another parameterization of the inference model in order to encourage a more expressive variational approximation. We will evaluate and compare the three permutation invariant contributions against each other. Finally we introduce our recent, preliminary work, on defining a deep generative model that combines a convolutional variant of the VAE framework and recent autoregressive modeling advances, denoted *feature map variational auto-encoder* (Maaløe and Winther, 2018) (cf. **Appendix E**).

5.1 Improving Permutation Invariant Deep Generative Models

This section cites three of the contributions in this thesis:

B Maaløe, L., Sønderby, C. K., Sønderby, S. K., Winther, O. (2016). Auxiliary deep generative models. In *Proceedings of the International Conference on Machine Learning*, pages 1445–1454.

C Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., Winther, O. (2016). Ladder variational autoencoders. In *Advances in Neural Information Processing Systems*, pages 3738-3746.

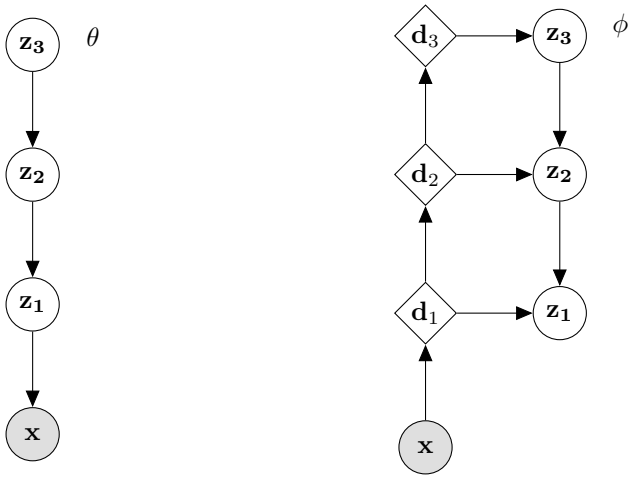
D Maaløe, L., Fraccaro, M., Winther, O. (2017). CaGeM: A cluster aware deep generative model. In *Neural Information Processing Systems Workshop* on Approximate Bayesian Inference.

5.1.1 Ladder Variational Auto-Encoders

In Sønderby et al. (2016) (cf. **Appendix B**) we introduce the *ladder variational auto-encoder* (LVAE) that, as the name implies, is highly inspired by the structure of the deterministic ladder network (Rasmus et al., 2015). The main purpose of the LVAE is to solve the lacking expressiveness of the normal VAE (discussed in Section 3.3) by defining a different variational approximation while maintaining the hierarchical generative structure (cf. Equation 3.34). One of the assumptions made in Sønderby et al. (2016) is that the inference path in the regular VAE (cf. Equation 3.36) is prone to too much noise in the beginning of training when stacking multiple layers. As we have seen in Section 3.3, this gives the model the incentive to *collapse* the latent variables. In order to avoid this tendency, and thereby achieve a more expressive approximation to the true posterior, the LVAE propose a deterministic path from the observed variable, $\mathbf{x}$, to the top latent variable, $\mathbf{z}_L$. The inference model is therefore decomposed by (cf. Figure 5.1):

$$q_\phi(\mathbf{Z}|\mathbf{x}) = q_\phi(\mathbf{z}_L|\mathbf{x}) \prod_{i=1}^{L-1} q_\phi(\mathbf{z}_i|\mathbf{z}_{i+1}, \mathbf{x}) \ . \tag{5.1}$$

We now have an inference model with a similar downwards path to the one in the generative model. This enables a trick in which we combine the generative and inference model by applying a precision-weight to the output of each latent



(a) Generative model p_θ

(b) Inference model q_ϕ

Figure 5.1: Graphical representation of LVAE from Sønderby et al. (2016) (cf. **Appendix B**) with the generative model parameterized by θ (a) and the variational approximation/inference model parameterized by ϕ (b) for a three latent variable model. $\mathbf{z}_1$, $\mathbf{z}_2$, and $\mathbf{z}_3$ represents the continuous latent variables, and $\mathbf{x}$ the observed variable.

variable:

$$q_\phi(\mathbf{z}_i|\mathbf{z}_{i+1}) = \mathcal{N}(\mathbf{z}_i|\hat{\mu}_{\phi,\theta,i}(\mathbf{z}_{i+1}), \hat{\sigma}_{\phi,\theta,i}(\mathbf{z}_{i+1})) \ , \tag{5.2}$$

$$\hat{\mu}_{\phi,\theta,i}(\mathbf{z}_{i+1}) = \frac{\mu_{\phi,i}(\mathbf{z}_{i+1}^{(\phi)})\sigma_{\phi,i}(\mathbf{z}_{i+1}^{(\phi)})^{-2} + \mu_{\theta,i}(\mathbf{z}_{i+1}^{(\theta)})\sigma_{\theta,i}(\mathbf{z}_{i+1}^{(\theta)})^{-2}}{\sigma_{\phi,i}(\mathbf{z}_{i+1}^{(\phi)})^{-2} + \sigma_{\theta,i}(\mathbf{z}_{i+1}^{(\theta)})^{-2}} \ , \tag{5.3}$$

$$\hat{\sigma}_{\phi,\theta,i}(\mathbf{z}_{i+1}) = \frac{1}{\sigma_{\phi,i}(\mathbf{z}_{i+1}^{(\phi)})^{-2} + \sigma_{\theta,i}(\mathbf{z}_{i+1}^{(\theta)})^{-2}} \ , \tag{5.4}$$

where $\mathbf{z}^{(\phi)}$ and $\mathbf{z}^{(\theta)}$ denotes the latent variable coming from the inference model and the generative model respectively. The LVAE thereby *ties* the output of the latent variables in the generative model together with the latent variable in the variational approximation. Despite the potential gain in having a deterministic path directly to the top latent variable, $\mathbf{z}_L$, our experiments proved that the latent variable would still have a tendency to collapse. Therefore, we introduce a *temperature*, that is scaling the regularization of the KL-divergence in the beginning of the parameter optimization. This process is called *warm-up*[1] and

[1]Concurrently to this research Bowman et al. (2016) introduced an equivalent procedure to warm-up.

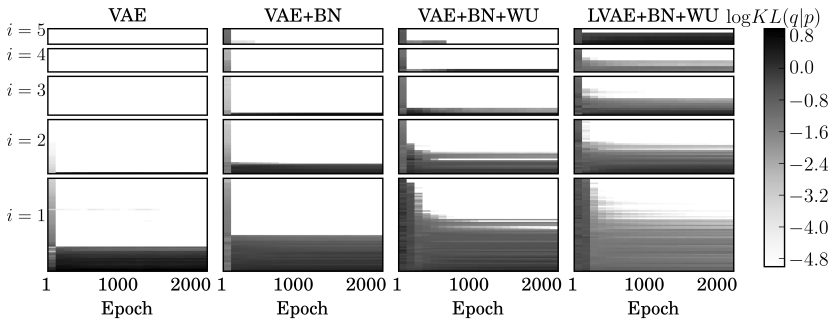


Figure 5.2: The log KL-divergence for each unit/neuron (sorted rows) for a latent variable $\mathbf{z}_i$ (box) from Sønderby et al. (2016) (cf. **Appendix B**). BN refers to batch normalization (Ioffe and Szegedy, 2015), and WU the warm-up scheme. The warm-up period for these experiments was a linear increase from 0 to 1 during the first 200 epochs. The number of units in each layer is 64-32-16-8-4.

the ELBO for the step τ is defined as:

$$\mathcal{L}_{\phi,\theta}(\mathbf{x})_\tau = \mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z}_1)\right] - T_\tau D_{KL}[q_\phi(\mathbf{Z}|\mathbf{x})||\ p_\theta(\mathbf{Z})]\ , \tag{5.5}$$

where T_τ denotes the τ^{th} element of the temperatures T[2]. In order to show that the latent variables stay active during optimization, we assume that (i) a latent variable is not collapsed if $D_{KL}^{(i)} > 0$, and (ii) that the KL-divergence is active beyond the warm-up period. The reason why (ii) is important, is that for $T_\tau = 0$ the LVAE is a regular auto-encoder. Figure 5.2 shows an analysis of the activation throughout the latent variable from the beginning of optimization to the end. As expected, the regular VAE collapse for $L > 2$ with only few active units in the second layer. By using batch normalization (Ioffe and Szegedy, 2015) throughout all layers, including the μ and σ layers of the latent variables, it results in more active units. We hypothesize that this is caused by less noisy gradients in the beginning of training, thus there is less of an incentive for the model to collapse the latent variables. The warm-up scheme improves the activation for the higher latent variables even more for the VAE; however, it is evident that the LVAE utilize much greater activation throughout all variables of the model. The question then arise to whether the model actually needs the utilization of 5 latent variables as opposed to a regular VAE with 1 or 2 latent variables. In Figure 5.3 we present a comparison of the train-dataset ELBO and the test-dataset ELBO with 1 IW sample and 5000 IW samples. It is interesting to see how the VAE with batch-normalization and warm-up performs close to the LVAE for the 5000 IW sample evaluations. However, when evaluating with

[2]Sønderby et al. (2016) applies a warm-up scheme in which the KL-divergence is scaled from 0 to 1 during the first 200 epochs.

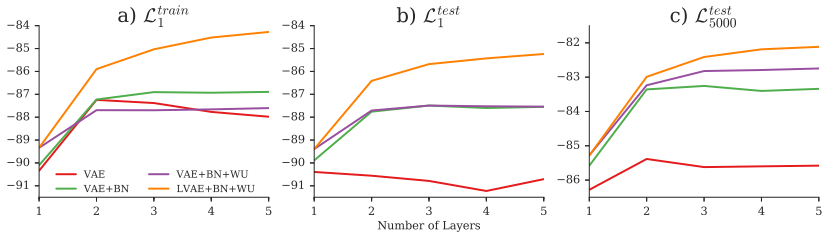


Figure 5.3: Log-likelihood scores for VAEs and LVAEs trained on permutation invariant and dynamically binarized MNIST (LeCun et al., 1998) from Sønderby et al. (2016) (cf. **Appendix B**).

just 1 IW sample it becomes apparent that there is a big discrepancy between the tightness of the bounds of the LVAE compared to the regular VAE. This indicates that the LVAE learns a better approximation to the true posterior distribution throughout the layers.

5.1.2 Comparing the Deep Generative Models

During the research of this thesis, we have developed AVAE, LVAE, and CaGeM that are all strong permutation invariant unsupervised generative models. They change the structure of the variational approximation in different ways where CaGeM also introduce a discrete latent variable. In Table 5.1 we present the comparison of the models that are directly comparable to a regular VAE. To the best of our knowledge these models are still among the state-of-the-art together with more complex models, e.g. VAEs with variational Gaussian processes (Tran et al., 2016) and discrete VAEs (Rolfe, 2017; Vahdat et al., 2018). When benchmarking these models there are several factors that make them more or less comparable:

(i) number of units in the latent variables as well as the deterministic densely connected networks,

(ii) depth of the latent variable hierarchy L that can enhance the expressiveness of the variational approximation significantly,

(iii) number of IW-samples used during optimization varies from 1 to 50, and shows a significant impact on the results,

(iv) normalization schemes, i.e. both LVAE and CaGeM use batch normalization as opposed to AVAE and IWAE. As depicted in Figure 5.2, this may have a big impact on the result.

However, some of the models are directly comparable. IWAE and AVAE (1 IW-sample) are trained equivalently, and from the result in Table 5.1 it is quite evident that we achieve a significant improvement by adding an auxiliary variable. Furthermore, the CaGeM has a significant impact over a 5 latent variable LVAE. This is somewhat surprising, since the LVAE is formulated from a much deeper hierarchy of latent variables. It indicates that the discrete variable introduce a strong addition to the VAE framework. Last but not least, it is staggering how much we can improve on the generative performance by adding a small fraction of labeled data points, CaGeM-100.

Model	L	IW	$\leq \log p(x)$
IWAE	2	1	-85.33
AVAE	2	1	-82.97
IWAE	2	50	-82.90
LVAE	5	1	-81.84
LVAE	5	10	-81.74
CaGeM-0 (C=10)	2	1	-81.60
CaGeM-100[a] (C=10)	2	1	-79.38

Table 5.1: Log-likelihood scores for AVAE (Maaløe et al., 2016), LVAE (Sønderby et al., 2016), CaGeM (Maaløe et al., 2017) (cf. **Appendix A; B; C**), and IWAE (Burda et al., 2015) trained on permutation invariant and dynamically binarized MNIST (LeCun et al., 1998). L denotes the number of continuous latent variables and IW the number of importance weighted samples used during training. In this comparison we have only included the best performing models that are directly comparable to the parameterizations of the normal VAE (Kingma and Welling, 2014; Rezende et al., 2014) (cf. Appendices for comparisons to other models, e.g. *variational Gaussian processes* (Tran et al., 2016)).

[a] CaGeM-100 utilize labeled information.

Concluding remarks An interesting perspective is that all models are complementary, thus the LVAE could easily be using an auxiliary variables $\mathbf{a}_i$ for each $\mathbf{z}_i$, and the discrete latent variable $\mathbf{y}$ from CaGeM can also be applied to the generative and inference model. This argument is not limited to these models, but also applies to other model parameterizations, such as the *variational Gaussian processes* (Tran et al., 2016), *normalizing flows* (Rezende and Mohamed, 2015), and *inverse autoregressive flows* (Kingma et al., 2016). In the next section we will present research where we are utilizing the hierarchical structure of the LVAE with an autoregressive model and spatial architecture.

For an implementation of the ladder variational auto-encoder, go to:

`github.com/larsmaaloee/variational-tensorflow`

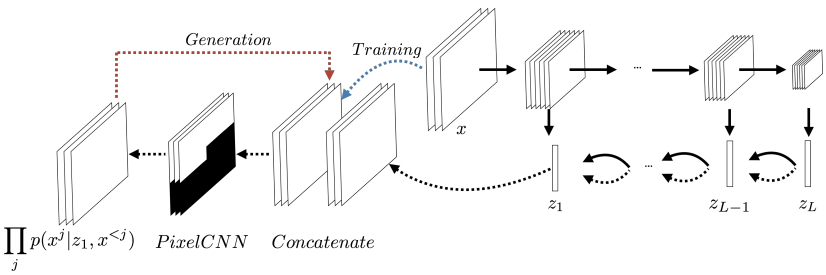


Figure 5.4: A visualization of FAME (cf. (Maaløe and Winther, 2018), **Appendix E**) where the solid lines denote the variational approximation (inference model) and dashed lines denote the generative model for training. When performing reconstructions during training, the input image is concatenated with the output of the generative model (blue) and when generating, the model follows a normal autoregressive sampling flow (red), while also using the stochastic latent variables $\mathbf{Z} = \mathbf{z}_1, ..., \mathbf{z}_L$. Both the variational approximation and the generative model follow a top-down hierarchical structure, enabling precision weighted stochastic variables in the variational approximation.

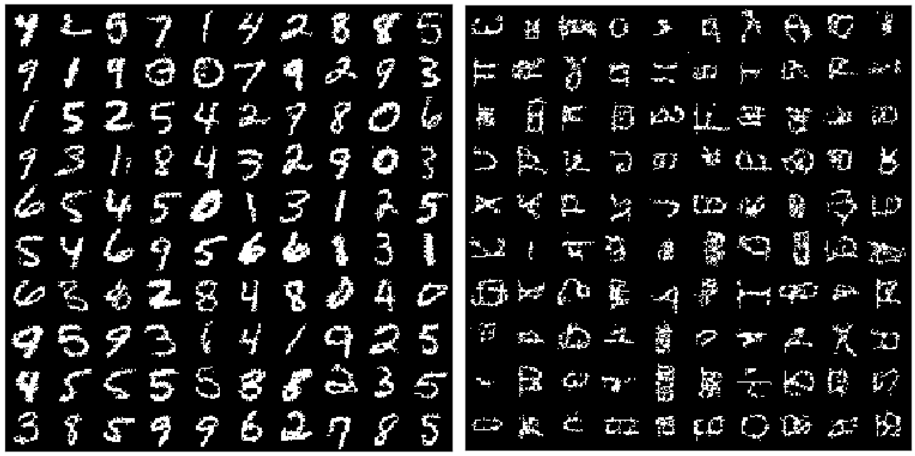
5.2 Utilizing Spatial Information in Deep Generative Models

This chapter cites one of the contributions in this thesis:

F Maaløe, L., Winther, O. (2018). Feature map variational auto-encoders. To be submitted.

The previous section presented how the unsupervised generative performance can be significantly improved by utilizing more flexible variational approximations. Recent publications utilize (non-permutation invariant) spatial and temporal information to gain significant improvements. Some of the most notable results are the deterministic autoregressive models: PixelRNN (van den Oord et al., 2016b), PixelCNN (van den Oord et al., 2016b), Gated PixelCNN (van den Oord et al., 2016c), and PixelCNN++ (Salimans et al., 2017). These models have also gained traction in audio processing with Wavenet (van den Oord et al., 2016a). The autoregressive models utilize the RNN and CNN layers to formulate a generative model where $p(\mathbf{x}_i|\mathbf{x}_{<i})$[3], and i denotes the pixel value when modeling images or the current time-step when modeling audio. One powerful feature of these models is that they are able to generate very complex

[3]For a detailed explanation on how autoregressive models are implemented, we refer the reader to van den Oord et al. (2016b) and Salimans et al. (2017).



(a) **(b)**

Figure 5.5: Random samples drawn from a $\mathcal{N}(0, I)$ distribution and propagated through the generative model of FAME (Maaløe and Winther, 2018) (cf. **Appendix E**) for the binarized MNIST (LeCun et al., 1998) (a) and OMNIGLOT (Lake et al., 2013) (b) dataset.

local information, which has been an inherent problem in the variational auto-encoder framework (Larsen et al., 2016). Furthermore, they have shown that the stacking of deep hierarchies of autoregressive layers achieves the embedding of global information. PixelCNN++ (Salimans et al., 2017) has taken this one step further by formulating the autoregressive dependency while performing dimensionality reduction, similar to an auto-encoder with skip-connections.

Variants that combine the autoregressive models with the variational auto-encoder framework has proven equivalent results to the purely autoregressive models: PixelVAE (Gulrajani et al., 2016) and *variational lossy auto-encoders* (VLAE) (Chen et al., 2017). It is noteworthy that these models learn a complementary global representation that can prove useful in the generative process. However, they do not improve on the generative performance, despite the fact that they introduce much more complexity. In this research we aim to formulate a VAE that is a stronger generative model. For this, we utilize the findings from the LVAE and complement it with convolutional layers for each deterministic layer. Next we utilize a shallow variant of the PixelCNN from van den Oord et al. (2016b). We name this model the *feature map variational auto-encoder* (FAME) (Maaløe and Winther, 2018) (cf. Figure 5.4 and **Appendix E**). The hope is that the VAE part will learn the global information and that the PixelCNN part will learn the granular details of the local representations.

Model	NLL
IWAE (Burda et al., 2015)	82.90
LVAE (Sønderby et al., 2016)	81.74
CaGeM (Maaløe et al., 2017)	81.60
DVAE (Rolfe, 2017)	80.04
VGP (Tran et al., 2016)	79.88
IAF VAE Kingma et al. (2016)	79.10
VLAE Chen et al. (2017)	78.53
FAME No Concatenation	78.73
FAME	**77.82**

Table 5.2: Negative log-likelihood performance on dynamically binarized MNIST in nats from (Maaløe and Winther, 2018), **Appendix E**. The evidence lower-bound is computed with 5000 importance weighted samples $\mathcal{L}^{5000}_{\theta,\phi}(\mathbf{x})$.

In Table 5.2 we present the results on the dynamically binarized MNIST dataset, on which the model significantly outperform previous state of the art (cf. Appendix **Appendix E** for more results and implementation details). Figure 5.5 shows generated samples from the learned model. The generative process with respect to the autoregressive model is cumbersome, hence one must generate each x_i sequentially. Therefore, we also evaluated FAME without the concatenation. This variant enables a single forward-pass in the generation model, since the model is no longer dependent on the input distribution. Surprisingly this model achieves a comparable result.

Concluding remarks The research on FAME is a *work in progress* and is included in this thesis to show some interesting directions for the VAE-based models. We are currently in the process of extending the approach towards natural images. An intriguing future direction is to improve further on the version that is not dependent on the autoregressive model approach. However, there may be a need for an even more expressive variational approximation to properly generate complex data sources such as natural images or audio.

Chapter 6

Condition Monitoring with Deep Generative Models

In this chapter we present, Maaløe et al. (2018) (cf. **Appendix D**), a real-life application of the modeling approaches presented throughout this thesis. The application is within data-driven condition monitoring in photovoltaic (PV) systems[1]. This is one in many scenarios where semi-supervised machine learning models are essential. The data sources are not directly interpretable; hence it is costly to acquire enough data for a supervised model to generalize. Furthermore, it is difficult to label data according to a specific class, due to the causal dependency within conditions. We will give a background on condition monitoring in PV systems followed by our findings.

[1]Photovoltaics refer to the process of utilizing solar cells in converting sun energy into electrons.

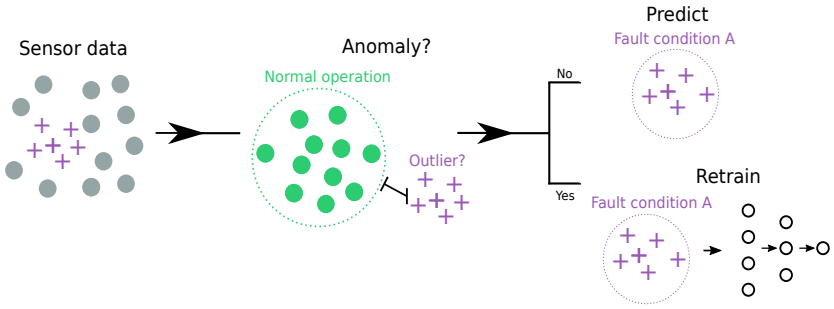


Figure 6.1: Visualization of the proposed condition monitoring system through semi-supervised learning from Maaløe et al. (2018) (cf. **Appendix D**). The semi-supervised learning models proposed in this project have the ability to capture anomalies by utilizing the formulation of the ELBO. This in turn can then indicate whether an input condition is an anomaly and whether the model must be retrained. If the condition is not anomalous, the model will be able to categorize the condition, either within the normal operation of the PV plant or within a fault category.

6.1 Condition Monitoring in Energy Production

Energy production systems such as PV and wind power plants are an ever increasing source of energy across the globe. Two of the key reasons for the rapid growth lie in the increase in power utilization and cost efficiency (Spataru et al., 2016; Bach-Andersen, 2017). To uphold maximized yield in such energy systems, maintenance and operation must be as streamlined as possible. Here condition monitoring becomes a vital instrument. Condition monitoring concerns the process of analyzing different system activities in order to discern a condition that results in a decrease in power utilization. This condition could potentially be a fault, but it can also be within normal operation, such as changes in weather patterns. An example is the monitoring of drive train vibration data in wind turbines (Bach-Andersen et al., 2017). Vibration data contains subtle patterns showing signs of degradation that can be monitored and detected prior to the system breaking. In PV systems we can divide the source of power loss into three categories:

(i) **Optical loss** concerns conditions such as soiling, shading, and snow (Laukamp et al., 2002). In a large-scale PV system this can be a result of many different events. They can be easily identified through visual inspection. However, in large-scale PV systems spanning more than 40 km^2, visual

inspection is far from easily conveyed[2]. Detecting these conditions from sensor measurements can be difficult, since the resulting power loss is highly irregular.

(ii) **Electrical circuit degradation** denotes open-circuit faults, closed-circuit faults, and more subtle faults due to partial degradation. Such partial degradation faults can be difficult to detect and cause an increase in series resistance of each PV array (King et al., 2000; Yang et al., 2013). The effect of an increase in series resistance will result in a drop of voltage.

(iii) **Solar cell degradation** has a wide variety of sources, such as thermo-mechanical stress, voltage stress, and seasonal variations (King et al., 2000; Köntges et al., 2014). The patterns for solar cell degradation can be extremely subtle and thereby hard to detect. However, the identification of these patterns can prove very valuable, since they may be the result of system-wide problems, e.g. bad installation practices.

There has been much effort in detecting certain conditions in energy production systems (Spataru et al., 2016; Alzahrani et al., 2017; Bach-Andersen et al., 2017; Bach-Andersen, 2017; Ali et al., 2017; Silvestre et al., 2013; Jiang and Maskell, 2015). However, to our knowledge there have been no attempts in performing end-to-end condition monitoring through semi-supervised learning. In our opinion this is vital in order to perform efficient monitoring, since it is intractable to perceive all possible conditions for the supervised learning paradigm, both because of the cumbersome labeling process and due to a risk of overlapping categories.

6.2 Evaluating the Condition Monitoring System

This chapter cites two of the contributions in this thesis:

A Maaløe, L., Sønderby, S. K., Sønderby, C. K., Winther, O. (2016). Auxiliary deep generative models. In *Proceedings of the International Conference on Machine Learning*, pages 1445–1454.

D Maaløe, L., Spataru, S. V., Sera, D., Winther, O. (2018). Condition monitoring in photovoltaic systems by semi-supervised machine learning. Submitted to IEEE Transactions of Industrial Informatics.

[2] $\approx \frac{2}{3}$ the size of Manhattan, New York, USA.

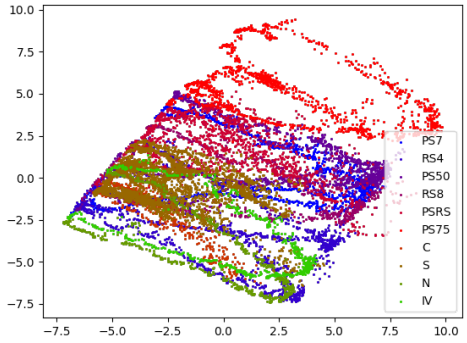


Figure 6.2: PCA (principal components 1 and 2) on the latent space of the $\mathbf{z}_1$ variable from Maaløe et al. (2018) (cf. **Appendix D**). The visualized SDGM is trained on a fully supervised learning task on the dataset proposed in Table 6.1.

In Maaløe et al. (2018) (cf. **Appendix D**) we use a dataset containing approximately 2 months of PV system sensor data (cf. Table 6.1) divided into 10 different categories. The dataset contains sensor data considering the current (I), voltage (V), in-plane irradiance (R), external temperature (TExt), PV module temperature (TMod), and wind speed (W). We deploy the SDGM (Maaløe et al., 2016) (cf. **Appendix A**) as an end-to-end condition monitoring machine learning framework. In order to detect anomalous conditions (cf. Figure 6.1) we utilize the ELBO from Equation 4.20 of which we can label a value significantly below the ELBO of the training data, as anomalous. Complementary to the anomaly detection we use the classifier $q_\phi(\mathbf{y}|\mathbf{z}_1, \mathbf{x})$ to detect conditions.

To test the framework we define 3 experiments:

(i) Benchmark the SDGM model against a densely connected neural network (MLP) with equivalent architecture to $q_\phi(\mathbf{y}|\mathbf{z}_1, \mathbf{x})$ and a multi-label linear regression model (MLR) on a fully supervised learning task.

(ii) Compare the 3 models when only presented with a fraction of labeled data during training. The SDGM will then utilize the unlabeled data, and we will achieve a measure of the added value of the semi-supervised approach.

(iii) Finally, we simulate real-life condition monitoring for a PV system by progressively adding data from a new system condition. We initialize the models with only 1 sample from each category, and then we add only the data from one category at a time. The data added contains 500 labeled samples and the remainder as unlabeled data.

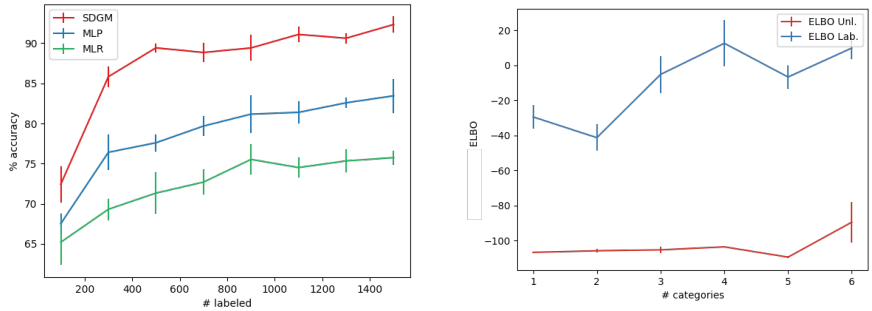
Condition/Fault Class	Description	Samples
PS7	Uniform shading on all lower cells of the modules	10.68%
RS4	50% increase in string series resistance	10.18%
PS50	Partial shading on 50% of a submodule	10.83%
RS8	100% increase in string series resistance	5.11%
PSRS	Combined 50% shading on a submodule with 50% increase in string Rs	10.93%
PS75	Shading on 50% of a submodule + 25% of another submodule	10.60%
C	Cloudy sky day	4.60%
S	Snow on the modules	27.64%
N	Clear sky day	4.67%
IV	Shading on 3/4 of cell area of 6 submodules	4.78%

Table 6.1: Dataset used for testing the semi-supervised learning model framework for condition monitoring (cf. Maaløe et al. (2018), **Appendix D**). The dataset is divided into 10 different categories where some lies within the normal conditions and others are characterized as fault states.

	Accuracy I,V	Accuracy I,V,G,TExt TMod,W
MLR	51.62%	77.33%
MLP	77.81%	89.11%
SDGM	**79.06**%	**92.47**%

Table 6.2: Supervised baseline on MLR, MLP and SDGM for a *simple* sensor input, {I, V}, and a more *complex* input, {I, V, G, TExt, TMod, and W} (cf. Maaløe et al. (2018), **Appendix D**).

In the fully supervised comparison from Table 6.2, we see a clear improvement of the non-linear models, SDGM and MLP, over the linear, MLR. This indicates that the categories are difficult to discriminate and therefore need the transformations of non-linear layers. Furthermore, it comes as no surprise that the addition of more sensor data {G, TExt, TMod, and W} improves the performance significantly. What is interesting, is that $q_\phi(\mathbf{y}|\mathbf{z}_1, \mathbf{x})$ outperform the equivalently expressive MLP. We hypothesize that this may be a result of *fuzzy* annotations. This is highly probable, especially in real-life systems, where a category is not always evidently one thing over another. E.g. a binary classification task over a natural image dataset, consisting of *dogs* and *cats*, may have some samples where dogs and cats are evenly represented. It may not be straightforward for a human annotator to label such samples. The SDGM has the ability to correlate the classification with the internal representation of the continuous latent variables during training, which can result in a correction of some of these fuzzy labels. From Figure 6.2 we see how the SDGM captures the



(a) (b)

Figure 6.3: (a) A comparison between SDGM Maaløe et al. (2018) (cf. **Appendix D**), MLP, and MLR for different fractions of labeled data. The labeled samples are randomly sampled and evenly distributed across categories (cf. Table 6.1) and all models are trained 10 times for each fraction. (b) SDGMs trained from datasets where we randomly subsample a single data-point from each category and then progressively add 500 randomly labeled data points for each category and train a new SDGM for each progression. The plot shows the ELBO for the data categories included during training (ELBO Lab.) and the data categories that are not included during training (ELBO Unl.). The categories that are progressively added is following the order of Table 6.1, i.e. first *{PS7}*, next *{PS7, RS4}*, until reaching *{PS7, RS4, PS50, RS8, PSRS, PS75}*.

patterns in the latent variable that follows the progression of the sun.

Figure 6.3a presents the increased value of utilizing the semi-supervised SDGM over the supervised MLP and MLR. When increasing the number of labeled data points we see that the improvement of utilizing the unlabeled data remains. An interesting finding is that, when utilizing 1500 labeled data points in the SDGM, the performance improves over the equivalent supervised experiment. This again, underlines indications of the *fuzzy annotation hypothesis*.

Finally, Figure 6.3b presents the real-life simulation. We show the ELBO for the categories that are not included during training, denoted *ELBO Unl.*, and the ELBO for the categories included during training, denoted *ELBO Lab.*. It is evident that SDGM is able to detect anomalies, and by deciding on a threshold value this can be used to re-label incoming sensor data that is not represented in the data distribution that was trained upon.

Concluding remarks We have presented a framework for condition monitoring by using the semi-supervised learning models proposed in this thesis. It would be interesting to further investigate the improved performance of the semi-supervised model over an equivalent supervised model on machine learning benchmark datasets, e.g. Imagenet (Deng et al., 2009).

A limitation to this study is that the dataset used is a small-scale dataset compared to a real-life PV system. Future improvements to the research would be to deploy the framework on a large-scale PV system and investigate on transfer learning capabilities.

Chapter 7

Conclusion

We have presented new approaches to the combination of deep learning and probabilistic modeling. They significantly improve unsupervised and semi-supervised learning within classification and generation tasks on real-life applications as well as machine learning benchmark datasets. The main contributions consist of:

AVAE The auxiliary variational auto-encoder achieves to make the variational approximation more flexible by introducing an auxiliary variable to the VAE model. This leads to a significant increase in performance for the unsupervised generation task.

ADGM The auxiliary deep generative model formulates an end-to-end approach that learns a global representation of the dataset in the latent auxiliary variable in order to improve on semi-supervised classification.

SDGM The skip deep generative model reformulates the generative model of the ADGM in order to propose a 2-latent variable hierarchical model that improves even further on the semi-supervised classification task.

LVAE The ladder variational auto-encoder changes the variational approximation of the VAE so that it forms a top-down structure similarly to the hierarchical generative model. This is used to share representations across the inference model and generative model during inference, and with adhering tricks

(warm-up and batch-normalization), results in a significant improvement on the unsupervised generation task.

CaGeM The cluster-aware generative model considers another view on semi-supervised learning that utilize the labeled information as additional information in order to achieve a better unsupervised and semi-supervised generation performance. The model formulates a cascade of classifiers that helps in clustering the input data within the approximated posterior.

FAME The feature map variational auto-encoder changes the architecture of the neural networks to model spatial information, while utilizing a recently proposed autoregressive framework. The model thereby improves significantly on image generation tasks.

The approaches presented in this thesis are all somewhat complementary, which leaves a variety of interesting future studies. A study could consider, embedding the auxiliary variable in the LVAE that in turn could have the potential to express much more complicated distributions in the latent variables, while maintaining the depth of the generative model. Another study could be to combine the top-down approach introduced in the LVAE with the discrete latent variable in the CaGeM in order to achieve improved cluster-aware generation. It would also be interesting to consider the strong representational learning of the FAME in a semi-supervised classification framework such as ADGM or SDGM.

There also exist a vast amount of contributions that are outside the scope of this study that would be interesting to pursue. Here, the learning of more complex posterior distributions by utilizing methods such as normalizing flows (Rezende and Mohamed, 2015), inverse autoregressive flows (Kingma et al., 2016), and variational Gaussian processes (Tran et al., 2016), would be interesting directions to follow.

The ADGM, SDGM, and CaGeM all include a discrete variable. Discrete variables are difficult to sample from, which is why we tended to marginalization. This could prove to be a cumbersome process in domains with many categories. Jang et al. (2016) have utilized the *Gumbel trick* (Gumbel, 1954) that has the potential of alleviating the marginalization. By introducing this into the semi-supervised frameworks we can potentially speed up training time and convergence rates significantly and scale to much larger datasets.

During the research we have also performed experiments on other data modalities and modeling paradigms, such as multimodal learning on text and images, VAEs in off-policy reinforcement learning, and semi-supervised learning for topic modeling (similarly to Maaløe et al. (2015a)). These studies have shown intriguing results but have not been pursued any further.

A final note on the research is that, since everything is based on deep neural networks, the different optimization and regularization tricks have a big impact on model performance. This is considered throughout the analysis of this thesis. However, in order to formalize a proper comparison to recent state-of-the-art modeling approaches for unsupervised and semi-supervised learning it would be necessary to parameterize the models and define the optimization algorithm and adhering hyperparameter settings equivalently. With the rapid pace of machine learning research, such a *review* study is close to incomprehensible. However, it would be interesting to formulate one that at least approximates a fair comparison.

Appendix A

Auxiliary Deep Generative Models

In this appendix we include the arxiv.org version of:

Maaløe, L., Sønderby, S. K., Sønderby, C. K., Winther, O. (2016). Auxiliary deep generative models. In *Proceedings of the International Conference on Machine Learning*, pages 1445–1454.

Auxiliary Deep Generative Models

Lars Maaløe[1] LARSMA@DTU.DK
Casper Kaae Sønderby[2] CASPERKAAE@GMAIL.COM
Søren Kaae Sønderby[2] SKAAESONDERBY@GMAIL.COM
Ole Winther[1,2] OLWI@DTU.DK
[1]Department of Applied Mathematics and Computer Science, Technical University of Denmark
[2]Bioinformatics Centre, Department of Biology, University of Copenhagen

Abstract

Deep generative models parameterized by neural networks have recently achieved state-of-the-art performance in unsupervised and semi-supervised learning. We extend deep generative models with auxiliary variables which improves the variational approximation. The auxiliary variables leave the generative model unchanged but make the variational distribution more expressive. Inspired by the structure of the auxiliary variable we also propose a model with two stochastic layers and skip connections. Our findings suggest that more expressive and properly specified deep generative models converge faster with better results. We show state-of-the-art performance within semi-supervised learning on MNIST, SVHN and NORB datasets.

1. Introduction

Stochastic backpropagation, deep neural networks and approximate Bayesian inference have made deep generative models practical for large scale problems (Kingma, 2013; Rezende et al., 2014), but typically they assume a mean field latent distribution where all latent variables are independent. This assumption might result in models that are incapable of capturing all dependencies in the data. In this paper we show that deep generative models with more expressive variational distributions are easier to optimize and have better performance. We increase the flexibility of the model by introducing auxiliary variables (Agakov and Barber, 2004) allowing for more complex latent distributions. We demonstrate the benefits of the increased flexibility by achieving state-of-the-art performance in the semi-supervised setting for the MNIST (LeCun et al., 1998), SVHN (Netzer et al., 2011) and NORB (LeCun et al., 2004) datasets.

Recently there have been significant improvements within the semi-supervised classification tasks. Kingma et al. (2014) introduced a deep generative model for semi-supervised learning by modeling the joint distribution over data and labels. This model is difficult to train end-to-end with more than one layer of stochastic latent variables, but coupled with a pretrained feature extractor it performs well. Lately the Ladder network (Rasmus et al., 2015; Valpola, 2014) and virtual adversarial training (VAT) (Miyato et al., 2015) have improved the performance further with end-to-end training.

In this paper we train deep generative models with multiple stochastic layers. The *Auxiliary Deep Generative Models (ADGM)* utilize an extra set of auxiliary latent variables to increase the flexibility of the variational distribution (cf. Sec. 2.2). We also introduce a slight change to the ADGM, a 2-layered stochastic model with skip connections, the *Skip Deep Generative Model (SDGM)* (cf. Sec. 2.4). Both models are trainable end-to-end and offer state-of-the-art performance when compared to other semi-supervised methods. In the paper we first introduce toy data to demonstrate that:

(i) The auxiliary variable models can fit complex latent distributions and thereby improve the variational lower bound (cf. Sec. 4.1 and 4.3).

(ii) By fitting a complex half-moon dataset using only six labeled data points the ADGM utilizes the data manifold for semi-supervised classification (cf. Sec. 4.2).

For the benchmark datasets we show (cf. Sec. 4.4):

(iii) State-of-the-art results on several semi-supervised classification tasks.

(iv) That multi-layered deep generative models for semi-supervised learning are trainable end-to-end without pre-training or feature engineering.

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

2. Auxiliary deep generative models

Recently Kingma (2013); Rezende et al. (2014) have coupled the approach of variational inference with deep learning giving rise to powerful probabilistic models constructed by an inference neural network $q(z|x)$ and a generative neural network $p(x|z)$. This approach can be perceived as a variational equivalent to the deep auto-encoder, in which $q(z|x)$ acts as the encoder and $p(x|z)$ the decoder. However, the difference is that these models ensures efficient inference over continuous distributions in the latent space z with automatic relevance determination and regularization due to the KL-divergence. Furthermore, the gradients of the variational upper bound are easily defined by backpropagation through the network(s). To keep the computational requirements low the variational distribution $q(z|x)$ is usually chosen to be a diagonal Gaussian, limiting the expressive power of the inference model.

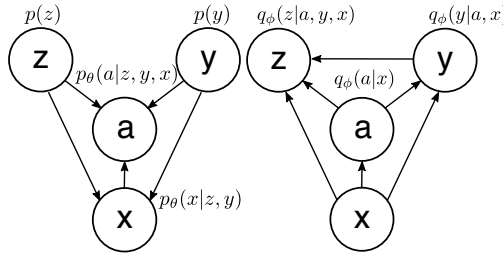
In this paper we propose a variational auxiliary variable approach (Agakov and Barber, 2004) to improve the variational distribution: The generative model is extended with variables a to $p(x, z, a)$ such that the original model is invariant to marginalization over a: $p(x, z, a) = p(a|x, z)p(x, z)$. In the variational distribution, on the other hand, a is used such that marginal $q(z|x) = \int q(z|a, x)p(a|x)da$ is a general non-Gaussian distribution. This hierarchical specification allows the latent variables to be correlated through a, while maintaining the computational efficiency of fully factorized models (cf. Fig. 1). In Sec. 4.1 we demonstrate the expressive power of the inference model by fitting a complex multimodal distribution.

2.1. Variational auto-encoder

The variational auto-encoder (VAE) has recently been introduced as a powerful method for unsupervised learning. Here a latent variable generative model $p_\theta(x|z)$ for data x is parameterized by a deep neural network with parameters θ. The parameters are inferred by maximizing the variational lower bound of the likelihood $-\sum_i \mathcal{U}_{\text{VAE}}(x_i)$ with

$$\begin{aligned} \log p(x) &= \log \int_z p(x, z)dz \\ &\geq \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p_\theta(x|z)p_\theta(z)}{q_\phi(z|x)}\right] \\ &\equiv -\mathcal{U}_{\text{VAE}}(x) \,. \end{aligned} \tag{1}$$

The inference model $q_\phi(z|x)$ is parameterized by a second deep neural network. The inference and generative parameters, θ and ϕ, are jointly trained by optimizing Eq. 1 with stochastic gradient ascent, where we use the reparameterization trick for backpropagation through the Gaussian latent variables (Kingma, 2013; Rezende et al., 2014).



(a) Generative model P. (b) Inference model Q.

Figure 1. Probabilistic graphical model of the ADGM for semi-supervised learning. The incoming joint connections to each variable are deep neural networks with parameters θ and ϕ.

2.2. Auxiliary variables

We propose to extend the variational distribution with auxiliary variables a: $q(a, z|x) = q(z|a, x)q(a|x)$ such that the marginal distribution $q(z|x)$ can fit more complicated posteriors $p(z|x)$. In order to have an unchanged generative model, $p(x|z)$, it is required that the joint mode $p(x, z, a)$ gives back the original $p(x, z)$ under marginalization over a, thus $p(x, z, a) = p(a|x, z)p(x, z)$. Auxiliary variables are used in the EM algorithm and Gibbs sampling and have previously been considered for variational learning by Agakov and Barber (2004). Concurrent with this work Ranganath et al. (2015) have proposed to make the parameters of the variational distribution stochastic, which leads to a similar model. It is important to note that in order not to fall back to the original VAE model one has to require $p(a|x, z) \neq p(a)$, see Agakov and Barber (2004) and App. A. The auxiliary VAE lower bound becomes

$$\begin{aligned} \log p(x) &= \log \int_a \int_z p(x, a, z)dadz \\ &\geq \mathbb{E}_{q_\phi(a,z|x)}\left[\log \frac{p_\theta(a|z, x)p_\theta(x|z)p(z)}{q_\phi(a|x)q_\phi(z|a, x)}\right] \\ &\equiv -\mathcal{U}_{\text{AVAE}}(x) \,. \end{aligned} \tag{2}$$

with $p_\theta(a|z, x)$ and $q_\phi(a|x)$ diagonal Gaussian distributions parameterized by deep neural networks.

2.3. Semi-supervised learning

The main focus of this paper is to use the auxiliary approach to build semi-supervised models that learn classifiers from labeled and unlabeled data. To encompass the class information we introduce an extra latent variable y. The generative model P is defined as $p(y)p(z)p_\theta(a|z, y, x)p_\theta(x|y, z)$ (cf. Fig. 1a):

$$p(z) = \mathcal{N}(z|0, \mathrm{I}), \tag{3}$$

$$p(y) = \text{Cat}(y|\pi), \tag{4}$$

$$p_\theta(a|z, y, x) = f(a; z, y, x, \theta), \tag{5}$$

$$p_\theta(x|z, y) = f(x; z, y, \theta) \,, \tag{6}$$

where a, y, z are the auxiliary variable, class label, and latent features, respectively. $\text{Cat}(\cdot)$ is a multinomial distribution, where y is treated as a latent variable for the unlabeled data points. In this study we only experimented with categorical labels, however the method applies to other distributions for the latent variable y. $f(x; z, y, \theta)$ is iid categorical or Gaussian for discrete and continuous observations x. $p_\theta(\cdot)$ are deep neural networks with parameters θ. The inference model is defined as $q_\phi(a|x)q_\phi(z|a, y, x)q_\phi(y|a, x)$ (cf. Fig. 1b):

$$q_\phi(a|x) = \mathcal{N}(a|\mu_\phi(x), \text{diag}(\sigma^2_\phi(x))), \tag{7}$$

$$q_\phi(y|a, x) = \text{Cat}(y|\pi_\phi(a, x)), \tag{8}$$

$$q_\phi(z|a, y, x) = \mathcal{N}(z|\mu_\phi(a, y, x), \text{diag}(\sigma^2_\phi(a, y, x))) \,. \tag{9}$$

In order to model Gaussian distributions $p_\theta(a|z, y, x)$, $p_\theta(x|z, y)$, $q_\phi(a|x)$ and $q_\phi(z|a, y, x)$ we define two separate outputs from the top deterministic layer in each deep neural network, $\mu_{\phi\vee\theta}(\cdot)$ and $\log\sigma^2_{\phi\vee\theta}(\cdot)$. From these outputs we are able to approximate the expectations $\mathbb{E}$ by applying the reparameterization trick.

The key point of the ADGM is that the auxiliary unit a introduce a latent feature extractor to the inference model giving a richer mapping between x and y. We can use the classifier (9) to compute probabilities for unlabeled data x_u being part of each class and to retrieve a cross-entropy error estimate on the labeled data x_l. This can be used in cohesion with the variational lower bound to define a good objective function in order to train the model end-to-end.

VARIATIONAL LOWER BOUND

We optimize the model by maximizing the lower bound on the likelihood (cf. App. B for more details). The variational lower bound on the marginal likelihood for a single *labeled data point* is

$$\begin{aligned}\log p(x, y) &= \log \int_a \int_z p(x, y, a, z) dz da \\ &\geq \mathbb{E}_{q_\phi(a,z|x,y)}\left[\log \frac{p_\theta(x, y, a, z)}{q_\phi(a, z|x, y)}\right] \\ &\equiv -\mathcal{L}(x, y) \,,\end{aligned} \tag{10}$$

with $q_\phi(a, z|x, y) = q_\phi(a|x)q_\phi(z|a, y, x)$. For unlabeled data we further introduce the variational distribution for y, $q_\phi(y|a, x)$:

$$\begin{aligned}\log p(x) &= \log \int_a \int_y \int_z p(x, y, a, z) dz dy da \\ &\geq \mathbb{E}_{q_\phi(a,y,z|x)}\left[\log \frac{p_\theta(x, y, a, z)}{q_\phi(a, y, z|x)}\right] \\ &\equiv -\mathcal{U}(x) \,,\end{aligned} \tag{11}$$

with $q_\phi(a, y, z|x) = q_\phi(z|a, y, x)q_\phi(y|a, x)q_\phi(a|x)$.

The classifier (9) appears in $-\mathcal{U}(x_u)$, but not in $-\mathcal{L}(x_l, y_l)$. The classification accuracy can be improved by introducing an explicit classification loss for labeled data:

$$\begin{aligned}\mathcal{L}_l(x_l, y_l) = \\ \mathcal{L}(x_l, y_l) + \alpha \cdot \mathbb{E}_{q_\phi(a|x_l)}[-\log q_\phi(y_l|a, x_l)] \,,\end{aligned} \tag{12}$$

where α is a weight between generative and discriminative learning. The α parameter is set to $\beta \cdot \frac{N_l+N_u}{N_l}$, where β is a scaling constant, N_l is the number of labeled data points and N_u is the number of unlabeled data points. The objective function for labeled and unlabeled data is

$$\mathcal{J} = \sum_{(x_l,y_l)} \mathcal{L}_l(x_l, y_l) + \sum_{(x_u)} \mathcal{U}(x_u) \,. \tag{13}$$

2.4. Two stochastic layers with skip connections

Kingma et al. (2014) proposed a model with two stochastic layers but were unable to make it converge end-to-end and instead resorted to layer-wise training. In our preliminary analysis we also found that this model: $p_\theta(x|z_1)p_\theta(z_1|z_2, y)p(z_2)p(y)$ failed to converge when trained end-to-end. On the other hand, the auxiliary model can be made into a two-layered stochastic model by simply reversing the arrow between a and x in Fig. 1a. We would expect that if the auxiliary model works well in terms of convergence and performance then this two-layered model (a is now part of the generative model): $p_\theta(x|y, a, z)p_\theta(a|z, y)p(z)p(y)$ should work even better because it is a more flexible generative model. The variational distribution is unchanged: $q_\phi(z|y, x, a)q_\phi(y|a, x)q_\phi(a|x)$. We call this the *Skip Deep Generative Model (SDGM)* and test it alongside the auxiliary model in the benchmarks (cf. Sec. 4.4).

3. Experiments

The SDGM and ADGM are each parameterized by 5 neural networks (NN): (1) auxiliary inference model $q_\phi(a|x)$, (2) latent inference model $q_\phi(z|a, y, x)$, (3) classification model $q_\phi(y|a, x)$, (4) generative model $p_\theta(a|\cdot)$, and (5) the generative model $p_\theta(x|\cdot)$.

The neural networks consists of M fully connected hidden layers with h_j denoting the output of a layer $j = 1, ..., M$. All hidden layers use rectified linear activation functions. To compute the approximations of the stochastic variables we place two independent output layers after h_M, μ and $\log\sigma^2$. In a forward-pass we are propagating the input x through the neural network by

$$h_M = \texttt{NN}(x) \tag{14}$$

$$\mu = \texttt{Linear}(h_M) \tag{15}$$

$$\log\sigma^2 = \texttt{Linear}(h_M) \,, \tag{16}$$

with `Linear` denoting a linear activation function. We then approximate the stochastic variables by applying the reparameterization trick using the μ and $\log \sigma^2$ outputs.

In the unsupervised toy example (cf. Sec. 4.1) we applied 3 hidden layers with $\dim(h) = 20$, $\dim(a) = 4$ and $\dim(z) = 2$. For the semi-supervised toy example (cf. Sec. 4.2) we used two hidden layers of $\dim(h) = 100$ and $\dim(a, z) = 10$.

For all the benchmark experiments (cf. Sec. 4.4) we parameterized the deep neural networks with two fully connected hidden layers. Each pair of hidden layers was of size $\dim(h) = 500$ or $\dim(h) = 1000$ with $\dim(a, z) = 100$ or $\dim(a, z) = 300$. The generative model was $p(y)p(z)p_\theta(a|z, y)p_\theta(x|z, y)$ for the ADGM and the SDGM had the augmented $p_\theta(x|a, z, y)$. Both have unchanged inference models (cf. Fig. 1b).

All parameters are initialized using the Glorot and Bengio (2010) scheme. The expectation over the a and z variables were performed by Monte Carlo sampling using the reparameterization trick (Kingma, 2013; Rezende et al., 2014) and the average over y by exact enumeration so

$$\mathbb{E}_{q_\phi(a,y,z|x)}\left[f(a, x, y, z)\right] \approx \frac{1}{N_{\text{samp}}} \sum_{i}^{N_{\text{samp}}} \sum_{y} q_\phi(y|a_i, x) f(a_i, x, y, z_{yi}), \tag{17}$$

with $a_i \sim q(a|x)$ and $z_{yi} \sim q(z|a, y, x)$.

For training, we have used the *Adam* (Kingma and Ba, 2014) optimization framework with a learning rate of 3e-4, exponential decay rate for the 1st and 2nd moment at 0.9 and 0.999, respectively. The β constant was between 0.1 and 2 throughout the experiments.

The models are implemented in Python using Theano (Bastien et al., 2012), Lasagne (Dieleman et al., 2015) and Parmesan libraries[1].

For the MNIST dataset we have combined the training set of 50000 examples with the validation set of 10000 examples. The test set remained as is. We used a batch size of 200 with half of the batch always being the 100 labeled samples. The labeled data are chosen randomly, but distributed evenly across classes. To speed up training, we removed the columns with a standard deviation below 0.1 resulting in an input size of $\dim(x) = 444$. Before each epoch the normalized MNIST images were binarized by sampling from a Bernoulli distribution with mean parameter set to the pixel intensities.

For the SVHN dataset we used the vectorized and cropped training set $\dim(x) = 3072$ with classes from 0 to 9, combined with the *extra* set resulting in 604388 data points. The test set is of size 26032. We trained on the *small* NORB dataset consisting of 24300 training samples and an equal amount of test samples distributed across 5 classes: *animal*, *human*, *plane*, *truck*, *car*. We normalized all NORB images following Miyato et al. (2015) using image pairs of 32x32 resulting in a vectorized input of $\dim(x) = 2048$. The labeled subsets consisted of 1000 evenly distributed labeled samples. The batch size for SVHN was 2000 and for NORB 200, where half of the batch was labeled samples. To avoid the phenomenon on modeling discretized values with a real-valued estimation (Uria et al., 2013), we added uniform noise between 0 and 1 to each pixel value. We normalized the NORB dataset by 256 and the SVHN dataset by the standard deviation on each color channel. Both datasets were assumed Gaussian distributed for the generative models $p_\theta(x|\cdot)$.

[1]Implementation is available in a repository named auxiliary-deep-generative-models on `github.com`.

4. Results

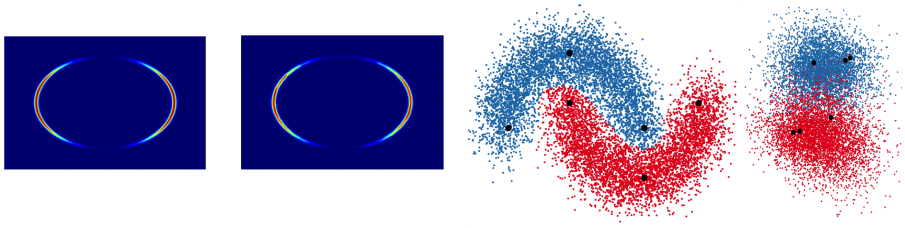
In this section we present two toy examples that shed light on how the auxiliary variables improve the distribution fit. Thereafter we investigate the unsupervised generative log-likelihood performance followed by semi-supervised classification performance on several benchmark datasets. We demonstrate state-of-the-art performance and show that adding auxiliary variables increase both classification performance and convergence speed (cf. Sec. 3 for details).

4.1. Beyond Gaussian latent distributions

In variational auto-encoders the inference model $q_\phi(z|x)$ is parameterized as a fully factorized Gaussian. We demonstrate that the auxiliary model can fit complicated posterior distributions for the latent space. To do this we consider the 2D potential model $p(z) = \exp(U(z))/Z$ (Rezende and Mohamed, 2015) that leads to the bound

$$\log Z \geq \mathbb{E}_{q_\phi(a,z)}\left[\log \frac{\exp(U(z))p_\theta(a|z)}{q_\phi(a)q_\phi(z|a)}\right]. \tag{18}$$

Fig. 2a shows the true posterior and Fig. 2b shows a density plot of z samples from $a \sim q_\phi(a)$ and $z \sim q_\phi(z|a)$ from a trained ADGM. This is similar to the findings of Rezende and Mohamed (2015) in which they demonstrate that by using normalizing flows they can fit complicated posterior distributions. The most frequent solution found in optimization is not the one shown, but one where Q fits only one of the two equivalent modes. The one and two mode solution will have identical values of the bound so it is to be expected that the simpler single mode solution will be easier to infer.



(a) (b) (c) (d)

Figure 2. (a) True posterior of the prior $p(z)$. (b) The approximation $q_\phi(z|a)q_\phi(a)$ of the ADGM. (c) Prediction on the half-moon data set after 10 epochs with only 3 labeled data points (black) for each class. (d) PCA plot on the 1st and 2nd principal component of the corresponding auxiliary latent space.

4.2. Semi-supervised learning on two half-moons

To exemplify the power of the ADGM for semi-supervised learning we have generated a 2D synthetic dataset consisting of two half-moons (top and bottom), where $(x_{\text{top}}, y_{\text{top}}) = (cos([0,\pi]), sin([0,\pi]))$ and $(x_{\text{bottom}}, y_{\text{bottom}}) = (1 - cos([0,\pi]), 1 - sin([0,\pi]) - 0.5)$, with added Gaussian noise. The training set contains 1e4 samples divided into batches of 100 with 3 labeled data points in each class and the test set contains 1e4 samples. A good semi-supervised model will be able to learn the data manifold for each of the half-moons and use this together with the limited labeled information to build the classifier.

The ADGM converges close to 0% classification error in 10 epochs (cf. Fig. 2c), which is much faster than an equivalent model without the auxiliary variable that converges in more than 100 epochs. When investigating the auxiliary variable we see that it finds a discriminating internal representation of the data manifold and thereby aids the classifier (cf. Fig. 2d).

4.3. Generative log-likelihood performance

We evaluate the generative performance of the unsupervised auxiliary model, AVAE, using the MNIST dataset. The inference and generative models are defined as

$$q_\phi(a, z|x) = q_\phi(a_1|x)q_\phi(z_1|a_1, x) \prod_{i=2}^{L} q_\phi(a_i|a_{i-1}, x)q_\phi(z_i|a_i, z_{i-1}) \,, \tag{19}$$

$$p_\theta(x, a, z) = p_\theta(x|z_1)p(z_L)p_\theta(a_L|z_L) \prod_{i=1}^{L-1} p_\theta(z_i|z_{i+1})p_\theta(a_i|z_{\geq i}) \,. \tag{20}$$

where L denotes the number of stochastic layers.

We report the lower bound from Eq. (2) for 5000 importance weighted samples and use the same training and parameter settings as in Sønderby et al. (2016) with warm-up[2], batch normalization and 1 Monte Carlo and IW sample for training.

	$\leq \log p(x)$
VAE+NF, L=1 (Rezende and Mohamed, 2015)	−85.10
IWAE, L=1, IW=1 (Burda et al., 2015)	−86.76
IWAE, L=1, IW=50 (Burda et al., 2015)	−84.78
IWAE, L=2, IW=1 (Burda et al., 2015)	−85.33
IWAE, L=2, IW=50 (Burda et al., 2015)	−82.90
VAE+VGP, L=2 (Tran et al., 2015)	−81.90
LVAE, L=5, IW=1 (Sønderby et al., 2016)	−82.12
LVAE, L=5, FT, IW=10 (Sønderby et al., 2016)	−81.74
Auxiliary VAE (AVAE), L=1, IW=1	−84.59
Auxiliary VAE (AVAE), L=2, IW=1	−82.97

Table 1. Unsupervised test log-likelihood on permutation invariant MNIST for the normalizing flows VAE (VAE+NF), importance weighted auto-encoder (IWAE), variational Gaussian process VAE (VAE+VGP) and Ladder VAE (LVAE) with FT denoting the finetuning procedure from Sønderby et al. (2016), IW the importance weighted samples during training, and L the number of stochastic latent layers $z_1, .., z_L$.

We evaluate the negative log-likelihood for the 1 and 2 layered AVAE. We found that warm-up was crucial for activation of the auxiliary variables. Table 1 shows log-likelihood scores for the permutation invariant MNIST dataset. The methods are not directly comparable, except for the Ladder VAE (LVAE) (Sønderby et al., 2016), since the training is performed differently. However, they give a good indication on the expressive power of the auxiliary variable model. The AVAE is performing better than the VAE with normalizing flows (Rezende and Mohamed, 2015) and the importance weighted auto-encoder with 1 IW sample (Burda et al., 2015). The results are also comparable to the Ladder VAE with 5 latent layers (Sønderby et al., 2016) and variational Gaussian process VAE (Tran et al., 2015). As shown in Burda et al. (2015) and Sønderby et al. (2016) increasing the IW samples and annealing the learning rate will likely increase the log-likelihood.

[2]Temperature on the KL-divergence going from 0 to 1 within the first 200 epochs of training.

	MNIST 100 LABELS	SVHN 1000 LABELS	NORB 1000 LABELS
M1+TSVM (KINGMA ET AL., 2014)	11.82% (±0.25)	55.33% (±0.11)	18.79% (±0.05)
M1+M2 (KINGMA ET AL., 2014)	3.33% (±0.14)	36.02%(±0.10)	-
VAT (MIYATO ET AL., 2015)	2.12%	24.63%	9.88%
LADDER NETWORK (RASMUS ET AL., 2015)	1.06% (±0.37)	-	-
AUXILIARY DEEP GENERATIVE MODEL (ADGM)	**0.96**% (±0.02)	22.86%	10.06% (±0.05)
SKIP DEEP GENERATIVE MODEL (SDGM)	1.32% (±0.07)	**16.61**% (±0.24)	**9.40**% (±0.04)

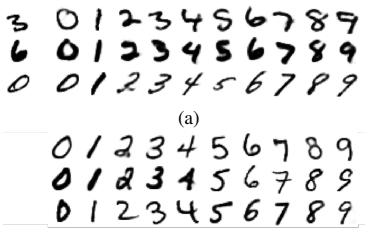
Table 2. Semi-supervised test error % benchmarks on MNIST, SVHN and NORB for randomly labeled and evenly distributed data points. The lower section demonstrates the benchmarks of the contribution of this article.

4.4. Semi-supervised benchmarks

MNIST EXPERIMENTS

Table 2 shows the performance of the ADGM and SDGM on the MNIST dataset. The ADGM's convergence to around 2% is fast (around 200 epochs), and from that point the convergence speed declines and finally reaching 0.96% (cf. Fig. 5). The SDGM shows close to similar performance and proves more stable by speeding up convergence, due to its more advanced generative model. We achieved the best results on MNIST by performing multiple Monte Carlo samples for $a \sim q_\phi(a|x)$ and $z \sim q_\phi(z|a, y, x)$.

A good explorative estimate of the models ability to comprehend the data manifold, or in other words be as close to the posterior distribution as possible, is to evaluate the generative model. In Fig. 3a we show how the SDGM, trained on *only* 100 labeled data points, has learned to separate style and class information. Fig 3b shows random samples from the generative model.

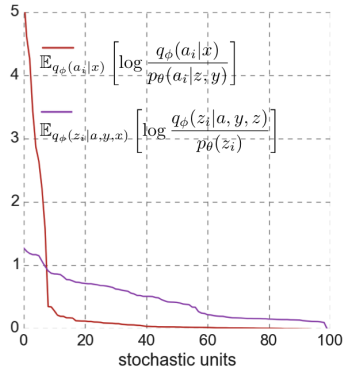


(a)

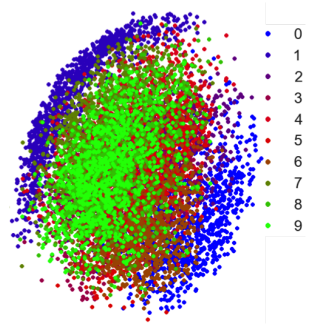
(b)

Figure 3. MNIST analogies. (a) Forward propagating a data point x (left) through $q_\phi(z|a, x)$ and generate samples $p_\theta(x|y_j, z)$ for each class label y_j (right). (b) Generating a sample for each class label from random generated Gaussian noise; hence with no use of the inference model.

Fig. 4a demonstrate the information contribution from the stochastic unit a_i and z_j (subscripts i and j denotes a unit) in the SDGM as measured by the average over the test set of the KL-divergence between the variational distribution and the prior. Units with little information content will be close to the prior distribution and the KL-divergence term will thus be close to 0. The number of clearly activated units in z and a is quite low $\sim$ 20, but there is a tail of slightly active units, similar results have been reported by Burda et al. (2015). It is still evident that we have information flowing through both variables though. Fig. 2d and 4b shows clustering in the auxiliary space for both the ADGM and SDGM respectively.



(a)



(b)

Figure 4. SDGM trained on 100 labeled MNIST. (a) The KL-divergence for units in the latent variables a and z calculated by the difference between the approximated value and its prior. (b) PCA on the 1st and 2nd principal component of the auxiliary latent space.

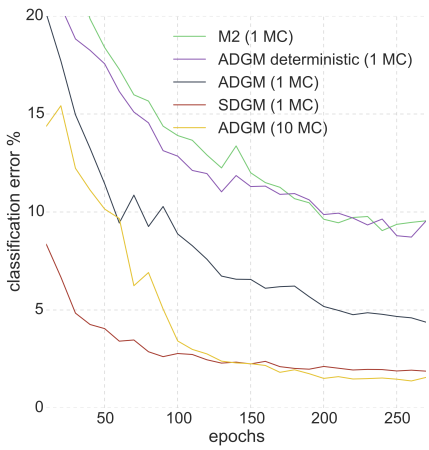


Figure 5. 100 labeled MNIST classification error % evaluated every 10 epochs between equally optimized SDGM, ADGM, M2 (Kingma et al., 2014) and an ADGM with a deterministic auxiliary variable.

In order to investigate whether the stochasticity of the auxiliary variable a or the network depth is essential to the models performance, we constructed an ADGM with a deterministic auxiliary variable. Furthermore we also implemented the M2 model of Kingma et al. (2014) using the exact same hyperparameters as for learning the ADGM. Fig. 5 shows how the ADGM outperforms both the M2 model and the ADGM with deterministic auxiliary variables. We found that the convergence of the M2 model was highly unstable; the one shown is the best obtained.

SVHN & NORB EXPERIMENTS

From Table 2 we see how the SDGM outperforms VAT with a relative reduction in error rate of more than 30% on the SVHN dataset. We also tested the model performance, when we omitted the SVHN *extra* set from training. Here we achieved a classification error of 29.82%. The improvements on the NORB dataset was not as significant as for SVHN with the ADGM being slightly worse than VAT and the SDGM being slightly better than VAT.

On SVHN the model trains to around 19% classification error in 100 epochs followed by a decline in convergence speed. The NORB dataset is a significantly smaller dataset and the SDGM converges to around 12% in 100 epochs. We also trained the NORB dataset on single images as opposed to image pairs (half the dataset) and achieved a classification error around 13% in 100 epochs.

For Gaussian input distributions, like the image data of SVHN and NORB, we found the SDGM to be more stable than the ADGM.

5. Discussion

The ADGM and SDGM are powerful deep generative models with relatively simple neural network architectures. They are trainable end-to-end and since they follow the principles of variational inference there are multiple improvements to consider for optimizing the models like using the importance weighted bound or adding more layers of stochastic variables. Furthermore we have only proposed the models using a Gaussian latent distribution, but the model can easily be extended to other distributions (Ranganath et al., 2014; 2015).

One way of approaching the stability issues of the ADGM, when training on Gaussian input distributions x is to add a *temperature* weighting between discriminative and stochastic learning on the KL-divergence for a and z when estimating the variational lower bound (Sønderby et al., 2016). We find similar problems for the Gaussian input distributions in van den Oord et al. (2016), where they restrict the dataset to ordinal values in order to apply a softmax function for the output of the generative model $p(x|\cdot)$. This discretization of data is also a possible solution. Another potential stabilizer is to add batch normalization (Ioffe and Szegedy, 2015) that will ensure normalization of each output batch of a fully connected hidden layer.

A downside to the semi-supervised variational framework is that we are summing over all classes in order to evaluate the variational bound for unlabeled data. This is a computationally costly operation when the number of classes grow. In this sense, the Ladder network has an advantage. A possible extension is to sample y when calculating the unlabeled lower bound $-\mathcal{U}(x_u)$, but this may result in gradients with high variance.

The framework is implemented with fully connected layers. VAEs have proven to work well with convolutional layers so this could be a promising step to further improve the performance. Finally, since we expect that the variational bound found by the auxiliary variable method is quite tight, it could be of interest to see whether the bound for $p(x, y)$ may be used for classification in the Bayes classifier manner $p(y|x) \propto p(x, y)$.

6. Conclusion

We have introduced a novel framework for making the variational distributions used in deep generative models more expressive. In two toy examples and the benchmarks we investigated how the framework uses the auxiliary variables to learn better variational approximations. Finally we have demonstrated that the framework gives state-of-the-art performance in a number of semi-supervised benchmarks and is trainable end-to-end.

A. Auxiliary model specification

In this appendix we study the theoretical optimum of the auxiliary variational bound found by functional derivatives of the variational objective. In practice we will resort to restricted deep network parameterized distributions. But this analysis nevertheless shed some light on the properties of the optimum. Without loss of generality we consider only auxiliary a and latent z: $p(a,z) = p(z)p(a|z)$, $p(z) = f(z)/Z$ and $q(a,z) = q(z|a)q(a)$. The results can be extended to the full semi-supervised setting without changing the overall conclusion. The variational bound for the auxiliary model is

$$\log Z \geq \mathbb{E}_{q(a,z)} \left[\log \frac{f(z)p(a|z)}{q(z|a)q(a)} \right] . \tag{21}$$

We can now take the functional derivative of the bound with respect to $p(a|z)$. This gives the optimum $p(a|z) = q(a,z)/q(z)$, which in general is intractable because it requires marginalization: $q(z) = \int q(z|a)q(a)da$.

One may also restrict the generative model to an uninformed a-model: $p(a,z) = p(z)p(a)$. Optimizing with respect to $p(a)$ we find $p(a) = q(a)$. When we insert this solution into the variational bound we get

$$\int q(a)\, \mathbb{E}_{q(z|a)} \left[\log \frac{f(z)}{q(z|a)} \right] da . \tag{22}$$

The solution to the optimization with respect to $q(a)$ will simply be a δ-function at the value of a that optimizes the variational bound for the z-model. So we fall back to a model for z without the auxiliary as also noted by Agakov and Barber (2004).

We have tested the uninformed auxiliary model in semi-supervised learning for the benchmarks and we got competitive results for MNIST but not for the two other benchmarks. We attribute this to two factors: in semi-supervised learning we add an additional classification cost so that the generic form of the objective is

$$\log Z \geq \mathbb{E}_{q(a,z)} \left[\log \frac{f(z)p(a)}{q(z|a)q(a)} + g(a) \right] , \tag{23}$$

we keep $p(a)$ fixed to a zero mean unit variance Gaussian and we use deep iid models for $f(z)$, $q(z|a)$ and $q(a)$. This taken together can lead to at least a local optimum which is different from the collapse to the pure z-model.

B. Variational bounds

In this appendix we give an overview of the variational objectives used. The generative model $p_\theta(x,a,y,z)$ for the *Auxiliary Deep Generative Model* and the *Skip Deep Generative Model* are defined as

ADGM:

$$p_\theta(x,a,y,z) = p_\theta(x|y,z)p_\theta(a|x,y,z)p(y)p(z) . \tag{24}$$

SDGM:

$$p_\theta(x,a,y,z) = p_\theta(x|a,y,z)p_\theta(a|x,y,z)p(y)p(z) . \tag{25}$$

The lower bound $-\mathcal{L}(x,y)$ on the labeled log-likelihood is defined as

$$\begin{aligned} \log p(x,y) &= \log \int_a \int_z p_\theta(x,y,a,z)dzda \\ &\geq \mathbb{E}_{q_\phi(a,z|x,y)} \left[\log \frac{p_\theta(x,y,a,z)}{q_\phi(a,z|x,y)} \right] \equiv -\mathcal{L}(x,y) , \end{aligned} \tag{26}$$

where $q_\phi(a,z|x,y) = q_\phi(a|x)q_\phi(z|a,y,x)$. We define the function $f(\cdot)$ to be $f(x,y,a,z) = \log \frac{p_\theta(x,y,a,z)}{q_\phi(a,z|x,y)}$. In the lower bound for the unlabeled data $-\mathcal{U}(x)$ we treat the discrete y[3] as a latent variable. We rewrite the lower bound in the form of Kingma et al. (2014):

$$\begin{aligned} \log p(x) &= \log \int_a \sum_y \int_z p_\theta(x,y,a,z)dzda \\ &\geq \mathbb{E}_{q_\phi(a,y,z|x)} \left[f(\cdot) - \log q_\phi(y|a,x) \right] \\ &= \mathbb{E}_{q_\phi(a|x)} \sum_y q_\phi(y|a,x) \mathbb{E}_{q_\phi(z|a,x)} \left[f(\cdot) \right] + \\ &\quad \mathbb{E}_{q_\phi(a|x)} \underbrace{\left[-\sum_y q_\phi(y|a,x) \log q_\phi(y|a,x) \right]}_{\mathcal{H}(q_\phi(y|a,x))} \\ &= \mathbb{E}_{q_\phi(a|x)} \left[\sum_y q_\phi(y|a,x) \mathbb{E}_{q_\phi(z|a,x)} \left[f(\cdot) \right] + \right. \\ &\quad \left. \mathcal{H}(q_\phi(y|a,x)) \right] \\ &\equiv -\mathcal{U}(x) , \end{aligned} \tag{27}$$

where $\mathcal{H}(\cdot)$ denotes the entropy. The objective function of $-\mathcal{L}(x,y)$ and $-\mathcal{U}(x)$ are given in Eq. (12) and Eq. (13).

[3] y is assumed to be multinomial but the model can easily be extended to different distributions.

Acknowledgements

We thank Durk P. Kingma and Shakir Mohamed for helpful discussions. This research was supported by the Novo Nordisk Foundation, Danish Innovation Foundation and the NVIDIA Corporation with the donation of TITAN X and Tesla K40 GPUs.

References

Agakov, F. and Barber, D. (2004). An Auxiliary Variational Method. In *Neural Information Processing*, volume 3316 of *Lecture Notes in Computer Science*, pages 561–566. Springer Berlin Heidelberg.

Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I. J., Bergeron, A., Bouchard, N., and Bengio, Y. (2012). Theano: new features and speed improvements. In *Deep Learning and Unsupervised Feature Learning, workshop at Neural Information Processing Systems*.

Burda, Y., Grosse, R., and Salakhutdinov, R. (2015). Importance Weighted Autoencoders. *arXiv preprint arXiv:1509.00519*.

Dieleman, S., Schlter, J., Raffel, C., Olson, E., Sønderby, S. K., Nouri, D., van den Oord, A., and and, E. B. (2015). Lasagne: First release.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS10).*, pages 249–256.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of International Conference of Machine Learning*, pages 448–456.

Kingma, D. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P., Rezende, D. J., Mohamed, S., and Welling, M. (2014). Semi-Supervised Learning with Deep Generative Models. In *Proceedings of the International Conference on Machine Learning*, pages 3581–3589.

Kingma, Diederik P; Welling, M. (2013). Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2278–2324.

LeCun, Y., Huang, F. J., and Bottou, L. (2004). Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 97–104.

Miyato, T., Maeda, S.-i., Koyama, M., Nakae, K., and Ishii, S. (2015). Distributional Smoothing with Virtual Adversarial Training. *arXiv preprint arXiv:1507.00677*.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *Deep Learning and Unsupervised Feature Learning, workshop at Neural Information Processing Systems 2011*.

Ranganath, R., Tang, L., Charlin, L., and Blei, D. M. (2014). Deep exponential families. *arXiv preprint arXiv:1411.2581*.

Ranganath, R., Tran, D., and Blei, D. M. (2015). Hierarchical variational models. *arXiv preprint arXiv:1511.02386*.

Rasmus, A., Berglund, M., Honkala, M., Valpola, H., and Raiko, T. (2015). Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, pages 3532–3540.

Rezende, D. J. and Mohamed, S. (2015). Variational Inference with Normalizing Flows. In *Proceedings of the International Conference of Machine Learning*, pages 1530–1538.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *arXiv preprint arXiv:1401.4082*.

Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., and Winther, O. (2016). Ladder variational autoencoders. *arXiv preprint arXiv:1602.02282*.

Tran, D., Ranganath, R., and Blei, D. M. (2015). Variational Gaussian process. *arXiv preprint arXiv:1511.06499*.

Uria, B., Murray, I., and Larochelle, H. (2013). Rnade: The real-valued neural autoregressive density-estimator. In *Advances in Neural Information Processing Systems*, pages 2175–2183.

Valpola, H. (2014). From neural pca to deep unsupervised learning. *arXiv preprint arXiv:1411.7783*.

van den Oord, A., Nal, K., and Kavukcuoglu, K. (2016). Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*.

Appendix B

Ladder Variational Autoencoders

In this appendix we include the arxiv.org version of:

Sønderby, C. K., Raiko, Tapani, Maaløe, L., Sønderby, S. K., Winther, O. (2016). Ladder variational autoencoders. In *Advances in Neural Information Processing Systems*, pages 3738-3746.

Ladder Variational Autoencoders

Casper Kaae Sønderby[*] casperkaae@gmail.com
Tapani Raiko[†] tapani.raiko@aalto.fi
Lars Maaløe[‡] larsma@dtu.dk
Søren Kaae Sønderby[*] skaaesonderby@gmail.com
Ole Winther[*,‡] olwi@dtu.dk

Abstract

Variational autoencoders are powerful models for unsupervised learning. However deep models with several layers of dependent stochastic variables are difficult to train which limits the improvements obtained using these highly expressive models. We propose a new inference model, the Ladder Variational Autoencoder, that recursively corrects the generative distribution by a data dependent approximate likelihood in a process resembling the recently proposed Ladder Network. We show that this model provides state of the art predictive log-likelihood and tighter log-likelihood lower bound compared to the purely bottom-up inference in layered Variational Autoencoders and other generative models. We provide a detailed analysis of the learned hierarchical latent representation and show that our new inference model is qualitatively different and utilizes a deeper more distributed hierarchy of latent variables. Finally, we observe that batch normalization and deterministic warm-up (gradually turning on the KL-term) are crucial for training variational models with many stochastic layers.

1 Introduction

The recently introduced variational autoencoder (VAE) [9, 18] provides a framework for deep generative models. In this work we study how the variational inference in such models can be improved while not changing the generative model. We introduce a new inference model using the same top-down dependency structure both in the inference and generative models achieving state-of-the-art generative performance.

VAEs, consisting of hierarchies of conditional stochastic variables, are highly expressive models retaining the computational efficiency of fully factorized models, Figure 1 a). Although highly flexible these models are difficult to optimize for deep hierarchies due to multiple layers of conditional stochastic layers. The VAEs considered here are trained by optimizing a variational approximate posterior lower bounding the intractable true posterior. Recently used inference are calculated purely bottom-up with no interaction between the inference and generative models [9, 17, 18]. We propose a new structured inference model using the same top-down dependency structure both in the inference and generative models. Here the approximate posterior distribution can be viewed as merging information from a bottom up computed approximate likelihood with top-down prior information from the generative distribution, see Figure 1 b). The sharing of information (and parameters) with the generative model gives the inference model knowledge of the current state of the generative model in each layer and the top down-pass recursively corrects the generative distribution with the data dependent approximate log-likelihood using a simple precision-weighted addition. This

[*]Bioinformatics Centre, Department of Biology, University of Copenhagen, Denmark
[†]Department of Computer Science, Aalto University, Finland
[‡]Department of Applied Mathematics and Computer Science, Technical University of Denmark

Submitted to 29th Conference on Neural Information Processing Systems (NIPS 2016).

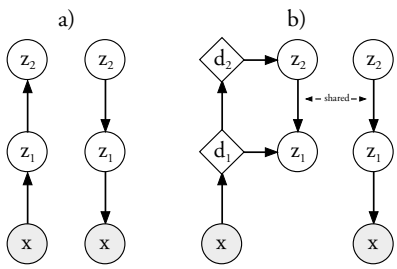


Figure 1: Inference (or encoder/recognition) and generative (or decoder) models for a) VAE and b) LVAE. Circles are stochastic variables and diamonds are deterministic variables.

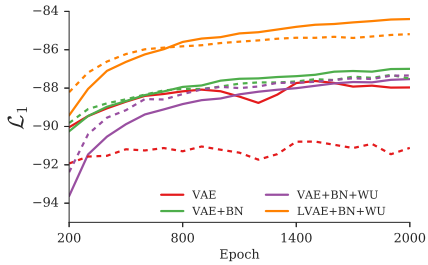


Figure 2: MNIST train (*full lines*) and test (*dashed lines*) set log-likelihood using one importance sample during training. The LVAE improves performance significantly over the regular VAE.

parameterization allows interactions between the *bottom-up* and *top-down* signals resembling the recently proposed Ladder Network [21, 16], and we therefore denote it Ladder-VAE (LVAE). For the remainder of this paper we will refer to VAEs as both the inference and generative model seen in Figure 1 a) and similarly LVAE as both the inference and generative model in Figure 1 b). We stress that the VAE and LVAE models only differ in the inference model, however these have similar number of parameters, whereas the generative models are identical.

Previous work on VAEs have been restricted to shallow models with one or two layers of stochastic latent variables. The performance of such models are constrained by the restrictive mean field approximation to the intractable posterior distribution. We found that purely bottom-up inference normally used in VAEs and gradient ascent optimization are only to a limited degree able to utilize the two layers of stochastic latent variables. We initially show that a warm-up period [1, 15, Section 6.2] to support stochastic units staying active in early training and batch normalization (BN) [6] can significantly improve performance of VAEs. Using these VAE models as competitive baselines we show that LVAE improves the generative performance achieving as good or better performance than other (often complicated) methods for creating flexible variational distributions such as: The Variational Gaussian Processes [20], Normalizing Flows [17], Importance Weighted Autoencoders [2] or Auxiliary Deep Generative Models[12]. Compared to the bottom-up inference in VAEs we find that LVAE: 1) have better generative performance 2) provides a tighter bound on the true log-likelihood and 3) can utilize deeper and more distributed hierarchies of stochastic variables. Lastly we study the learned latent representations and find that these differ qualitatively between the LVAE and VAE with the LVAE capturing more high level structure in the datasets.

In summary our contributions are:

- A new inference model combining an approximate Gaussian likelihood with the generative model resulting in better generative performance than the normally used bottom-up VAE inference
- We provide a detailed study of the learned latent distributions and show that LVAE learns both a deeper and more distributed representation when compared to VAE
- We show that a deterministic warm-up period and batch normalization are important for training deep stochastic models.

2 Methods

VAEs and LVAEs simultaneously train a generative model $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})$ for data $\mathbf{x}$ using latent variables $\mathbf{z}$, and an inference model $q_\phi(\mathbf{z}|\mathbf{x})$ by optimizing a variational lower bound to the likelihood $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z})d\mathbf{z}$. In the generative model p_θ, the latent variables $\mathbf{z}$ are split into L

layers $\mathbf{z}_i$, $i = 1 \dots L$ as follows:

$$p_\theta(\mathbf{z}) = p_\theta(\mathbf{z}_L) \prod_{i=1}^{L-1} p_\theta(\mathbf{z}_i|\mathbf{z}_{i+1}) \tag{1}$$

$$p_\theta(\mathbf{z}_i|\mathbf{z}_{i+1}) = \mathcal{N}\left(\mathbf{z}_i|\mu_{p,i}(\mathbf{z}_{i+1}), \sigma^2_{p,i}(\mathbf{z}_{i+1})\right), \;\; p_\theta(\mathbf{z}_L) = \mathcal{N}\left(\mathbf{z}_L|\mathbf{0}, \mathbf{I}\right) \tag{2}$$

$$p_\theta(\mathbf{x}|\mathbf{z}_1) = \mathcal{N}\left(\mathbf{x}|\mu_{p,0}(\mathbf{z}_1), \sigma^2_{p,0}(\mathbf{z}_1)\right) \text{ or } P_\theta(\mathbf{x}|\mathbf{z}_1) = \mathcal{B}\left(\mathbf{x}|\mu_{p,0}(\mathbf{z}_1)\right) \tag{3}$$

where observation models is matching either continuous-valued (Gaussian $\mathcal{N}$) or binary-valued (Bernoulli $\mathcal{B}$) data, respectively. We use subscript p (and q) to highlight if μ or σ^2 sigma belongs to the generative or inference distributions respectively. The hierarchical specification allows the lower layers of the latent variables to be highly correlated but still maintain the computational efficiency of fully factorized models. The variational principle provides a tractable lower bound on the log likelihood which can be used as a training criterion $\mathcal{L}$.

$$\log p(\mathbf{x}) \geq E_{q_\phi(z|x)}\left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right] = \mathcal{L}(\theta, \phi; \mathbf{x}) \tag{4}$$

$$= -KL(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) + E_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right], \tag{5}$$

where *KL* is the Kullback-Leibler divergence. A strictly tighter bound on the likelihood may be obtained at the expense of a K-fold increase of samples by using the importance weighted bound [2]:

$$\log p(\mathbf{x}) \geq E_{q_\phi(z^{(1)}|x)} \dots E_{q_\phi(z^{(K)}|x)}\left[\log \sum_{k=1}^{K} \frac{p_\theta(\mathbf{x}, \mathbf{z}^{(\mathbf{k})})}{q_\phi(\mathbf{z}^{(\mathbf{k})}|\mathbf{x})}\right] \geq \mathcal{L}_K(\theta, \phi; \mathbf{x}) \,. \tag{6}$$

The generative and inference parameters, θ and ϕ, are jointly trained by optimizing Eq. (5) using stochastic gradient descent where we use the reparametrization trick for stochastic backpropagation through the Gaussian latent variables [9, 18]. The $KL[q_\phi|p_\theta]$ is calculated analytically at each layer when possible and otherwise approximated using Monte Carlo sampling.

2.1 Variational autoencoder inference model

VAE inference models are parameterized as a bottom-up process similar to [2, 8]. Conditioned on the stochastic layer below each stochastic layer is specified as a fully factorized gaussian distribution:

$$q_\phi(\mathbf{z}|\mathbf{x}) = q_\phi(\mathbf{z}_1|\mathbf{x}) \prod_{i=2}^{L} q_\phi(\mathbf{z}_i|\mathbf{z}_{i-1}) \tag{7}$$

$$q_\phi(\mathbf{z}_1|\mathbf{x}) = \mathcal{N}\left(\mathbf{z}_1|\mu_{q,1}(\mathbf{x}), \sigma^2_{q,1}(\mathbf{x})\right) \tag{8}$$

$$q_\phi(\mathbf{z}_i|\mathbf{z}_{i-1}) = \mathcal{N}\left(\mathbf{z}_i|\mu_{q,i}(\mathbf{z}_{i-1}), \sigma^2_{q,i}(\mathbf{z}_{i-1})\right), \; i = 2 \dots L. \tag{9}$$

In this parameterization the inference and generative distributions are computed separately with no explicit sharing of information. In the beginning of the training procedure this might cause problems since the inference models have to approximately match the highly variable generative distribution in order to optimize the likelihood. The functions $\mu(\cdot)$ and $\sigma^2(\cdot)$ in the generative and VAE inference models are implemented as:

$$\mathbf{d}(\mathbf{y}) = \texttt{MLP}(\mathbf{y}) \tag{10}$$

$$\mu(\mathbf{y}) = \texttt{Linear}(\mathbf{d}(\mathbf{y})) \tag{11}$$

$$\sigma^2(\mathbf{y}) = \texttt{Softplus}(\texttt{Linear}(\mathbf{d}(\mathbf{y}))) \,, \tag{12}$$

where `MLP` is a two layered multilayer perceptron network, `Linear` is a single linear layer, and `Softplus` applies $\log(1 + \exp(\cdot))$ nonlinearity to each component of its argument vector ensuring positive variances. In our notation, each `MLP`$(\cdot)$ or `Linear`$(\cdot)$ gives a new mapping with its own parameters, so the deterministic variable $\mathbf{d}$ is used to mark that the `MLP`-part is shared between μ and σ^2 whereas the last `Linear` layer is not shared.

2.2 Ladder variational autoencoder inference model

We propose a new inference model that recursively corrects the generative distribution with a data dependent approximate likelihood term. First a deterministic upward pass computes the approximate

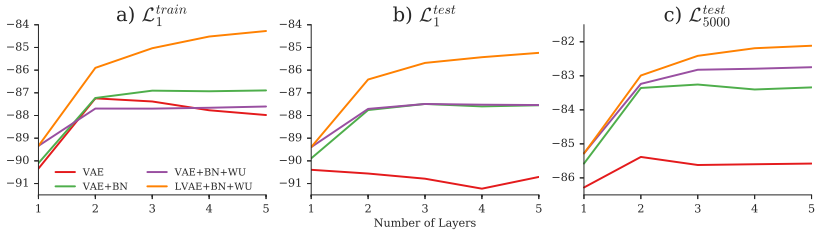


Figure 3: MNIST log-likelihood values for VAEs and the LVAE model with different number of latent layers, Batch normalization (*BN*) and Warm-up (*WU*). a) Train log-likelihood, b) test log-likelihood and c) test log-likelihood with 5000 importance samples.

likelihood contributions:

$$\mathbf{d}_n = \texttt{MLP}(\mathbf{d}_{n-1}) \tag{13}$$

$$\hat{\mu}_{q,i} = \texttt{Linear}(\mathbf{d}_i), i = 1 \dots L \tag{14}$$

$$\hat{\sigma}^2_{q,i} = \texttt{Softplus}(\texttt{Linear}(\mathbf{d}_i)), i = 1 \dots L \tag{15}$$

where $\mathbf{d}_0 = \mathbf{x}$. This is followed by a stochastic downward pass recursively computing both the approximate posterior and generative distributions:

$$q_\phi(\mathbf{z}|\mathbf{x}) = q_\phi(\mathbf{z}_L|\mathbf{x}) \prod_{i=1}^{L-1} q_\phi(\mathbf{z}_i|\mathbf{z}_{i+1}) \tag{16}$$

$$\sigma_{q,i} = \frac{1}{\hat{\sigma}^{-2}_{q,i} + \sigma^{-2}_{p,i}} \tag{17}$$

$$\mu_{q,i} = \frac{\hat{\mu}_{q,i}\hat{\sigma}^{-2}_{q,i} + \mu_{p,i}\sigma^{-2}_{p,i}}{\hat{\sigma}^{-2}_{q,i} + \sigma^{-2}_{p,i}} \tag{18}$$

$$q_\phi(\mathbf{z}_i|\cdot) = \mathcal{N}\left(\mathbf{z}_i|\mu_{q,i}, \sigma^2_{q,i}\right), \tag{19}$$

where $\mu_{q,L} = \hat{\mu}_{q,L}$ and $\sigma^2_{q,L} = \hat{\sigma}^2_{q,L}$. The inference model is a precision-weighted combination of $\hat{\mu}_q$ and $\hat{\sigma}^2_q$ carrying bottom-up information and μ_p and σ^2_p from the generative distribution carrying *top-down* prior information. This parameterization has a probabilistic motivation by viewing $\hat{\mu}_q$ and $\hat{\sigma}^2_q$ as the approximate gaussian likelihood that is combined with a gaussian prior μ_p and σ^2_p from the generative distribution. Together these form the approximate posterior distribution $q_\theta(\mathbf{z}|\mathbf{z}, \mathbf{x})$ using the same top-down dependency structure both in the inference and generative model.

A line of motivation, already noted in [3], is that a purely bottom-up inference process as in i.e. VAEs does not correspond well with real perception, where iterative interaction between bottom-up and top-down signals produces the final activity of a unit[4]. Notably it is difficult for the purely bottom-up inference networks to model the *explaining away* phenomenon, see [22, Chapter 5] for a recent discussion on this phenomenon. The LVAE model provides a framework with the wanted interaction, while not increasing the number of parameters.

2.3 Warm-up from deterministic to variational autoencoder

The variational training criterion in Eq. (5) contains the reconstruction term $p_\theta(\mathbf{x}|\mathbf{z})$ and the variational regularization term. The variational regularization term causes some of the latent units to become inactive during training [13] because the approximate posterior for unit k, $q(z_{i,k}|\dots)$ is regularized towards its own prior $p(z_{i,k}|\dots)$, a phenomenon also recognized in the VAE setting [2, 1]. This can be seen as a virtue of automatic relevance determination, but also as a problem when many units collapse early in training before they learned a useful representation. We observed that such units

[4]The idea was dismissed at the time, since it could introduce substantial theoretical complications.

remain inactive for the rest of the training, presumably trapped in a local minima or saddle point at $KL(q_{i,k}|p_{i,k}) \approx 0$, with the optimization algorithm unable to re-activate them.

We alleviate the problem by initializing training using the reconstruction error only (corresponding to training a standard deterministic auto-encoder), and then gradually introducing the variational regularization term:

$$\mathcal{L}(\theta, \phi; \mathbf{x})_T = -\beta KL(q_\phi(z|x)||p_\theta(\mathbf{z})) + E_{q_\phi(z|x)}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right], \tag{20}$$

where β is increased linearly from 0 to 1 during the first N_t epochs of training. We denote this scheme *warm-up* (abbreviated *WU* in tables and graphs) because the objective goes from having a delta-function solution (corresponding to zero temperature) and then move towards the fully stochastic variational objective. This idea have previously been considered in [15, Section 6.2] and more recently in [1].

3 Experiments

To test our models we use the standard benchmark datasets MNIST, OMNIGLOT [10] and NORB [11]. The largest models trained used a hierarchy of five layers of stochastic latent variables of sizes 64, 32, 16, 8 and 4, going from bottom to top. We implemented all mappings using `MLP`'s with two layers of deterministic hidden units. In all models the `MLP`'s between x and z_1 or d_1 were of size 512. Subsequent layers were connected by `MLP`'s of sizes 256, 128, 64 and 32 for all connections in both the VAE and LVAE. Shallower models were created by removing latent variables from the top of the hierarchy. We sometimes refer to the five layer models as 64-32-16-8-4, the four layer models as 64-32-16-8 and so fourth. The models were trained end-to-end using the Adam [7] optimizer with a mini-batch size of 256. We report the train and test log-likelihood lower bounds, Eq. (5) as well as the approximated true log-likelihood calculated using 5000 importance weighted samples, Eq. (6). The models were implemented using the Theano [19], Lasagne [4] and Parmesan[5] frameworks. The source code is available at [6]

For MNIST, we used a sigmoid output layer to predict the mean of a Bernoulli observation model and leaky rectifiers ($\max(x, 0.1x)$) as nonlinearities in the `MLP`'s. The models were trained for 2000 epochs with a learning rate of 0.001 on the complete training set. Models using warm-up used $N_t = 200$. Similarly to [2], we resample the binarized training values from the real-valued images using a Bernoulli distribution after each epoch which prevents the models from over-fitting. Some of the models were fine-tuned by continuing training for 2000 epochs while multiplying the learning rate with 0.75 after every 200 epochs and increase the number of Monte Carlo and importance weighted samples to 10 to reduce the variance in the approximation of the expectations in Eq. (4) and improve the inference model, respectively.

Models trained on the OMNIGLOT dataset[7], consisting of 28x28 binary images images were trained similar to above except that the number of training epochs was 1500.

Models trained on the NORB dataset[8], consisting of 32x32 grays-scale images with color-coding rescaled to $[0, 1]$, used a Gaussian observation model with mean and variance predicted using a linear and a softplus output layer respectively. The settings were similar to the models above except that: *hyperbolic tangent* was used as nonlinearities in the `MLP`'s and the number of training epochs was 2000.

3.1 Generative log-likelihood performance

In Figure 3 we show the train and test set log-likelihood on MNIST dataset for a series of different models with varying number of stochastic layers.

Consider the $\mathcal{L}_1^{test}$, Figure 3 b), the VAE without batch-normalization and warm-up does not improve for additional stochastic layers beyond one whereas VAEs with batch normalization and warm-up

[5] github.com/casperkaae/parmesan

[6] github.com/casperkaae/LVAE

[7] The OMNIGLOT data was partitioned and preprocessed as in [2], https://github.com/yburda/iwae/tree/master/datasets/OMNIGLOT

[8] The NORB dataset was downloaded in resized format from github.com/gwtaylor/convnet_matlab

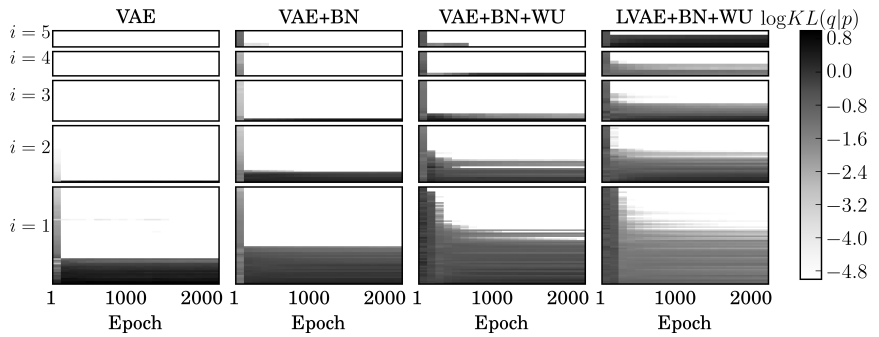


Figure 4: $\log KL(q|p)$ for each latent unit is shown at different training epochs. Low KL (white) corresponds to an inactive unit. The units are sorted for visualization. It is clear that vanilla VAE cannot train the higher latent layers, while introducing batch normalization helps. Warm-up creates more active units early in training, some of which are then gradually pruned away during training, resulting in a more distributed final representation. Lastly, we see that the LVAE activates the highest number of units in each layer.

	$\leq \log p((x))$
VAE 1-layer + NF [17]	-85.10
IWAE, 2-layer + IW=1 [2]	-85.33
IWAE, 2-layer + IW=50 [2]	-82.90
VAE, 2-layer + VGP [20]	-81.90
LVAE, 5-layer	-82.12
LVAE, 5-layer + finetuning	-81.84
LVAE, 5-layer + finetuning + IW=10	-81.74

Table 1: Test set MNIST performance for importance weighted autoencoder (IWAE), VAE with normalizing flows (NF) and VAE with variational gaussian process(VGP). Number of importance weighted (IW) samples used for training is one unless otherwise stated.

improve performance up to three layers. The LVAE models performs better improving performance for each additional layer reaching $\mathcal{L}_1^{test} = -85.23$ with five layers which is significantly higher than the best VAE score at -87.49 using three layers. As expected the improvement in performance is decreasing for each additional layer, but we emphasize that the improvements are consistent even for the addition of the top-most layers. In Figure 3 c) the approximated true log-likelihood estimated using 5000 importance weighted samples is seen. Again the LVAE models performs better than the VAE reaching $\mathcal{L}_{5000}^{test} = -82.12$ compared to the best VAE at -82.74. These results show that the LVAE achieves both a higher approximate log-likelihood score, but also a significantly tighter lower bound on the log-likelihood $\mathcal{L}_1^{test}$. The models in Figure 3 were trained using fixed learning rate and one Monte Carlo (MC) and one importance weighted (IW) sample. To improve performance we fine-tuned the best performing five layer LVAE models by training these for a further 2000 epochs with annealed learning rate and increasing the number of IW samples and see a slight improvements in the test set log-likelihood values, Table 1. We saw no signs of over-fitting for any of our models even though the hierarchical latent representations are highly expressive as seen in Figure 2.

Comparing the results obtained here with current state-of-the art results on permutation invariant MNIST, Table 1, we see that the LVAE performs better than the normalizing flow VAE and importance weighted VAE and comparable to the Variational Gaussian Process VAE. However we note that these results are not directly comparable to these due to differences in the training procedure.

To test the models on more challenging data we used the OMNIGLOT dataset, consisting of characters from 50 different alphabets with 20 samples of each character. The log-likelihood values, Table 2,

	VAE	VAE +BN	VAE +BN +WU	LVAE +BN +WU
OMNIGLOT				
64	-111.21	-105.62	-104.51	$-$
64-32	-110.58	-105.51	-102.61	-102.63
64-32-16	-111.26	-106.09	-102.52	-102.18
64-32-16-8	-111.58	-105.66	-102.66	-102.21
64-32-16-8-4	-110.46	-105.45	-102.48	**-102.11**
NORB				
64	2741	3198	3338	$-$
64-32	2792	3224	3483	3272
64-32-16	2786	3235	3492	**3519**
64-32-16-8	2689	3201	3482	3449
64-32-16-8-4	2654	3198	3422	3455

Table 2: Test set log-likelihood scores for models trained on the OMNIGLOT and NORB datasets. The left most column show dataset and the number of latent variables i each model.

shows similar trends as for MNIST with the LVAE achieving the best performance using five layers of latent variables, see the appendix for further results. The best log-likelihood results obtained here, -102.11, is higher than the best results from [2] at -103.38, which were obtained using more latent variables (100-50 vs 64-32-16-8-4) and further using 50 importance weighted samples for training.

We tested the models using a continuous Gaussian observation model on the NORB dataset consisting of gray-scale images of 5 different toy objects under different illuminations and observation angles. The LVAE achieves a slightly higher score than the VAE, however none of the models see an increase in performance for more using more than three stochastic layers. We found the Gaussian observation models to be harder to optimize compared to the Bernoulli models, a finding also recognized in [23], which might explain the lower utilization of the topmost latent layers in these models.

3.2 Latent representations

The probabilistic generative models studied here automatically tune the model complexity to the data by reducing the effective dimension of the latent representation due to the regularization effect of the priors in Eq. (4). However, as previously identified [15, 2], the latent representation is often overly sparse with few stochastic latent variables propagating useful information.

To study the importance of individual units, we split the variational training criterion $\mathcal{L}$ into a sum of terms corresponding to each unit k in each layer i. For stochastic latent units, this is the KL-divergence between $q(z_{i,k}|\cdot)$ and $p(z_i|z_{i+1})$. Figure 4 shows the evolution of these terms during training. This term is zero if the inference model is collapsed onto the prior carrying no information about the data, making the unit inactive. For the models without warm-up we find that the KL-divergence for each unit is stable during all training epochs with only very few new units activated during training. For the models trained with warm-up we initially see many active units which are then gradually pruned away as the variational regularization term is introduced. At the end of training warm-up results in more active units indicating a more distributed representation and the LVAE model produces both the deepest and most distributed latent representation.

We also study the importance of layers by splitting the training criterion layer-wise as seen in Figure 5. This measures how much of the representation work (or innovation) is done on each layer. The VAEs use the lower layers the most whereas the highest layers are not (or only to a limited degree) used. Contrary to this, the LVAE puts much more importance to the higher layers which shows that it learns both a deeper and qualitatively different hierarchical latent representation which might explain the better performance of the model.

To qualitatively study the learned representations, PCA plots of $\mathbf{z}_i \sim q(\mathbf{z}_i|\cdot)$ are seen in Figure 6. For vanilla VAE, the latent representations above the second layer are completely collapsed on a standard normal prior. Including Batch normalization and warm-up activates one additional layer each in the VAE. The LVAE utilizes all five latent layers and the latent representation shows progressively more

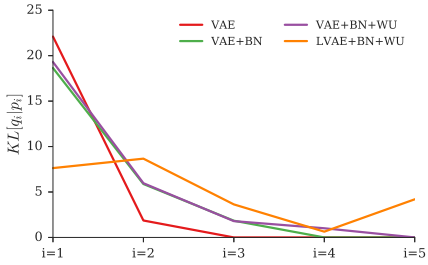


Figure 5: Layer-wise $KL[q|p]$ divergence going from the lowest to the highest layers. In the VAE models the KL divergence is highest in the lowest layers whereas it is more distributed in the LVAE model

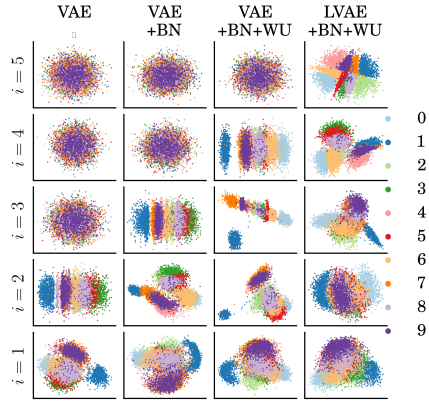


Figure 6: PCA-plots of samples from $q(z_i|z_{i-1})$ for 5-layer VAE and LVAE models trained on MNIST. Color-coded according to true class label

clustering according to class, which is clearly seen in the topmost layer of this model. These findings indicate that the LVAE produce a structured high-level latent representations that are likely useful for semi-supervised learning.

4 Conclusion and Discussion

We presented a new inference model for VAEs combining a bottom-up data-dependent approximate likelihood term with a prior information from the generative distribution. We showed that this parameterization 1) increases the approximated log-likelihood compared to VAEs, 2) provides a tighter bound on the log-likelihood and 3) learns a deeper and qualitatively different latent representation of the data. Secondly we showed that deterministic warm-up and batch-normalization are important for optimizing deep VAEs and LVAEs. Especially the large benefits in generative performance and depth of learned hierarchical representations using batch normalization were surprising given the additional noise introduced. This is something that is not fully understood and deserves further investigation and although batch normalization is not novel we believe that this finding in the context of VAEs are important.

The inference in LVAE is computed recursively by correcting the generative distribution with a data-dependent approximate likelihood contribution. Compared to purely bottom-up inference, this parameterization makes the optimization easier since the inference is simply correcting the generative distribution instead of fitting the two models separately. We believe this explicit parameter sharing between the inference and generative distribution can generally be beneficial in other types of recursive variational distributions such as DRAW [5] where the ideas presented here are directly applicable. Further the LVAE is orthogonal to other methods for improving the inference distribution such as Normalizing flows [17], Variational Gaussian Process [20] or Auxiliary Deep generative models [12] and combining with these might provide further improvements.

Other directions for future work include extending these models to semi-supervised learning which will likely benefit form the learned deep structured hierarchies of latent variables and studying more elaborate inference schemes such as a k-step iterative inference in the LVAE [14].

Acknowledgments

This research was supported by the Novo Nordisk Foundation, Danish Innovation Foundation and the NVIDIA Corporation with the donation of TITAN X and Tesla K40 GPUs.

References

[1] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

[2] Y. Burda, R. Grosse, and R. Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

[3] P. Dayan, G. E. Hinton, R. M. Neal, and R. S. Zemel. The Helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

[4] S. Dieleman, J. Schlüter, C. Raffel, E. Olson, S. K. Sønderby, D. Nouri, A. van den Oord, and E. B. and. Lasagne: First release., Aug. 2015.

[5] K. Gregor, I. Danihelka, A. Graves, and D. Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[6] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[7] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[8] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, 2014.

[9] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[10] B. M. Lake, R. R. Salakhutdinov, and J. Tenenbaum. One-shot learning by inverting a compositional causal process. In *Advances in neural information processing systems*, 2013.

[11] Y. LeCun, F. J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Computer Vision and Pattern Recognition*. IEEE, 2004.

[12] L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther. Auxiliary deep generative models. *Proceedings of the 33nd International Conference on Machine Learning*, 2016.

[13] D. J. MacKay. Local minima, symmetry-breaking, and model pruning in variational free energy minimization. *Inference Group, Cavendish Laboratory, Cambridge, UK*, 2001.

[14] T. Raiko, Y. Li, K. Cho, and Y. Bengio. Iterative neural autoregressive distribution estimator NADE-k. In *Advances in Neural Information Processing Systems*, 2014.

[15] T. Raiko, H. Valpola, M. Harva, and J. Karhunen. Building blocks for variational Bayesian learning of latent variable models. *The Journal of Machine Learning Research*, 8, 2007.

[16] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, 2015.

[17] D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[18] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[19] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[20] D. Tran, R. Ranganath, and D. M. Blei. Variational Gaussian process. *arXiv preprint arXiv:1511.06499*, 2015.

[21] H. Valpola. From neural PCA to deep unsupervised learning. In J. L. E. Bingham, S. Kaski and J. Lampinen, editors, *Advances in Independent Component Analysis and Learning Machines*, chapter 8, pages 143–171. 2015. arXiv preprint arXiv:1411.7783.

[22] G. van den Broeke. What auto-encoders could learn from brains - generation as feedback in unsupervised deep learning and inference, 2016. MSc thesis, Aalto University, Finland.

[23] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

A Additional Results

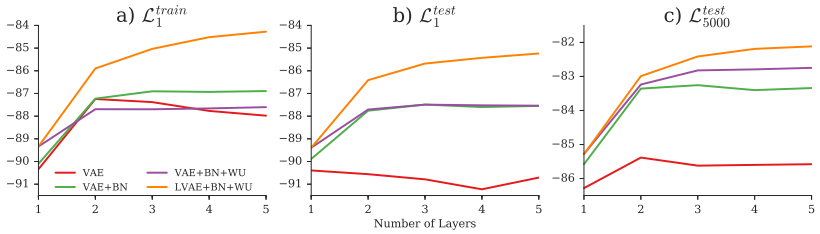


Figure 7: MNIST log-likelihood values for VAEs and the LVAE model with different number of latent layers, Batch normalization (*BN*) and Warm-up (*WU*). a) Train log-likelihood, b) test log-likelihood and c) test log-likelihood with 5000 importance samples. Note that the LVAE without batch normalization performed very poorly why some of the results fall outside the range of the plots

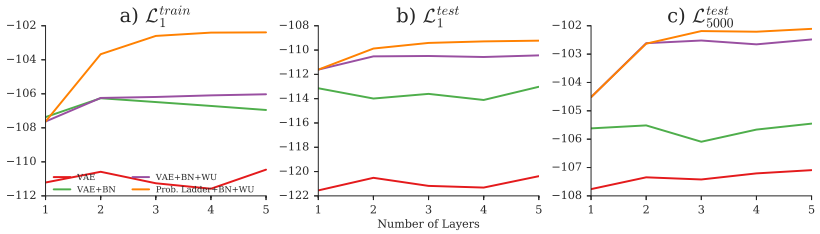


Figure 8: OMNIGLOT log-likelihood values for VAEs and the LVAE model with different number of latent layers, Batch normalization (*BN*) and Warm-up (*WU*). a) Train log-likelihood, b) test log-likelihood and c) test log-likelihood with 5000 importance samples

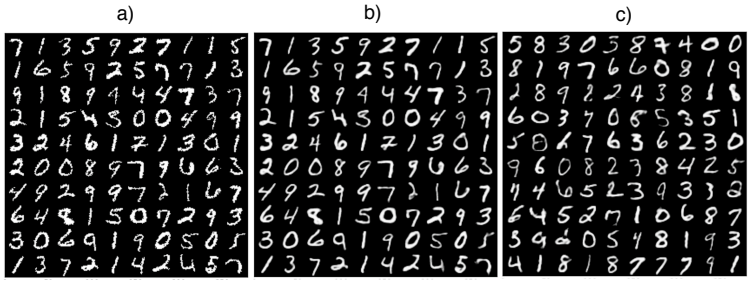


Figure 9: MNIST samples. a) True data, b) Conditional Reconstructions and c) Samples from the prior distribution

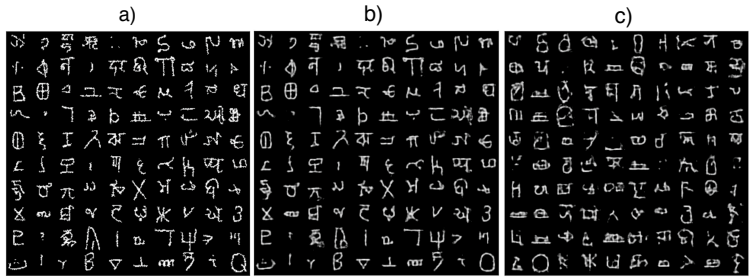


Figure 10: OMNIGLOT samples. a) True data, b) Conditional Reconstructions and c) Samples from the prior distribution

Appendix C

CaGeM: A cluster aware deep generative model

In this appendix we include the arxiv.org version of:

Maaløe, L., Fraccaro, M., Winther, O. (2017). CaGeM: A cluster aware deep generative model. In *Neural Information Processing Systems Workshop* on Approximate Bayesian Inference.

Semi-Supervised Generation with Cluster-aware Generative Models

Lars Maaløe [1] **Marco Fraccaro** [1] **Ole Winther** [1]

Abstract

Deep generative models trained with large amounts of unlabelled data have proven to be powerful within the domain of unsupervised learning. Many real life data sets contain a small amount of labelled data points, that are typically disregarded when training generative models. We propose the *Cluster-aware Generative Model*, that uses unlabelled information to infer a latent representation that models the natural clustering of the data, and additional labelled data points to refine this clustering. The generative performances of the model significantly improve when labelled information is exploited, obtaining a log-likelihood of -79.38 nats on permutation invariant MNIST, while also achieving competitive semi-supervised classification accuracies. The model can also be trained fully unsupervised, and still improve the log-likelihood performance with respect to related methods.

1. Introduction

Variational Auto-Encoders (VAE) (Kingma, 2013; Rezende et al., 2014) and Generative Adversarial Networks (GAN) (Goodfellow et al., 2014) have shown promising generative performances on data from complex high-dimensional distributions. Both approaches have spawn numerous related deep generative models, not only to model data points like those in a large unlabelled training data set, but also for semi-supervised classification (Kingma et al., 2014; Maaløe et al., 2016; Springenberg, 2015; Salimans et al., 2016). In semi-supervised classification a few points in the training data are endowed with class labels, and the plethora of unlabelled data aids to improve a supervised classification model.

Could a few labelled training data points in turn improve a deep generative model? This reverse perspective, doing semi-supervised generation, is investigated in this work. Many of the real life data sets contain a small amount of labelled data, but incorporating this partial knowledge in the generative models is not straightforward, because of the risk of overfitting towards the labelled data. This overfitting can be avoided by finding a good scheme for updating the parameters, like the one introduced in the models for semi-supervised classification (Kingma et al., 2014; Maaløe et al., 2016). However, there is a difference in optimizing the model towards optimal classification accuracy and generative performance. We introduce the *Cluster-aware Generative Model (CaGeM)*, an extension of a VAE, that improves the generative performances, by being able to model the natural clustering in the higher feature representations through a discrete variable (Bengio et al., 2013). The model can be trained fully unsupervised, but its performances can be further improved using labelled class information that helps in constructing well defined clusters. A generative model with added labelled data information may be seen as parallel to how humans rely on abstract domain knowledge in order to efficiently infer a causal model from property induction with very few labelled observations (Tenenbaum et al., 2006).

Supervised deep learning models with no stochastic units are able to learn multiple levels of feature abstraction. In VAEs, however, the addition of more stochastic layers is often accompanied with a built-in pruning effect so that the higher layers become disconnected and therefore not exploited by the model (Burda et al., 2015a; Sønderby et al., 2016). As we will see, in CaGeM the possibility of learning a representation in the higher stochastic layers that can model clusters in the data drastically reduces this issue. This results in a model that is able to disentangle some of the factors of variation in the data and that extracts a hierarchy of features beneficial during the generation phase. By using only 100 labelled data points, we present state of the art log-likelihood performance on permutation-invariant models for MNIST, and an improvement with respect to comparable models on the OMNIGLOT data set. While the main focus of this paper is semi-supervised generation, we also show that the same model is able to achieve competitive semi-supervised classification results.

[1]Technical University of Denmark. Correspondence to: Lars Maaløe <larsma@dtu.dk>, Marco Fraccaro <marfra@dtu.dk>, Ole Winther <olwi@dtu.dk>.

2. Variational Auto-encoders

A *Variational Auto-Encoder (VAE)* (Kingma, 2013; Rezende et al., 2014) defines a deep generative model for data x that depends on latent variable z or a hierarchy of latent variables, e.g. $z = [z_1, z_2]$, see Figure 1a for a graphical representation. The joint distribution of the two-level generative model is given by

$$p_\theta(x, z_1, z_2) = p_\theta(x|z_1)p_\theta(z_1|z_2)p(z_2) \ ,$$

where

$$\begin{aligned} p_\theta(z_1|z_2) &= \mathcal{N}(z_1; \mu^1_\theta(z_2), \sigma^1_\theta(z_2)) \\ p(z_2) &= \mathcal{N}(z_2; 0, I) \end{aligned}$$

are Gaussian distributions with a diagonal covariance matrix and $p_\theta(x|z_1)$ is typically a parameterized Gaussian (continuous data) or Bernoulli distribution (binary data). The probability distributions of the generative model of a VAE are parameterized using deep neural networks whose parameters are denoted by θ. Training is performed by optimizing the *Evidence Lower Bound (ELBO)*, a lower bound to the intractable log-likelihood $\log p_\theta(x)$ obtained using Jensen's inequality:

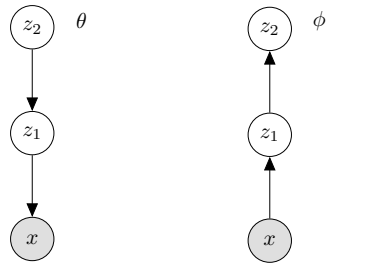
$$\begin{aligned} \log p_\theta(x) &= \log \iint p_\theta(x, z_1, z_2) \mathrm{d}z_1 \mathrm{d}z_2 \\ &\geq \mathbb{E}_{q_\phi(z_1,z_2|x)} \left[\log \frac{p_\theta(x, z_1, z_2)}{q_\phi(z_1, z_2|x)} \right] = \mathcal{F}(\theta, \phi) \ . \end{aligned} \tag{1}$$

The introduced variational distribution $q_\phi(z_1, z_2|x)$ is an approximation to the model's posterior distribution $p_\theta(z_1, z_2|x)$, defined with a bottom-up dependency structure where each variable of the model depends on the variable below in the hierarchy:

$$\begin{aligned} q_\phi(z_1, z_2|x) &= q_\phi(z_1|x)q_\phi(z_2|z_1) \\ q_\phi(z_1|x) &= \mathcal{N}(z_1; \mu^1_\phi(x), \sigma^1_\phi(x)) \\ q_\phi(z_2|z_1) &= \mathcal{N}(z_2; \mu^2_\phi(z_1), \sigma^2_\phi(z_1)) \ . \end{aligned}$$

Similar to the generative model, the mean and diagonal covariance of both Gaussian distributions defining the inference network q_ϕ are parameterized with deep neural networks that depend on parameters ϕ, see Figure 1b for a graphical representation.

We can learn the parameters θ and ϕ by jointly maximizing the ELBO $\mathcal{F}(\theta, \phi)$ in (1) with stochastic gradient ascent, using Monte Carlo integration to approximate the intractable expectations and computing low variance gradients with the reparameterization trick (Kingma, 2013; Rezende et al., 2014).



(a) Generative model p_θ (b) Inference model q_ϕ

Figure 1: Generative model and inference model of a Variational Auto-Encoder with two stochastic layers.

Inactive stochastic units A common problem encountered when training VAEs with bottom-up inference networks is given by the so called *inactive units* in the higher layers of stochastic variables (Burda et al., 2015a; Sønderby et al., 2016). In a 2-layer model for example, VAEs often learn $q_\phi(z_2|z_1) = p(z_2) = \mathcal{N}(z_2; 0, I)$, i.e. the variational approximation of z_2 uses no information coming from the data point x through z_1. If we rewrite the ELBO in (1) as

$$\begin{aligned} \mathcal{F}(\theta, \phi) = \mathbb{E}_{q_\phi(z_1,z_2|x)} \left[\log \frac{p_\theta(x, z_1|z_2)}{q_\phi(z_1|x)} \right] - \\ \mathbb{E}_{q_\phi(z_1|x)} \left[KL \left[q_\phi(z_2|z_1) || p(z_2) \right] \right] \end{aligned}$$

we can see that $q_\phi(z_2|z_1) = p(z_2)$ represents a local maxima of our optimization problem where the KL-divergence term is set to zero and the information flows by first sampling in $\widetilde{z_1} \sim q_\phi(z_1|x)$ and then computing $p_\theta(x|\widetilde{z_1})$ (and is therefore independent from z_2). Several techniques have been developed in the literature to mitigate the problem of inactive units, among which we find annealing of the KL term (Bowman et al., 2015; Sønderby et al., 2016) or the use of free bits (Kingma et al., 2016).

Using ideas from Chen et al. (2017), we notice that the inactive units in a VAE with 2 layers of stochastic units can be justified not only as a poor local maxima, but also from the modelling point of view. Chen et al. (2017) give a *bits-back coding* interpretation of Variational Inference for a generative model of the form $p(x, z) = p(x|z)p(z)$, with data x and stochastic units z. The paper shows that if the decoder $p(x|z)$ is powerful enough to explain most of the structure in the data (e.g. an autoregressive decoder), then it will be convenient for the model to set $q(z|x) = p(z)$ not to incur in an extra optimization cost of $KL[q(z|x)||p(z|x)]$. The inactive z_2 units in a 2-layer VAE can therefore be seen as caused by the flexible distribution $p_\theta(x, z_1|z_2)$ that is able to explain most of the structure in the data without using information from z_2. By making $q_\phi(z_2|z_1) = p(z_2)$, the

model can avoid the extra cost of $KL\left[q_\phi(z_2|x)||p_\theta(z_2|x)\right]$. A more detailed discussion on the topic can be found in Appendix A.

It is now clear that if we want a VAE to exploit the power of additional stochastic layers we need to define it so that the benefits of encoding meaningful information in z_2 is greater than the cost $KL\left[q_\phi(z_2|x)||p_\theta(z_2|x)\right]$ that the model has to pay. As we will discuss below, we will achieve this by aiding the generative model to do representation learning.

3. Cluster-aware Generative Models

Hierarchical models parameterized by deep neural networks have the ability to represent very flexible distributions. However, in the previous section we have seen that the units in the higher stochastic layers of a VAE often become inactive. We will show that we can help the model to exploit the higher stochastic layers by explicitly encoding a useful representation, i.e. the ability to model the natural clustering of the data (Bengio et al., 2013), which will also be needed for semi-supervised generation.

We favor the flow of higher-level global information through z_2 by extending the generative model of a VAE with a discrete variable y representing the choice of one out of K different clusters in the data. The joint distribution $p_\theta(x, z_1, z_2)$ is computed by marginalizing over y:

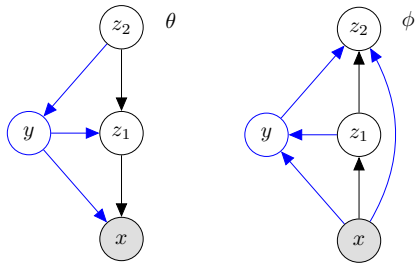
$$\begin{aligned} p_\theta(x, z_1, z_2) &= \sum_y p_\theta(x, y, z_1, z_2) \\ &= \sum_y p_\theta(x|y, z_1) p_\theta(z_1|y, z_2) p_\theta(y|z_2) p(z_2) \ . \end{aligned}$$

We call this model *Cluster-aware Generative Model (CaGeM)*, see Figure 2 for a graphical representation. The introduced categorical distribution $p_\theta(y|z_2) = \text{Cat}(y; \pi_\theta(z_2))$ (π_θ represents the class distribution) depends solely on z_2, that needs therefore to stay active for the model to be able to represent clusters in the data. We further add the dependence of z_1 and x on y, so that they can now both also represent cluster-dependent information.

3.1. Inference

As done for the VAE in (1), we can derive the ELBO for CaGeM by maximizing the log-likelihood

$$\begin{aligned} \log p_\theta(x) &= \log \iint p_\theta(x, z_1, z_2) \mathrm{d}z_1 \mathrm{d}z_2 \\ &= \log \iint \sum_y p_\theta(x, y, z_1, z_2) \mathrm{d}z_1 \mathrm{d}z_2 \\ &\geq \mathbb{E}_{q_\phi(y, z_1, z_2|x)} \left[\log \frac{p_\theta(x, y, z_1, z_2)}{q_\phi(y, z_1, z_2|x)} \right] \ . \end{aligned}$$



(a) Generative model p_θ (b) Inference model q_ϕ

Figure 2: Generative model and inference model of a CaGeM with two stochastic layers (black and blue lines). The black lines only represent a standard VAE.

We define the variational approximation $q_\phi(y, z_1, z_2|x)$ over the latent variables of the model as

$$q_\phi(y, z_1, z_2|x) = q_\phi(z_2|x, y, z_1) q_\phi(y|z_1, x) q_\phi(z_1|x) \ ,$$

where

$$\begin{aligned} q_\phi(z_1|x) &= \mathcal{N}(z_1; \mu^1_\phi(x), \sigma^1_\phi(x)) \\ q_\phi(z_2|x, y, z_1) &= \mathcal{N}(z_2; \mu^2_\phi(y, z_1), \sigma^2_\phi(y, z_1)) \\ q_\phi(y|z_1, x) &= \text{Cat}(y; \pi_\phi(z_1, x)) \end{aligned}$$

In the inference network we then reverse all the dependencies among random variables in the generative model (the arrows in the graphical model in Figure 2). This results in a *bottom-up* inference network that performs a feature extraction that is fundamental for learning a good representation of the data. Starting from the data x we construct higher levels of abstraction, first through the variables z_1 and y, and finally through the variable z_2, that includes the global information used in the generative model. In order to make the higher representation more expressive we add a skip-connection from x to z_2, that is however not fundamental to improve the performances of the model.

With this factorization of the variational distribution $q_\phi(y, z_1, z_2|x)$, the ELBO can be written as

$$\begin{aligned} \mathcal{F}(\theta, \phi) = \mathbb{E}_{q_\phi(z_1|x)} \Bigg[\sum_y q_\phi(y|z_1, x) \cdot \\ \cdot \mathbb{E}_{q_\phi(z_2|x, y, z_1)} \left[\log \frac{p_\theta(x, y, z_1, z_2)}{q_\phi(y, z_1, z_2|x)} \right] \Bigg] \ . \end{aligned}$$

We maximize $\mathcal{F}(\theta, \phi)$ by jointly updating, with stochastic gradient ascent, the parameters θ of the generative model and ϕ of the variational approximation. When computing the gradients, the summation over y is performed analytically, whereas the intractable expectations over z_1 and z_2

are approximated by sampling. We use the reparameterization trick to reduce the variance of the stochastic gradients.

4. Semi-Supervised Generation with CaGeM

In some applications we may have class label information for some of the data points in the training set. In the following we will show that CaGeM provides a natural way to exploit additional labelled data to improve the performance of the generative model. Notice that this *semi-supervised generation* approach differs from the more traditional *semi-supervised classification* task that uses unlabelled data to improve classification accuracies (Kingma et al., 2014; Maaløe et al., 2016; Salimans et al., 2016). In our case in fact, it is the labelled data that supports the generative task. Nevertheless, we will see in our experiment that CaGeM also leads to competitive semi-supervised classification performances.

To exploit the class information, we first set the number of clusters K equal to the number of classes C. We can now define two classifiers in CaGeM:

1. In the inference network we can compute the class probabilities given the data, i.e. $q_\phi(y|x)$, by integrating out the stochastic variables z_1 from $q_\phi(y, z_1|x)$

$$\begin{aligned} q_\phi(y|x) &= \int q_\phi(y, z_1|x)\mathrm{d}z_1 \\ &= \int q_\phi(y|z_1, x)q_\phi(z_1|x)\mathrm{d}z_1 \end{aligned}$$

2. Another set of class-probabilities can be computed using the generative model. Given the posterior distribution $p_\theta(z_2|x)$ we have in fact

$$p_\theta(y|x) = \int p_\theta(y|z_2)p_\theta(z_2|x)\mathrm{d}z_2 \,.$$

The posterior over z_2 is intractable, but we can approximate it using the variational approximation $q_\phi(z_2|x)$, that is obtained by marginalizing out y and the variable z_1 in the joint distribution $q_\phi(y, z_1, z_2|x)$:

$$\begin{aligned} p_\theta(y|x) &\approx \int p_\theta(y|z_2)q_\phi(z_2|x)\mathrm{d}z_2 \\ &= \int p_\theta(y|z_2)\left(\int \sum_{\widetilde{y}} q_\phi(z_2|x, \widetilde{y}, z_1)\cdot \right. \\ &\qquad \left. \cdot\, q_\phi(\widetilde{y}|z_1, x)q_\phi(z_1|x)\mathrm{d}z_1\right)\mathrm{d}z_2 \,. \end{aligned}$$

While for the labels $\widetilde{y}$ the summation can be carried out analytically, for the variable z_1 and z_2 we use Monte Carlo integration. For each of the C classes we will then obtain a different z_2^c sample $(c = 1, \dots C)$ with a corresponding weight given by $q_\phi(\widetilde{y}^c|z_1, x)$. This therefore resembles a *cascade* of classifiers, as the class probabilities of the $p_\theta(y|x)$ classifier will depend on the probabilities of the classifier $q_\phi(y|z_1, x)$ in the inference model.

As our main goal is to learn representations that will lead to good generative performance, we interpret the classification of the additional labelled data as a secondary task that aids in learning a z_2 feature space that can be easily separated into clusters. We can then see this as a form of *semi-supervised clustering* (Basu et al., 2002), where we know that some data points belong to the same cluster and we are free to learn a data manifold that makes this possible.

The optimal features for the classification task could be very different from the representations learned for the generative task. This is why it is important not to update the parameters of the distributions over z_1, z_2 and x, in both generative model and inference model, using labelled data information. If this is not done carefully, the model could be prone to overfitting towards the labelled data. We define as θ_y the subset of θ containing the parameters in $p_\theta(y|z_2)$, and as ϕ_y the subset of ϕ containing the parameters in $q_\phi(y|z_1, x)$. θ_y and ϕ_y then represent the incoming arrows to y in Figure 2. We update the parameters θ and ϕ jointly by maximizing the new objective

$$\mathcal{I} = \sum_{\{x_u\}} \mathcal{F}(\theta, \phi) - \alpha \left(\sum_{\{x_l, y_l\}} (\mathcal{H}_p(\theta_y, \phi_y) + \mathcal{H}_q(\phi_y)) \right)$$

where $\{x_u\}$ is the set of unlabelled training points, $\{x_l, y_l\}$ is the set of labelled ones, and $\mathcal{H}_p$ and $\mathcal{H}_q$ are the standard categorical cross-entropies for the $p_\theta(y|x)$ and $q_\phi(y|x)$ classifiers respectively. Notice that we consider the cross-entropies only a function of θ_y and ϕ_y, meaning that the gradients of the cross-entropies with respect to the parameters of the distributions over z_1, z_2 and x will be 0, and will not depend on the labelled data (as needed when learning meaningful representations of the data to be used for the generative task). To match the relative magnitudes between the ELBO $\mathcal{F}(\theta, \phi)$ and the two cross-entropies we set $\alpha = \beta \frac{N_u + N_l}{N_l}$ as done in (Kingma et al., 2014; Maaløe et al., 2016), where N_u and N_l are the numbers of unlabelled and labelled data points, and β is a scaling constant.

5. Experiments

We evaluate CaGeM by computing the generative log-likelihood performance on MNIST and OMNIGLOT (Lake et al., 2013) datasets. The model is parameterized by feed-forward neural networks (`NN`) and linear layers (`Linear`),

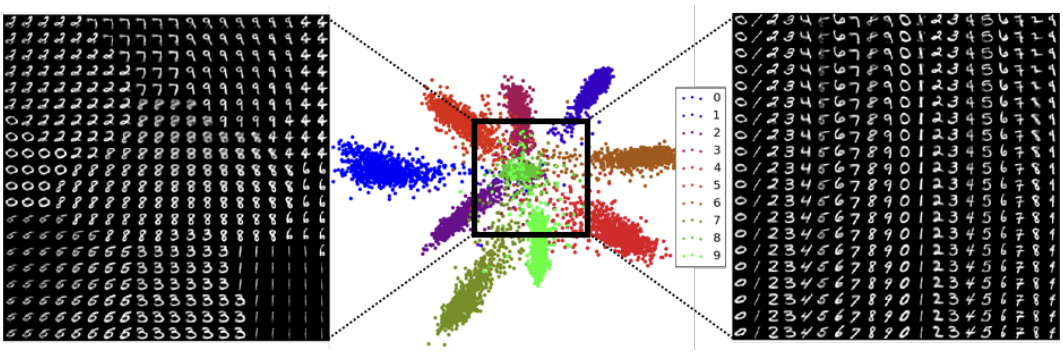


Figure 3: Visualizations from CaGeM-100 with a 2-dimensional z_2 space. The middle plot shows the latent space, from which we generate random samples (left) and class conditional random samples (right) with a mesh grid (black bounding box). The relative placement of the samples in the scatter plot corresponds to a digit in the mesh grid.

so that, for Gaussian outputs, each collection of incoming edges to a node in Figure 2 is defined as:

$$d = \texttt{NN}(x) \qquad \mu = \texttt{Linear}(d) \qquad \log \sigma = \texttt{Linear}(d)\ .$$

For Bernoulli distributed outputs we simply define a feed-forward neural network with a sigmoid activation function for the output. Between dense layers we use the rectified linear unit as non-linearity and batch-normalization (Ioffe & Szegedy, 2015). We only collect statistics for the batch-normalization during unlabelled inference. For the log-likelihood experiments we apply temperature on the KL-terms during the first 100 epochs of training (Bowman et al., 2015; Sønderby et al., 2016). The stochastic layers are defined with $\texttt{dim}(z_1) = 64$, $\texttt{dim}(z_2) = 32$ and 2-layered neural feed-forward networks with respectively 1024 and 512 units in each layer. Training is performed using the Adam optimizer (Kingma & Ba, 2014) with an initial learning rate of 0.001 and annealing it by $.75$ every 50 epochs. The experiments are implemented with Theano (Bastien et al., 2012), Lasagne (Dieleman et al., 2015) and Parmesan[1].

For both datasets we report unsupervised and semi-supervised permutation invariant log-likelihood performance and for MNIST we also report semi-supervised classification errors. The input data is dynamically binarized and the ELBO is evaluated by taking 5000 importance-weighted (IW) samples, denoted $\mathcal{F}_{5000}$. We evaluate the performance of CaGeM with different numbers of labelled samples referred to as CaGeM-#$\texttt{labels}$. When used, the labelled data is randomly sampled evenly across the class distribution. All experiments across datasets are run with the same architecture.

[1]A variational repository named parmesan on Github.

6. Results

Table 1 shows the generative log-likelihood performances of different variants of CaGeM on the MNIST data set. We can see that the more labelled samples we use, the better the generative performance will be. Even though the results are not directly comparable, since CaGeM exploits a small fraction supervised information, we find that using only 100 labelled samples (10 samples per class), CaGeM-100 model achieves state of the art log-likelihood performance on permutation invariant MNIST with a simple 2-layered model. We also trained a ADGM-100 from Maaløe et al. (2016)[2] in order to make a fair comparison on generative log-likelihood in a semi-supervised setting and reached a performance of -86.06 nats. This indicates that models that are highly optimized for improving semi-supervised classification accuracy may be a suboptimal choice for generative modeling.

CaGeM could further benefit from the usage of non-permutation invariant architectures suited for image data, such as the autoregressive decoders used by IAF VAE (Kingma et al., 2016) and VLAE (Chen et al., 2017). The fully unsupervised CaGeM-0 results show that by defining clusters in the higher stochastic units, we achieve better performances than the closely related IWAE (Burda et al., 2015a) and LVAE (Sønderby et al., 2016) models. It is finally interesting to see from Table 1 that CaGeM-0 performs well even when the number of clusters are different from the number of classes in the labelled data set.

In Figure 4 we show in detail how the performance of CaGeM increases as we add more labelled data points. We can also see that the ELBO $\mathcal{F}_1^{test}$ tightens when

[2]We used the code supplied in the repository named auxiliary-deep-generative-models on Github.

	$\leq \log p(x)$
NON-PERMUTATION INVARIANT	
DRAW+VGP (TRAN ET AL., 2016)	-79.88
IAF VAE (KINGMA ET AL., 2016)	-79.10
VLAE (CHEN ET AL., 2017)	-78.53
PERMUTATION INVARIANT	
AVAE, L=2, IW=1 (MAALØE ET AL., 2016)	-82.97
IWAE, L=2, IW=50 (BURDA ET AL., 2015A)	-82.90
LVAE, L=5, IW=10 (SØNDERBY ET AL., 2016)	-81.74
VAE+VGP, L=2 (TRAN ET AL., 2016)	-81.32
DVAE (ROLFE, 2017)	-80.04
CAGEM-0, L=2, IW=1, K=20	-82.18
CAGEM-0, L=2, IW=1, K=10	-81.60
CAGEM-20, L=2, IW=1	-81.47
CAGEM-50, L=2, IW=1	-80.49
CAGEM-100, L=2, IW=1	-79.38

Table 1: Test log-likelihood for permutation invariant and non-permutation invariant MNIST. L, IW and K denotes the number of stochastic layers (if it is translatable to the VAE), the number of importance weighted samples used during inference, and the number of predefined clusters used.

adding more labelled information, as compared to $\mathcal{F}_1^{\text{LVAE}} = -85.23$ and $\mathcal{F}_1^{\text{VAE}} = -87.49$ (Sønderby et al., 2016).

The PCA plots of the z_2 variable of a VAE, CaGeM-0 and CaGeM-100 are shown in Figure 5 . We see how CaGeMs encode clustered information into the higher stochastic layer. Since CaGeM-0 is unsupervised, it forms less class-dependent clusters compared to the semi-supervised CaGeM-100, that fits its z_2 latent space into 10 nicely separated clusters. Regardless of the labelled information added during inference, CaGeM manages to activate a high amount of units, as for CaGeM we obtain $KL[q_\phi(z_2|x,y)||p(z_2)] \approx 17$ nats, while a LVAE with 2 stochastic layers obtains ≈ 9 nats.

The generative model in CaGeM enables both random samples, by sampling the class variable $y \sim p_\theta(y|z_2)$ and feeding it to $p_\theta(x|z_1, y)$, and class conditional samples by fixing y. Figure 3 shows the generation of MNIST digits from CaGeM-100 with $\texttt{dim}(z_2) = 2$. The images are generated by applying a linearly spaced mesh grid within the latent space z_2 and performing random generations (left) and conditional generations (right). When generating samples in CaGeM, it is clear how the latent units z_1 and z_2 capture different modalities within the true data distribution, namely style and class.

Regardless of the fact that CaGeM was designed to optimize the semi-supervised generation task, the model can also be used for classification by using the classifier $p_\theta(y|x)$. In Table 2 we show that the semi-supervised classification accuracies obtained with CaGeM are comparable

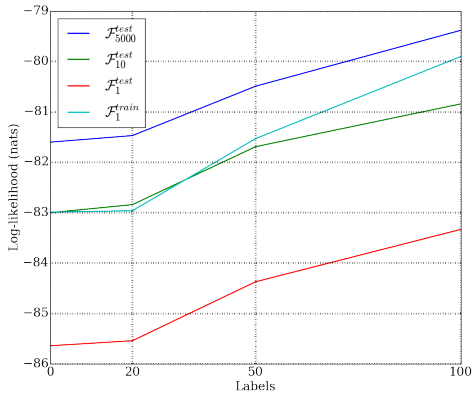


Figure 4: Log-likelihood scores for CaGeM on MNIST with 0, 20, 50 and 100 labels with 1, 10 and 5000 IW samples.

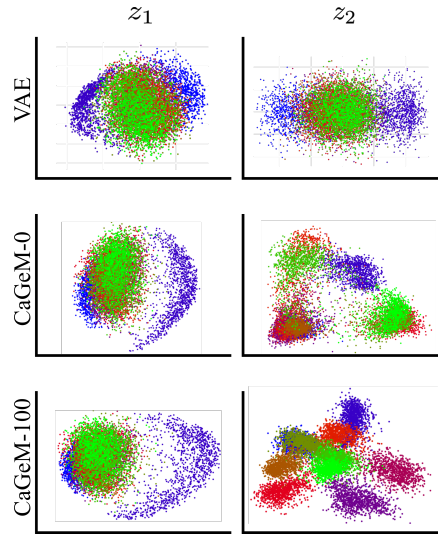


Figure 5: PCA plots of the stochastic units z_1 and z_2 in a 2-layered model trained on MNIST. The colors corresponds to the true labels.

to the performance of GANs (Salimans et al., 2016).

The OMNIGLOT dataset consists of 50 different alphabets of handwritten characters, where each character is sparsely represented. In this task we use the alphabets as the cluster information, so that the z_2 representation should divide correspondingly. From Table 3 we see an improvement

LABELS	20	50	100
M1+M2 (KINGMA ET AL., 2014)	-	-	3.33% (±0.14)
VAT (MIYATO ET AL., 2015)	-	-	2.12%
CATGAN (SPRINGENBERG, 2015)	-	-	1.91% (±0.1)
SDGM (MAALØE ET AL., 2016)	-	-	1.32% (±0.07)
LADDER NETWORK (RASMUS ET AL., 2015)	-	-	1.06% (±0.37)
ADGM (MAALØE ET AL., 2016)	-	-	0.96% (±0.02)
IMP. GAN (SALIMANS ET AL., 2016)	16.77% (±4.52)	2.21% (±1.36)	0.93% (±0.65)
CAGEM	15.86%	2.42%	1.16%

Table 2: Semi-supervised test error % benchmarks on MNIST for 20, 50, and 100 randomly chosen and evenly distributed labelled samples. Each experiment was run 3 times with different labelled subsets and the reported accuracy is the mean value.

over other comparable VAE architectures (VAE, IWAE and LVAE), however, the performance is far from the once reported from the auto-regressive models (Kingma et al., 2016; Chen et al., 2017). This indicates that the alphabet information is not as strong as for a dataset like MNIST. This is also indicated from the accuracy of CaGeM-500, reaching a performance of $\approx 24\%$. Samples from the model can be found in Figure 6.

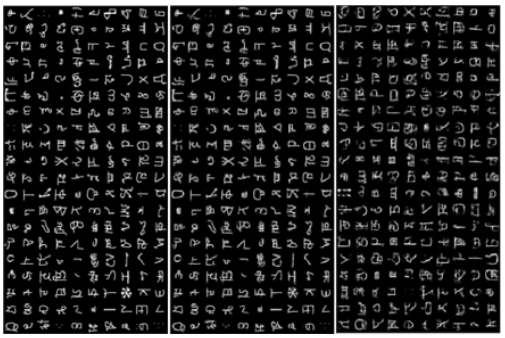


Figure 6: Generations from CaGeM-500. (left) The input images, (middle) the reconstructions, and (right) random samples from z_2.

	$\leq \log p(x)$
VAE, L=2, IW=50 (BURDA ET AL., 2015A)	−106.30
IWAE, L=2, IW=50 (BURDA ET AL., 2015A)	−103.38
LVAE, L=5, FT, IW=10 (SØNDERBY ET AL., 2016)	−102.11
RBM (BURDA ET AL., 2015B)	−100.46
DBN (BURDA ET AL., 2015B)	−100.45
DVAE (ROLFE, 2017)	−97.43
CAGEM-500, L=2, IW=1	−100.86

Table 3: Generative test log-likelihood for permutation invariant OMNIGLOT.

7. Discussion

As we have seen from our experiments, CaGeM offers a way to exploit the added flexibility of a second layer of stochastic units, that stays active as the modeling performances can greatly benefit from capturing the natural clustering of the data. Other recent works have presented alternative methods to mitigate the problem of inactive units when training flexible models defined by a hierarchy of stochastic layers. Burda et al. (2015a) used importance samples to improve the tightness of the ELBO, and showed that this new training objective helped in activating the units of a 2-layer VAE. Sønderby et al. (2016) trained Ladder Variational Autoencoders (LVAE) composed of up to 5 layers of stochastic units, using a top-down inference network that forces the information to flow in the higher stochastic layers. Contrarily to the bottom-up inference network of CaGeM, the top-down approach used in LVAEs does not enforce a clear separation between the role of each stochastic unit, as proven by the fact that all of them encode some class information. Longer hierarchies of stochastic units unrolled in time can be found in the sequential setting (Krishnan et al., 2015; Fraccaro et al., 2016). In these applications the problem of inactive stochastic units appears when using powerful autoregressive decoders (Fraccaro et al., 2016; Chen et al., 2017), but is mitigated by the fact that new data information enters the model at each time step.

The discrete variable y of CaGeM was introduced to be able to define a better learnable representation of the data, that helps in activating the higher stochastic layer. The combination of discrete and continuous variables for deep generative models was also recently explored by several authors. Maddison et al. (2016); Jang et al. (2016) used a continuous relaxation of the discrete variables, that makes it possible to efficiently train the model using stochastic backpropagation. The introduced Gumbel-Softmax variables allow to sacrifice log-likelihood performances to avoid the computationally expensive integration over y. Rolfe (2017) presents a new class of probabilistic models that combines an undirected component consisting of a bipartite Boltzmann machine with binary units and a directed component with multiple layers of continuous variables.

Traditionally, semi-supervised learning applications of deep generative models such as Variational Auto-encoders and Generative Adversarial Networks (Goodfellow et al., 2014) have shown that, whenever only a small fraction of labelled data is available, the supervised classification task can benefit from additional unlabelled data (Kingma et al., 2014; Maaløe et al., 2016; Salimans et al., 2016). In this work we consider the semi-supervised problem from a different perspective, and show that the generative task of CaGeM can benefit from additional labelled data. As a by-product of our model however, we also obtain competitive semi-supervised classification results, meaning that CaGeM is able to share statistical strength between the generative and classification tasks.

When modeling natural images, the performance of CaGeM could be further improved using more powerful autoregressive decoders such as the ones in (Gulrajani et al., 2016; Chen et al., 2017). Also, an even more flexible variational approximation could be obtained using auxiliary variables (Ranganath et al., 2015; Maaløe et al., 2016) or normalizing flows (Rezende & Mohamed, 2015; Kingma et al., 2016).

8. Conclusion

In this work we have shown how to perform semi-supervised generation with CaGeM. We showed that CaGeM improves the generative log-likelihood performance over similar deep generative approaches by creating clusters for the data in its higher latent representations using unlabelled information. CaGeM also provides a natural way to refine the clusters using additional labelled information to further improve its modelling power.

A. The Problem of Inactive Units

First consider a model $p(x)$ without latent units. We consider the asymptotic average properties, so we take the expectation of the log-likelihood over the (unknown) data distribution $p_{\text{data}}(x)$:

$$\begin{aligned}\mathbb{E}_{p_{\text{data}}(x)}\left[\log p(x)\right] &= \mathbb{E}_{p_{\text{data}}(x)}\left[\log\left(p_{\text{data}}(x)\frac{p(x)}{p_{\text{data}}(x)}\right)\right] \\ &= -\mathcal{H}(p_{\text{data}}) - KL(p_{\text{data}}(x)||p(x)) \,,\end{aligned}$$

where $\mathcal{H}(p) = -\mathbb{E}_{p(x)}\left[\log p(x)\right]$ is the entropy of the distribution and $KL(\cdot||\cdot)$ is the KL-divergence. The expected log-likelihood is then simply the baseline entropy of the data generating distribution minus the deviation between the data generating distribution and our model for the distribution.

For the latent variable model $p_{\text{lat}}(x) = \int p(x|z)p(z)dz$ the log-likelihood bound is:

$$\log p_{\text{lat}}(x) \geq \mathbb{E}_{q(z|x)}\left[\log \frac{p(x|z)p(z)}{q(z|x)}\right] \,.$$

We take the expectation over the data generating distribution and apply the same steps as above

$$\begin{aligned}\mathbb{E}_{p_{\text{data}}(x)}\left[\log p_{\text{lat}}(x)\right] &\geq \mathbb{E}_{p_{\text{data}}(x)}\mathbb{E}_{q(z|x)}\left[\log \frac{p(x|z)p(z)}{q(z|x)}\right] \\ &= -\mathcal{H}(p_{\text{data}}) - KL(p_{\text{data}}(x)||p_{\text{lat}}(x)) \\ &\quad - \mathbb{E}_{p_{\text{data}}(x)}\left[KL(q(z|x)||p(z|x))\right] \,,\end{aligned}$$

where $p(z|x) = p(x|z)p(z)/p_{\text{lat}}(x)$ is the (intractable) posterior of the latent variable model. This results shows that we pay an additional price (the last term) for using an approximation to the posterior.

The latent variable model can choose to ignore the latent variables, $p(x|z) = \hat{p}(x)$. When this happens the expression falls back to the log-likelihood without latent variables. We can therefore get an (intractable) condition for when it is advantageous for the model to use the latent variables:

$$\begin{aligned}\mathbb{E}_{p_{\text{data}}(x)}\left[\log p_{\text{lat}}(x)\right] > \mathbb{E}_{p_{\text{data}}(x)}\left[\log \hat{p}(x)\right] + \\ \mathbb{E}_{p_{\text{data}}(x)}\left[KL(q(z|x)||p(z|x))\right] \,.\end{aligned}$$

The model will use latent variables when the log-likelihood gain is so high that it can compensate for the loss $KL(q(z|x)||p_{\text{lat}}(z|x))$ we pay by using an approximate posterior distribution.

The above argument can also be used to understand why it is harder to get additional layers of latent variables to become active. For a two-layer latent variable model $p(x, z_1, z_2) = p(x|z_1)p(z_1|z_2)p(z_2)$ we use a variational distribution $q(z_1, z_2|x) = q(z_2|z_1, x)q(z_1|x)$ and decompose the log likelihood bound using $p(x, z_1, z_2) = p(z_2|z_1, x)p(z_1|x)p_{\text{lat},2}(x)$:

$$\begin{aligned}&\mathbb{E}_{p_{\text{data}}(x)}\left[\log p_{\text{lat},2}(x)\right] \\ &\quad\geq \mathbb{E}_{p_{\text{data}}(x)}\mathbb{E}_{q(z_1,z_2|x)}\left[\log \frac{p(x|z_1)p(z_1|z_2)p(z_2)}{q(z_1, z_2|x)}\right] \\ &\quad= -H(p_{\text{data}}) - KL(p_{\text{data}}(x)||p_{\text{lat},2}(x)) \\ &\qquad- \mathbb{E}_{p_{\text{data}}(x)}\mathbb{E}_{q(z_1|x)}\left[KL(q(z_2|z_1, x)||p(z_2|z_1, x))\right] \\ &\qquad- \mathbb{E}_{p_{\text{data}}(x)}KL(q(z_1|x)||p(z_1|x)) \,.\end{aligned}$$

Again this expression falls back to the one-layer model when $p(z_1|z_2) = \hat{p}(z_1)$. So whether to use the second layer of stochastic units will depend upon the potential diminishing return in terms of log likelihood relative to the extra KL-cost from the approximate posterior.

Acknowledgements

We thank Ulrich Paquet for fruitful feedback. The research was supported by Danish Innovation Foundation, the NVIDIA Corporation with the donation of TITAN X GPUs. Marco Fraccaro is supported by Microsoft Research through its PhD Scholarship Programme.

References

Bastien, Frédéric, Lamblin, Pascal, Pascanu, Razvan, Bergstra, James, Goodfellow, Ian J., Bergeron, Arnaud, Bouchard, Nicolas, and Bengio, Yoshua. Theano: new features and speed improvements. In *Deep Learning and Unsupervised Feature Learning, workshop at Neural Information Processing Systems*, 2012.

Basu, Sugato, Banerjee, Arindam, and Mooney, Raymond J. Semi-supervised clustering by seeding. In *Proceedings of the International Conference on Machine Learning*, 2002.

Bengio, Yoshua, Courville, Aaron, and Vincent, Pascal. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8), 2013.

Bowman, S.R., Vilnis, L., Vinyals, O., Dai, A.M., Jozefowicz, R., and Bengio, S. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

Burda, Yuri, Grosse, Roger, and Salakhutdinov, Ruslan. Importance Weighted Autoencoders. *arXiv preprint arXiv:1509.00519*, 2015a.

Burda, Yuri, Grosse, Roger, and Salakhutdinov, Ruslan. Accurate and conservative estimates of mrf log-likelihood using reverse annealing. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2015b.

Chen, Xi, Kingma, Diederik P., Salimans, Tim, Duan, Yan, Dhariwal, Prafulla, Schulman, John, Sutskever, Ilya, and Abbeel, Pieter. Variational Lossy Autoencoder. In *International Conference on Learning Representations*, 2017.

Dieleman, Sander, Schlter, Jan, Raffel, Colin, Olson, Eben, Sønderby, Søren K, Nouri, Daniel, van den Oord, Aaron, and and, Eric Battenberg. Lasagne: First release., August 2015.

Fraccaro, Marco, Sønderby, Søren Kaae, Paquet, Ulrich, and Winther, Ole. Sequential neural models with stochastic layers. In *Advances in Neural Information Processing Systems*. 2016.

Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. In *Advances in Neural Information Processing Systems*. 2014.

Gulrajani, Ishaan, Kumar, Kundan, Ahmed, Faruk, Ali Taiga, Adrien, Visin, Francesco, Vazquez, David, and Courville, Aaron. PixelVAE: A latent variable model for natural images. *arXiv e-prints*, 1611.05013, November 2016.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of International Conference on Machine Learning*, 2015.

Jang, Eric, Gu, Shixiang, and Poole, Ben. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Kingma, Diederik and Ba, Jimmy. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 12 2014.

Kingma, Diederik P., Rezende, Danilo Jimenez, Mohamed, Shakir, and Welling, Max. Semi-Supervised Learning with Deep Generative Models. In *Proceedings of the International Conference on Machine Learning*, 2014.

Kingma, Diederik P, Salimans, Tim, Jozefowicz, Rafal, Chen, Xi, Sutskever, Ilya, and Welling, Max. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*. 2016.

Kingma, Diederik P; Welling, Max. Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*, 12 2013.

Krishnan, Rahul G, Shalit, Uri, and Sontag, David. Deep Kalman filters. *arXiv:1511.05121*, 2015.

Lake, Brenden M, Salakhutdinov, Ruslan R, and Tenenbaum, Josh. One-shot learning by inverting a compositional causal process. In *Advances in Neural Information Processing Systems*. 2013.

Maaløe, Lars, Sønderby, Casper K., Sønderby, Søren K., and Winther, Ole. Auxiliary Deep Generative Models. In *Proceedings of the International Conference on Machine Learning*, 2016.

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

Miyato, Takeru, Maeda, Shin-ichi, Koyama, Masanori, Nakae, Ken, and Ishii, Shin. Distributional Smoothing with Virtual Adversarial Training. *arXiv preprint arXiv:1507.00677*, 7 2015.

Ranganath, Rajesh, Tran, Dustin, and Blei, David M. Hierarchical variational models. *arXiv preprint arXiv:1511.02386*, 11 2015.

Rasmus, Antti, Berglund, Mathias, Honkala, Mikko, Valpola, Harri, and Raiko, Tapani. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, 2015.

Rezende, Danilo J., Mohamed, Shakir, and Wierstra, Daan. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *arXiv preprint arXiv:1401.4082*, 04 2014.

Rezende, Danilo Jimenez and Mohamed, Shakir. Variational Inference with Normalizing Flows. In *Proceedings of the International Conference on Machine Learning*, 2015.

Rolfe, Jason Tyler. Discrete variational autoencoders. In *Proceedings of the International Conference on Learning Representations*, 2017.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016.

Sønderby, Casper Kaae, Raiko, Tapani, Maaløe, Lars, Sønderby, Søren Kaae, and Winther, Ole. Ladder variational autoencoders. In *Advances in Neural Information Processing Systems 29*. 2016.

Springenberg, J.T. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*, 2015.

Tenenbaum, Joshua B., Griffiths, Thomas L., and Kemp, Charles. Theory-based Bayesian models of inductive learning and reasoning. *Trends in cognitive sciences*, 10 (7):309–318, July 2006.

Tran, Dustin, Ranganath, Rajesh, and Blei, David M. Variational Gaussian process. In *Proceedings of the International Conference on Learning Representations*, 2016.

Appendix D

Condition monitoring in PV systems by semi-supervised machine learning

---

In this appendix we include:

Maaløe, L., Spataru, S. V., Sera, D., Winther, O. (2018). Condition monitoring in photovoltaic systems by semi-supervised machine learning. Submitted to IEEE Transactions of Industrial Informatics.

Condition Monitoring in Photovoltaic Systems by Semi-Supervised Machine Learning

Lars Maaløe & Ole Winther
Applied Mathematics and Computer Science
Technical University of Denmark
Lyngby, 2800, Denmark
{larsma,olwi}@dtu.dk

Sergiu Spataru & Dezso Sera
Energy Technology Power Electronic Systems
Aalborg University
Aalborg, 9220, Denmark
{ssp,des}@et.aau.dk

***Abstract*—With the rapid increase in photovoltaic energy production there is a need for *smart* condition monitoring systems ensuring maximum throughput. Complex methods such as drone inspections are costly and labor intensive, hence condition monitoring by utilizing sensor data is attractive. In order to recognize meaningful patterns from the sensor data there is a need for expressive machine learning models. However, supervised machine learning, e.g. regression models, suffer from the cumbersome process of annotating data. By utilizing recent state of the art semi-supervised machine learning based on probabilistic modeling, we show that we are able to perform condition monitoring in a photovoltaic system with high accuracy and only a small fraction of annotated data. The modeling approach utilizes all the unsupervised data by jointly learning a low-dimensional feature representation and a classification model in an end-to-end fashion. By analysis of the feature representation, new internal condition monitoring states can be detected, proving a practical way of updating the model for better monitoring. We present (i) an analysis that compares the proposed model to corresponding purely supervised approaches, (ii) a study on the semi-supervised capabilities of the model, and (iii) an experiment in which we simulate a real-life condition monitoring system.**

I. Introduction

With an ever increasing growth in photovoltaic (PV) energy production the sheer size of individual power plants are growing at a rapid pace [1]. Building and operating such PV plants has become a viable business in many countries. High PV energy production and maximized yield are fundamental for a profit margin. A challenge is not solely detecting an anomaly in the PV power plant, but also to optimize the operation and maintenance costs once detected [2]. Here condition monitoring plays a crucial role since it is key to identify the specific system state to ascertain its impact on energy production and ensure minimal maintenance cost, e.g. panel cleaning, replacement, circuit or diode check [3]. Another challenge is the size of the PV plants. Minimally, the performance of the strings or arrays needs to be monitored. In a MW range there will be hundreds of PV performance computational streams to monitor in real time or periodically [2].

Many PV plant conditions can result in decreased yield. Amongst the conditions are (i) weather patterns, (ii) PV panel aging, (iii) evolving faults, e.g. diode failure or glass breakage, and (iv) faulty installation of the PV panels [4]. It is quite simple to detect anomalies in energy production, however, it is more complex to find the accurate source of the anomaly. Furthermore, the cause may be a result of a chain of events for which the causality is highly non-trivial.

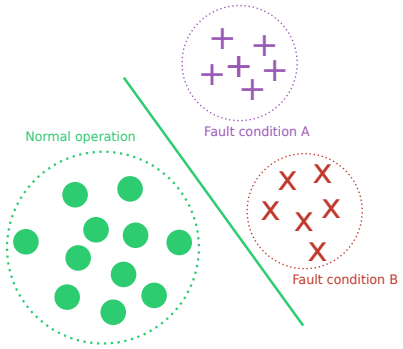


Fig. 1: Example of a supervised classification algorithm that is *trained* on a two-dimensional input $\{x_1, x_2\}$ so that it learns to discriminate between two categories: (i) the normal operation and (ii) fault conditions, of a PV plant. In this example the two categories are linearly separable, but that is often not the case in real-life settings.

Several alternatives for better condition monitoring exist many of which include quite costly add-ons, e.g. increased amount/accuracy of sensors and infrared inspection [5]. Another complementary approach is traditional statistical analysis of the data [6], but this is resource intensive. A less expensive alternative is a data-driven approach in which supervised machine learning models parameterized by for example neural networks learn from the vast amount of incoming sensor data. These machine learning models have proven efficient within noise resiliency and in finding non-linear correlations [7], [8], [9], [10], [11]. However, there is an inherent problem in assumptions made when applying highly expressive neural networks to the problem of condition monitoring since they are mostly formulated in a supervised setting. This means that we generally expect a large dataset containing condition-data with adhering labels. Therefore, in order to get started, one

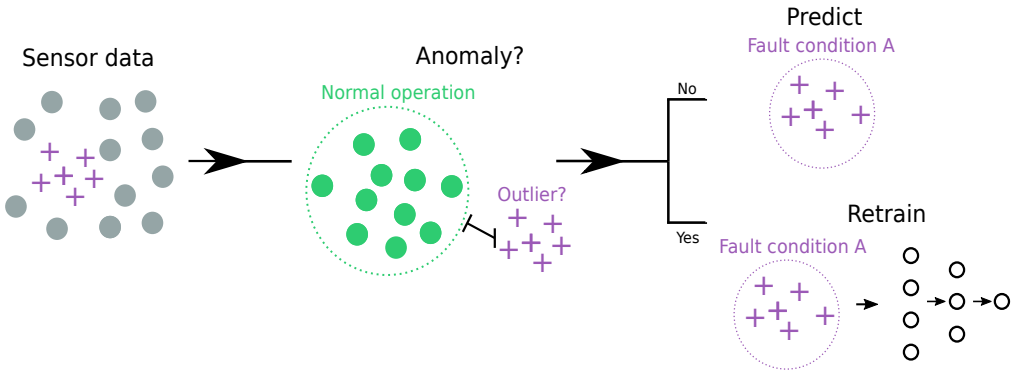


Fig. 2: A visualization of the condition monitoring system. The sensor data is propagated through the machine learning framework, and the model detects whether the data point is an outlier. If it is an outlier the sensor data must be manually inspected, and the machine learning framework retrained. If the incoming sensor data is not an outlier, the framework will predict the state of the condition. If the fault state is detected as a fault maintenance will be scheduled accordingly.

must (i) predefine all potential non-overlapping conditions that may happen in a PV plant, (ii) have a vast distribution of annotated data-points for each condition, and (iii) expect no anomalies from the already defined problem. It is quite clear that (i) will introduce a constraint on how specific we can be in defining a condition, since many have a tendency to overlap. (ii) is also limiting since the data of a PV plant is not directly interpretable by a human. Therefore one needs to engage in a costly annotation of data-points in order to train the relatively data-hungry neural networks. Finally, (iii) is posing a limit of supervised neural networks, since they are normally not modeled with an uncertainty, resulting in a risk of an overly confident estimate of a severe anomaly [12].

Before proposing a solution to the above it is important to specify how PV plant condition-data can be defined. In this research the conditions are expressed by the output of sensors, monitoring the PV array current, voltage, in-plane irradiance, external temperature, PV module temperature, and wind speed. The sensor inputs are recorded with a specific temporal resolution. The hypothesis is that, in cohesion, all of these sensor inputs will have unique patterns representing a PV plant condition. We propose a state of the art semi-supervised probabilistic machine learning framework that can capture the unique patterns and cluster them accordingly to their respective similarity. Furthermore, as part of the framework a supervised classifier, learned from a pre-defined annotation process, categorizes each of these clusters. The probabilistic framework thus models the joint distribution of the condition data and the PV plant state. This should be contrasted to traditional supervised approaches that model the state *given* the condition data. The big advantage of the model is that it can capture condition data anomalies while also classifying *known* conditions. In addition to this, the number of annotated data-points needed is very low.

The machine learning framework works by learning a distribution over the PV power plant conditions and thereby correlate *new* data points with the learned distribution. In recent years there has been several notable contributions within probabilistic semi-supervised learning methods. Amongst them are [13], [14] that utilize the variational auto-encoder framework (VAE) [15], [16] for a Bayesian approach in modeling the joint probability between the data and labels. In this paper we utilize the skip deep generative model (SDGM) from [14].

The paper is structured such that we give a background to PV condition monitoring, supervised machine learning for fault detection, and the SDGM. Next we introduce the experimental setup followed by results. We show that SDGM can indeed be used as a machine learning model for condition monitoring, and performs significantly better than its supervised counterparts, even in a fully supervised setting. Finally, we simulate a real-life condition monitoring setup where PV plant conditions are introduced sequentially. In these experiments we show how SDGM is able to detect anomalies, and that retraining the system improves condition monitoring performance.

II. Detection and identification of PV power loss and failures trough classification methods

A. PV failures and factors causing power loss

There exist a number of external factors that can cause power loss in a PV system, in addition to PV specific degradation modes [4]. These can be roughly categorized in three groups. The first group covers optical losses and degradation, such as soiling, snow, or shading affecting the module surface [17], as well as discoloration of the encapsulant [4]. These optical power loss factors can be relatively easily detected through visual inspection, however this is not always feasible, for large or hard to reach PV installations. Moreover, detecting them from production measurements can be difficult, since their associated failure patterns in the power measurements

are irregular, depending on the size and relative position of the soiling, shading, etc. Detecting such failures is important since some of them can be remedied relatively easy, through cleaning of the PV panels.

A second category of factors causing power loss in a PV system, is the degradation of the electrical circuit of the PV module. In the most severe cases these are represented by open-circuit and short-circuit faults within the PV array and associated cabling [4]. But there can also be partial degradation, due to moisture ingress and corrosion of the electrical pathways [18], causing an increased series resistance of the PV array [19]. Such faults are generally difficult to detect through visual inspection, and require thermal IR imaging or electroluminescence to detect. However, they cause more predictable patterns in the production measurements, such as voltage drops proportional to the increase in series resistance. Such failure can cause localized heating and hot-spots, posing a risk of arcing and fire.

The third category corresponds to degradation of the solar cells, which in turn can occur due to a number of stress factors such as: (i) thermo-mechanical stress, causing solar cell cracks, associated with increased series resistance, shunting and localized heating [19], [4]; (ii) voltage stress, causing potential-induced degradation, primarily associated with a decrease in the cells shunt resistance, but also corrosion and delamination in the case of some thin film technologies [20]; (iii) diurnal and seasonal variations affecting solar cells with metastable performance behavior, such as certain thin film technologies [21]. Degradation modes in this category are more difficult to detect, and the associated failure patterns in production measurements are more complex. Nonetheless, identifying such failures in their incipient phase is of utmost importance, since they are a symptom of a more serious, system-wide problems, such as bad system design, installation practice or module quality, which should be resolved while the modules and PV system are still in warranty.

The types of power loss factors and degradation modes that can affect PV systems are varied and difficult to formalize. And, only few of them may affect a PV system within its lifetime depending mainly on the solar cell technology, panel design and quality, environmental and operational conditions, and installation and maintenance practices.

B. Failure detection through supervised classification

Two of the main prerequisites for implementing supervised classification in a condition monitoring system, are: (i) the *a priori* knowledge of the fault types/classes that will occur/need to be detected in the PV system; and (ii) representative measurement datasets for each of the fault classes, necessary for *training* the classification model. Once these perquisites are met, and appropriately monitored, production variables are chosen as input, and classifiers are trained for each fault class (cf. Figure 1). Once trained, each classifier will operate continuously, monitoring the production variables, and will be able to discern if the system is in *Normal operation*, or a specific fault class has occurred (denoted as *Fault condition "A" and "B"* in the example in Figure 1).

There exist many types of supervised classification algorithms, e.g. support vector machines (SVM) [22], random forest (RF) [23], and multilabel logistic regression (MLR). These are all very expressive models, however, with the rise of deep learning [24], we have seen a multitude of improvements from models that can capture highly non-linear correlations in the data. The improvements mainly concern areas such as image classification [25] and automatic speech recognition [26]. However, the more *expressive* models also gain traction within renewable energy, e.g. for condition monitoring in wind turbines [10] and as forecasting models for solar irradiance [27]. Defining the deep neural network is not a simple task, due to the vast number of choices that needs to be taken in regards to type of architecture, depth, regularization and much more.

The main challenge in implementing a supervised classification algorithm for detecting faults in a PV system, is obtaining the necessary PV production measurement datasets characterizing the different fault classes. Since there are no standardized fault classes and representative datasets, faults of different types and severities can occur throughout the 25+ year expected lifetime of the PV system.

C. Proposed failure detection through semi-supervised classification

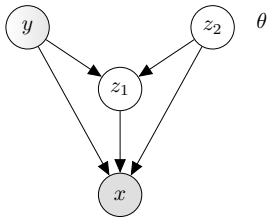
A possible solution is to combine a supervised classification method with a data clustering method, that is able to detect anomalous patterns in the monitored PV production data. Next, on-site inspection of the event/fault by maintenance personal, can help identify the type or *class* of this event/fault. The associated production measurements can then be used to retrain a supervised classifier for the detected event/fault class, such that future instances of the event/fault will be automatically detected and identified by the condition monitoring system, which is continuously learning new fault classes, as it's operating (cf. Figure 2).

D. Theory of the semi-supervised learning framework

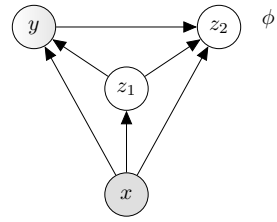
We propose to solve the problem for semi-supervised condition monitoring by learning a feature representation z of the PV condition data x as a continuous conditional probability density function, $p(z|x)$, and the classification task of the PV state y as a discrete conditional probability density, $p(y|x)$. In order to learn both models jointly from both labeled and unlabeled data the two models must be define in a way where they share parameters. By applying Bayes theorem we can formulate the problem by:

$$p(z, y|x) = \frac{p(x|z, y)p(z)p(y)}{\int_{z,y} p(x|z, y)p(y)p(z)dz} \ , \tag{1}$$

where we assume the latent variable feature representation z and state labels are y to be a priori statistically independent, $p(z, y) = p(z)p(y)$. In a scenario with complex input distributions, e.g. sensor input from a PV power plant,



(a) Generative model p_θ



(b) Inference model q_ϕ

Fig. 3: The graphical model of the SDGM for semi-supervised learning [14]. The model is defined by two continuous latent variables, z_1 and z_2, a partially observed discrete latent variable y, and a fully observed input, x. (a) depicts the generative model and (b) the inference model, also known as the variational approximation. Each union of incoming edges to a node defines a densely connected deep neural network.

the posterior $p(z, y|x)$, becomes intractable. Therefore we formulate the problem such that we learn an approximation, $q(z, y|x)$, to the posterior through variational inference [28]. SDGM is an example of this probabilistic framework which enables the use of stochastic gradient ascent methods for optimizing the parameters of the generative model, $p_\theta(x, y, z)$, and the variational approximation, $q_\phi(z, y|x)$. θ and ϕ denotes the parameters of the generative model and the variational approximation (also denoted inference model), respectively. Both are constructed from deep neural networks (cf. Figure 3). We learn the model parameters by jointly maximizing the objective $\mathcal{L}(x_l, y_l)$ for labeled data x_l, y_l and $\mathcal{U}(x_u)$ for unlabeled data x_u:

$$\mathcal{J} = \sum_{x_l, y_l} \mathcal{L}(x_l, y_l) + \sum_{x_u} \mathcal{U}(x_u) \ . \tag{2}$$

SDGM defines two continuous latent variables, $z = z_1, z_2$ and the discrete partially observed latent variable y. The continuous distributions for the latent variables z are defined as Gaussian distributions and the discrete distribution y is a Categorical distribution. For the labeled data we optimize the parameters, θ, ϕ w.r.t. a lower bound on the evidence $p(x)$ (ELBO):

$$\begin{aligned} &\log p_\theta(x, y) = \log \int_{z_1} \int_{z_2} p_\theta(x, y, z_1, z_2) dz_2 dz_1 \\ &\geq \mathbb{E}_{q_\phi(z_1, z_2|x, y)} \left[\log \frac{p_\theta(x, y, z_1, z_2)}{q_\phi(z_1, z_2|x, y)} \right] \\ &\equiv \mathcal{F}(x, y) \ , \end{aligned} \tag{3}$$

with

$$q_\phi(z_1, z_2|x, y) = q_\phi(z_1|x) q_\phi(z_2|z_1, y, x), \tag{4}$$

$$p_\theta(x, y, z_1, z_2) = p_\theta(x|z_1, z_2, y) p_\theta(z_1|y, z_2) p_\theta(y) p_\theta(z_2) \ . \tag{5}$$

Since the labeled ELBO does not include the classification error we add the Categorical cross-entropy loss

$$\mathcal{L}(x, y) = \mathcal{F}(x, y) + \alpha \cdot \mathbb{E}_{q_\phi(z_1, z_2|x, y)} \left[\log q_\theta(y|z_1, x) \right] \ , \tag{6}$$

where α is a constant scaling term defined as a hyper-parameter. Similarly to the labeled loss we define the unlabeled loss for the unlabeled ELBO:

$$\begin{aligned} &\log p_\theta(x) = \log \int_{z_1} \sum_{y} \int_{z_2} p(x, y, z_1, z_2) dz_2 dy dz_1 \\ &\geq \mathbb{E}_{q_\phi(z_1, y, z_2|x)} \left[\log \frac{p_\theta(x, y, z_1, z_2)}{q_\phi(z_1, y, z_2|x)} \right] \\ &\equiv \mathcal{U}(x) \ , \end{aligned} \tag{7}$$

where

$$q_\phi(z_1, z_2, y|x) = q_\phi(z_1|x) q_\phi(y|z_1, x) q_\phi(z_2|z_1, y, x). \tag{8}$$

In this paper we restrict the experiments to only use densely connected neural networks, but simple extensions to the model include recurrent neural networks and convolutional neural networks that has proven efficient in modeling temporal as well as spatial information within condition monitoring [27], [10].

Besides being amongst state-of-the-art within semi-supervised image classification, SDGM posses another intriguing property for condition monitoring as opposed to other semi-supervised approaches. Since we are optimizing the ELBO, we can use this as an anomaly measure. Thus, if the value of the ELBO for a specific data-point is far below the value of the ELBO that was evaluated during optimization, we can define the data-point as an anomaly.

III. Experimental application and tests

In order to validate whether the proposed SDGM can be utilized for condition monitoring we have recorded a dataset of the sensor data from a small-scale PV plant (cf. Table I), consisting on an inverter and two strings. During the timespan of the recording, we witnessed 10 different categories that we used as labels. In order to benchmark the machine learning framework we have defined two comparable supervised machine learning models. The dataset in cohesion with the two supervised models will act as a benchmark of the performance of the proposed machine learning framework.

TABLE I: The sensor output from the PV system that was used to train the machine learning models. The PV system comprised of two strings and an inverter.

SENSOR INPUT	DESCRIPTION
I	CURRENT
V	VOLTAGE
G	PLANE-OF-ARRAY IRRADIANCE
TEXT	EXTERNAL TEMPERATURE
TMOD	MODULE TEMPERATURE
W	WINDSPEED

A. Field test setup and dataset

To evaluate the progressive learning and fault detection capabilities of the proposed condition monitoring system, we performed measurements and tests on a 0.9 kWp roof-mounted PV string (eight multicrystalline silicon modules). The PV string was connected to a 6 kWp Danfoss TLX Pro string inverter that was continuously monitoring the string current (I), voltage (V), plane-of-array irradiance (G), external temperature (TExt), module temperature (TMod), and windspeed (W), with a 1 minute sampling time.

Since the test PV system is normally not affected by any faults, we created seven power loss events/fault classes, by applying different types of shading on the panels, as well as by connecting different power resistors on series with the PV string, to emulate series resistance type faults. In addition we also recorded PV production for when the PV system was covered by snow, for a clear sky and a cloudy sky day. The ten conditions/fault classes are outlined in Table II, and will be used as *class labels* for testing the classifiers in the next sections.

Another important step in designing a classification model is choosing appropriate input variables. Minimally, PV array current and voltage are monitored in a PV system, and we denote this case as the *simple monitoring* case. Additional monitoring input variables can be the solar irradiance, module temperature, external temperature, wind speed. These are less commonly monitored in small PV installations, due to the additional cost of the sensors, however, in larger PV plants these are usually monitored by accurate weather stations. We will denote the case including the ambient conditions as input variables, the *complex monitoring* case.

The categories are skewed in accordance to the weather pattern during the 2 months, e.g. there is a majority of data points for which there was snow (cf. Table II). For each learned model, we run a 5-fold Monte Carlo cross-validation with a random split of 80% for training and 20% for testing. The labeled samples are either sampled uniformly or progressively for each PV system state category.

B. Machine learning setup

In order to evaluate the proposed machine learning framework we first define a solid baseline for comparison. Since the SDGM is parameterized by neural networks, we construct a supervised neural network for classification with equivalent parameterization to $q_\phi(y|z_1, x)$. Furthermore, we also define a simple linear classification model, in order to conclude whether the added complexity from the neural networks is needed for modeling this dataset. The supervised deep neural network for classification, is denoted multi-layer perceptron (MLP), and the linear model is referred to as multi-label regression (MLR).

(i) In the first experiment we benchmark SDGM against MLP and MLR in a fully supervised setting, thus all labels for the entire dataset is given during training. The aim of this experiment is to see whether MLP performs significantly better than MLR and whether SDGM performs approximately equivalent to MLP. We perform this experiment on both the simple and complex monitoring case.

(ii) Next, we investigate the semi-supervised performance of the SDGM. In order to do this, we simulate a scenario where only a fraction of data in the PV sensor dataset is given. Since MLP and MLR are supervised models, they are only able to learn from this fraction of labeled data, whereas SDGM can utilize the unlabeled fraction also. The fraction of labeled data is randomly sampled uniformly across categories, such that there is an even representation of each category in the labeled dataset.

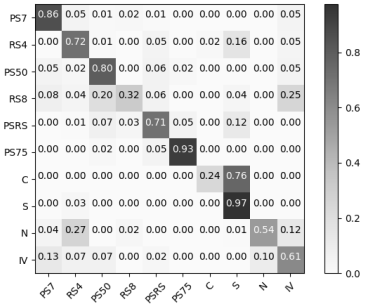
(iii) Finally, we simulate a real-life PV plant condition monitoring system, in which we assume that each condition is introduced to the power plant sequentially (cf. Figure 2). First we initialize the dataset with only 1 labeled data-point from each category, in order to introduce the minimal amount of categorical knowledge in the classifier. Next we introduce 500 labeled samples from the first category in Table II and optimize MLP and SDGM. Now we can estimate the ELBO in Equation 7 for the data-points of the categories that are included during training and the ones that are not. We can also estimate the accuracy of the classifiers in SDGM and the MLP. We will expect that the estimate of the ELBO will be significantly lower for the categories that are not included in the dataset as opposed to the ELBO for the categories that are included. This indicates an anomaly. Next we progressively include another category from Table II and perform the same analysis until we have evaluated 6 categories.

The SDGM[1] consists of 2 densely connected deep neural neural networks with parameters θ in the generative model and 3 densely connected deep neural networks with parameters ϕ in the inference model. The neural networks in both the SDGM and MLP contains 2 hidden layers with 50 units in each. We use the ReLU [29] activation function as a non-linearity and ADAM [13] for optimizing the parameters. For the MLP we use a dropout [30] rate of 0.5 and for the MLR we use L2 regularization. Model training is stopped upon saturation of the validation error. The α constant is defined as in [14]. During optimization of the SDGM we utilize the warm-up introduced in [31], [32].

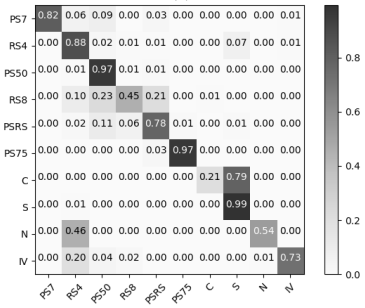
[1]For details on experimental implementation and code refer to [14] and github.com/larsmaaloee/auxiliary-deep-generative-models.

TABLE II: An overview of the PV system dataset used for this research. The dataset comprises of 10 categories from approximately 15,000 data samples of a varied representation.

CONDITION/FAULT CLASS	DESCRIPTION	SAMPLES
PS7	UNIFORM SHADING ON ALL LOWER CELLS OF THE MODULES	10.68%
RS4	50% INCREASE IN STRING SERIES RESISTANCE	10.18%
PS50	PARTIAL SHADING ON 50% OF A SUBMODULE	10.83%
RS8	100% INCREASE IN STRING SERIES RESISTANCE	5.11%
PSRS	COMBINED 50% SHADING ON A SUBMODULE WITH 50% INCREASE IN STRING RS	10.93%
PS75	SHADING ON 50% OF A SUBMODULE + 25% OF ANOTHER SUBMODULE	10.60%
C	CLOUDY SKY DAY	4.60%
S	SNOW ON THE MODULES	27.64%
N	CLEAR SKY DAY	4.67%
IV	SHADING ON 3/4 OF CELL AREA OF 6 SUBMODULES	4.78%



(a)



(b)

Fig. 4: Normalized confusion matrices for (a) MLR and (b) MLP trained on the fully labeled *complex* dataset. The x-axis denotes the predicted labels and the y-axis the true labels.

IV. RESULTS

We propose three experiments, introduced above. In the first experiment we will benchmark the SDGM against the MLP and MLR in a fully-supervised setting. Next we will evaluate the semi-supervised power of the SDGM. Finally, we present results for the study in which we simulate a real-life condition monitoring system.

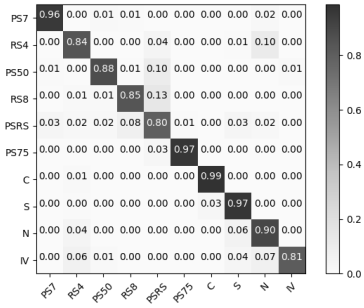
A. Supervised Condition Monitoring Accuracy

TABLE III: Fully supervised baseline on MLR, MLP and SDGM with the *simple* sensor input, {I, V}, and the *complex* input, {I, V, G, TExt, TMod, and W}.

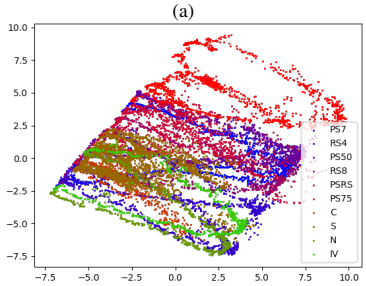
	ACCURACY I,V	ACCURACY I,V,G,TEXT TMOD,W
MLR	51.62%	77.33%
MLP	77.81%	89.11%
SDGM	**79.06%**	**92.47%**

Table III presents the baseline results of MLR, MLP and SDGM in a fully supervised learning setup. By utilizing more sensor attributes (complex vs. simple), the performance increases well over 10% across all models. This proves that the additional sensor inputs (G, TExt, TMod, and W) are very useful for condition monitoring. When comparing the non-linear MLP to the linear MLR we also achieve a significant improvement in performance, indicating that the input data is not linearly separable, and that the added complexity of the neural networks is worthwhile. The most surprising finding is that the SDGM is performing significantly better than the MLP. We believe that this is due to the fact that SDGM also learns a latent clustering of the data that is correlated with the PV state. Thereby, the model can discriminate between the labels and the cluster representations, meaning that it can put less emphasis on labeled information that does not seem to correlate with the distribution. Hence, if a small fraction of faulty labels exist in the training dataset, SDGM is able to *ignore* this information and thereby achieve better generalization towards the validation dataset.

Figure 4 shows how the wrongly classified examples from the MLR and MLP are quite similar. The highest misclassification rate lies between cloudy and snowy weather, {C, S}. Other misclassification rates mainly lie between {RS8, PS50}, {N, RS4}, {RS8, IV}, and {N, RS4}. When we compare the results of MLR and MLP to the SDGM (cf. Figure

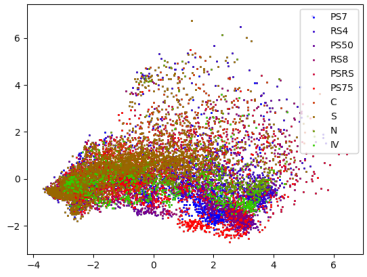


(a)

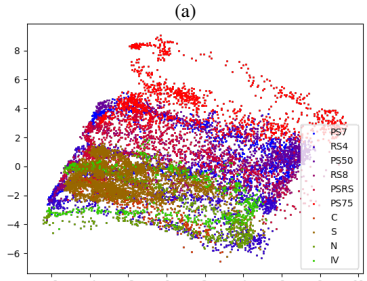


(b)

Fig. 5: (a) Normalized confusion matrices for SDGM trained on the fully labeled *complex* dataset. (b) PCA (principal components 1 and 2) on the latent space.



(a)



(b)

Fig. 6: PCA (principal component 1 and 2) visualization of the latent space for SDGMs trained a dataset with only 100 labeled samples. (a) Shows the latent space for a model trained on the simple dataset, {I, V} and (b) for {I, V, G, TExt, TMod, W} as input.

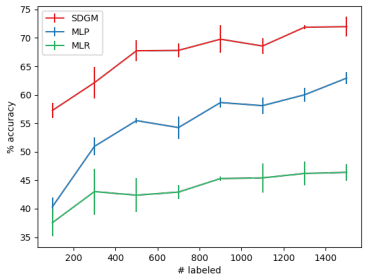
5a) we can read from the confusion matrix that the SDGM manages to learn the difference between cloudy and snowy, {C, S}. Furthermore the remainder of the most prominent misclassification rates are significantly decreased. In order to analyze what is learned in the latent variables of SDGM, we plot the first two principal components from a principal component analysis (PCA) (cf. Figure 5b). The visualization of the latent space shows clear discrimination between categories. Furthermore we can also see that the data lies on manifolds resembling the movement of the sun.

B. Semi-Supervised Condition Monitoring Accuracy

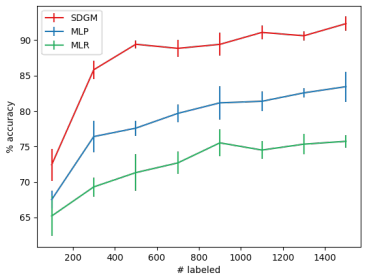
In order to evaluate the semi-supervised performance of SDGM, we define 8 datasets with different fractions of labeled data that is randomly subsampled across the categories in Table II for each of the trained models, {100, 300, ..., 1500}. Figure 7 shows SDGM's significant increase in performance by utilizing the knowledge in the unlabeled data. For the simple dataset, with {I, V} as input, we see that the supervised models, MLR and MLP, achieves an accuracy of 35%-45% by learning from 100 labeled data-points, whereas the SDGM achieves 55%-60%. As expected, the relative improvement from using SDGM stays significant when introducing more labels. Similarly, to the supervised analysis above, all models achieve a significant improvement when adding more sensor inputs, {I, V, G, TExt, TMod, W}. When comparing the results of the semi-supervised SDGM with the supervised SDGM, we see that the models trained on 1500 labeled data points actually exceeds the performance of the fully-supervised model, 93.12% compared to 92.47%. Again the reason for this may be that with fewer labeled examples, SDGM put a larger emphasis on the unlabeled data and thereby it is not as prone to faulty annotations. In Figure 6 we visualize the latent representations by PCA for the SDGM trained with 100 labeled data-points on the simple and complex input. It is clear that the model trained on the complex is better at discriminating between the categories than the model trained on the simple input. Furthermore, when comparing Figure 5b with 6b we see clear indications that the increase in labels results in better discrimination between condition states.

C. Adding PV Conditions Progressively

Figure 8a presents the results of a SDGM and MLP learned up to 6 categories. As expected the accuracy for all categories increases when more categories are added to the dataset. Again it is clear that the SDGM is able to utilize the information from the unlabeled examples as well as the very sparse information from the other categories to significantly outperform the MLP. In Figure 8b we visualize the *level of certainty*, ELBO (cf. Equation 7), and can easily discriminate the categories included during training from the categories

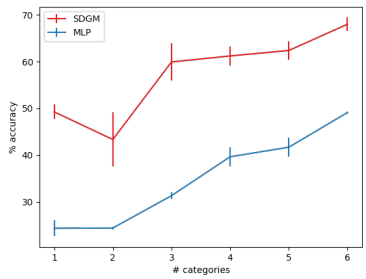


(a)

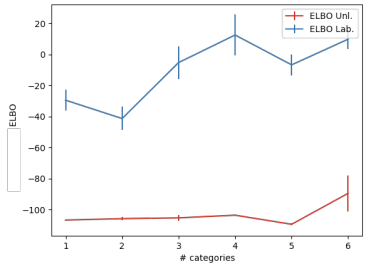


(b)

Fig. 7: Comparison between the supervised MLP, MLR and the semi-supervised SDGM trained on an increasing amount of randomly sampled and evenly distributed labeled data points. For each number of labeled data points, we trained 10 different models since there may exist a large variance between the quality of the subsampled labeled data points. (a) shows the accuracy with one standard deviation for models trained on the simple input distribution {I,V}, and (b) the accuracy for models trained on the complex input distribution, {I, V, G, TExt, TMod, W}.



(a)



(b)

Fig. 8: SDGMs and MLPs trained from datasets where we randomly subsample a single data-point from each category and then progressively add 500 randomly labeled data points for each category and training a new MLP and SDGM for each progression. (a) presents the accuracy of the classifiers for the SDGM and MLP. (b) presents the ELBO for the data categories included during training (ELBO Lab.) and the data categories that are not included during training (ELBO Unl.). The categories that are progressively added is following the order of Table II, i.e. first {*PS7*}, next {*PS7, RS4*}, until reaching {*PS7, RS4, PS50, RS8, PSRS, PS75*}.

that are not included. So for a model trained on only {PS7} data, it is easy to detect {RS4, PS50, RS8, PSRS, PS75, C, S, N, IV} conditions as anomalous, and for a model trained on {PS7,RS4} it is easy to detect {PS50, RS8, PSRS, PS75, C, S, N, IV} as anomalous. In order to state whether a PV plant condition is an anomaly the operator needs to define a threshold value. In this experiment a suitable threshold could be that PV plant conditions with an ELBO below -60 nats is considered an anomaly. Upon realization of an anomaly, the PV plant operator will initiate a brief annotation process and retrain the SDGM framework, so that the new states is now within the known operational condition.

V. Conclusion

In this research we have proposed an approach to PV condition monitoring that simultaneously learns classification and anomaly detection models. This can significantly improve the throughput of energy production and lower the maintenance cost of PV power plants. We have shown that the approach is easy to train on a rather simple dataset and that it is easily interpretable by evaluating the classification results, the latent representations, and the ELBO. The main limitations of this research lies in the dataset used. Due to the representation and the amount of samples, it does not resemble the vast amount of data one could acquire from a large-scale PV power plant. However, deep neural networks have a tendency to improve when introducing more data, meaning that we can hypothesize that the results would only improve. Another interesting direction for future research would be to investigate the possibility for *transfer learning* between PV power plant configurations, so that one could seamlessly deploy a SDGM that is learned on one PV plant, to another.

Acknowledgment

The research was supported by Innovation Fund Denmark and the NVIDIA Corporation with the donation of TITAN X GPUs.

REFERENCES

[1] I. PVPS, "Trends 2017 in phtovoltaic applications - survey report of selected iea countries between 1992 and 2016," International Energy Agency, Report, 25 Jan 2018.

[2] G. Mtter, T. Krametz, and P. Steirer, "Experiences with a performance package for multi-mw pv plants based on computations on top of monitoring," in *31st European Photovoltaic Solar Energy Conference and Exhibition*. WIP, Conference Proceedings, pp. 1675 – 1678.

[3] A. Woyte, M. Richter, D. Moser, N. Reich, M. Green, S. Mau, and H. G. Beyer, "Analytical monitoring of grid-connected photovoltaic systems," International Energy Agency, Report, 2014.

[4] M. Köntges, S. Kurtz, C. Packard, U. Jahn, K. A. Berger, K. Kato, T. Friesen, H. Liu, and M. Van Iseghem, "Review of failures of photovoltaic modules," International Energy Agency, Report, March 2014 2014.

[5] C. Buerhop-Lutz, H. Scheuerpflug, T. Pickel, C. Camus, J. Hauch, and C. Brabec, "Ir-imaging a tracked pv-plant using an unmanned aerial vehicle," in *32nd European Photovoltaic Solar Energy Conference and Exhibition*. WIP, Conference Proceedings, pp. 2016 – 2020.

[6] S. Vergura, G. Acciani, V. Amoruso, G. E. Patrono, and F. Vacca, "Descriptive and inferential statistics for supervising and monitoring the operation of pv plants," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 11, pp. 4456–4464, 2009.

[7] S. Silvestre, C. Aissa, and E. Karatepe, "Automatic fault detection in grid connected pv systems," vol. 94, pp. 119–127, 06 2013.

[8] L. L. Jiang and D. L. Maskell, "Automatic fault detection and diagnosis for photovoltaic systems using combined artificial neural network and analytical based methods," in *Proceedings of the IEEE International Joint Conference on Neural Networks*. IEEE Computer Society, 2015.

[9] M. H. Ali, A. Rabhi, A. E. H., and G. M. Tina, "Real time fault detection in photovoltaic systems," *Procedia Energy*, pp. 914–923, 2017.

[10] M. Bach-Andersen, B. Rømer-Odgaard, and O. Winther, "Deep learning for automated drivetrain fault detection," *Wind Energy*, vol. 21, pp. 29–41, Oct. 2017.

[11] M. Bach-Andersen, "A diagnostic and predictive framework for wind turbine drive train monitoring," Ph.D. dissertation, Technical University of Denmark, 2017.

[12] Y. Gal, "Uncertainty in deep learning," Ph.D. dissertation, University of Cambridge, 2016.

[13] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling, "Semi-Supervised Learning with Deep Generative Models," in *In Proceedings of the International Conference on Machine Learning*, 2014, pp. 3581–3589.

[14] L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther, "Auxiliary Deep Generative Models," in *Proceedings of the International Conference of Machine Learning*, 2016.

[15] M. Kingma, Diederik P; Welling, "Auto-Encoding Variational Bayes," in *In Proceedings of the International Conference on Learning Representations*.

[16] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic Backpropagation and Approximate Inference in Deep Generative Models," *arXiv preprint arXiv:1401.4082*, 04 2014.

[17] H. Laukamp, T. Schoen, and D. Ruoss, "Reliability study of grid connected pv systems, field experience and recommended design practice," International Energy Agency, Report, 2002.

[18] B. B. Yang, N. R. Sorensen, P. D. Burton, J. M. Taylor, A. C. Kilgo, D. G. Robinson, and J. E. Granata, "Reliability model development for photovoltaic connector lifetime prediction capabilities," in *39th IEEE Photovoltaic Specialists Conference (PVSC)*, 2013, Conference Proceedings, pp. 0139–0144.

[19] D. L. King, M. A. Quintana, J. A. Kratochvil, D. E. Ellibee, and B. R. Hansen, "Photovoltaic module performance and durability following long-term field exposure," *Progress in Photovoltaics: Research and Applications*, vol. 8, no. 2, pp. 241–256, 2000.

[20] W. Luo, Y. S. Khoo, P. Hacke, V. Naumann, D. Lausch, S. P. Harvey, J. P. Singh, J. Chai, Y. Wang, A. G. Aberle *et al.*, "Potential-induced degradation in photovoltaic modules: a critical review," *Energy & Environmental Science*, vol. 10, no. 1, pp. 43–68, 2017.

[21] T. Silverman, U. Jahn, G. Friesen, M. Pravettoni, M. Apolloni, A. Louwen, M. Schweiger, and G. Belluardo, "Characterisation of performance of thin-film photovoltaic technologies," International Energy Agency, Report, May 2014 2014.

[22] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[23] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct 2001.

[24] G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, pp. 1527–1554, 2006.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[26] W. Xiong, L. Wu, F. Alleva, J. Droppo, X. Huang, and A. Stolcke, "The microsoft 2017 conversational speech recognition system," *arXiv preprint arXiv:1708.06073*, 2017.

[27] A. Alzahrani, P. Shamsi, C. Dagli, and M. Ferdowsi, "Solar irradiance forecasting using deep neural networks," *Procedia Computer Science*, pp. 304–313, Nov. 2017.

[28] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Machine Learning*, vol. 37, no. 2, pp. 183–233, Nov. 1999.

[29] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks." in *AISTATS*, ser. JMLR Proceedings, vol. 15. JMLR.org, 2011, pp. 315–323.

[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *JMLR*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.

[31] C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther, "Ladder variational autoencoders," in *Advances in Neural Information Processing Systems*, 2016.

[32] S. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," *arXiv preprint arXiv:1511.06349*, 2015.

Appendix E

Feature map variational auto-encoders

In this appendix we include:

Maaløe, L., Winther, O. (2018). Feature map variational auto-encoders. To be submitted.

FEATURE MAP VARIATIONAL AUTO-ENCODERS

Lars Maaløe, Ole Winther
{larsma,olwi}@dtu.dk

Department of Applied Mathematics and Computer Science, Technical University of Denmark

ABSTRACT

There have been multiple attempts with variational auto-encoders (VAE) to learn powerful global representations of complex data using a combination of latent stochastic variables and an autoregressive model over the dimensions of the data. However, for the most challenging natural image tasks the purely autoregressive model still outperform the combined stochastic-autoregressive models. In this paper, we present simple additions to the VAE framework that generalize by embedding spatial information in the variational auto-encoder framework. We significantly improve the state-of-the-art results on MNIST and OMNIGLOT when the feature map parameterization are combined with the autoregressive PixelCNN approach. Interestingly, we also observe close to state-of-the-art results without the autoregressive part. This opens the possibility for high quality image generation with only one forward-pass.

1 INTRODUCTION

In representation learning the goal is to learn a posterior latent distribution that explains the observed data well (Bengio et al., 2013). Learning good representations from data can be used for various tasks such as generative modelling and semi-supervised learning (Kingma, 2013; Rezende et al., 2014; Kingma et al., 2014; Rasmus et al., 2015; Maaløe et al., 2016). The decomposition of variational auto-encoders (VAE) (Kingma, 2013; Rezende et al., 2014) provides the potential to disentangle the internal representation of the input data from local to global features through a hierarchy of stochastic latent variables. This makes the VAE an obvious candidate for learning good representations. However, in order to make inference tractable VAEs contain simplifying assumptions, which limits their ability to learn a good posterior latent representation.

In complex data distributions with temporal dependencies (e.g. text, images and audio), the VAE assumption on conditional independence in the input distribution limits the ability to learn local structures. This has a significant impact on its generative performance, and thereby also the learned representations. Additionally, the one-layered VAE model with a $\mathcal{N}(0, I)$ latent prior poses serious constraints on the posterior complexity that the model is able to learn. A deep hierarchy of stochastic latent variables should endow the model with more expressiveness, but the VAE has a tendency to *skip* the learning of the higher representations since they pose a direct cost in its optimization term.

There have been several attempts to eliminate the limitations of the VAE. Some concern formulating a more expressive variational distribution (Burda et al., 2015b; Rezende & Mohamed, 2015; Tran et al., 2016; Maaløe et al., 2016) where other concern learning a deeper hierarchy of latent variables (Sønderby et al., 2016). These contributions have resulted in better performance, but are still limited when modelling complex data distributions where a conditional independence does not apply. When parameterizing the VAE decoder with recurrent neural networks (Krishnan et al., 2015; Bowman et al., 2015; Fraccaro et al., 2016), the decoding architecture gets too powerful which results in unused latent stochastic variables (Chen et al., 2017).

The limitations of the VAE have spawned interest towards other generative models such as Generative Adversarial Networks (GAN) (Goodfellow et al., 2014) and the autoregressive PixelCNN/PixelRNN models (van den Oord et al., 2016b). These methods have proven powerful in learning good generative models, but the lack of stochastic latent variables makes them less suitable for representation learning purposes (Chen et al., 2017). Lately, we have seen several successful

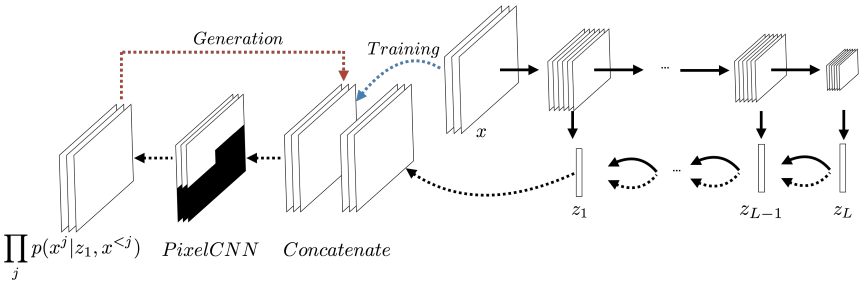


Figure 1: A visualization of FAME where the solid lines denote the variational approximation (inference/encoder/recognition) network and dashed lines denote the generative model (decoder) network for training. When performing reconstructions during training, the input image is concatenated with the output of the generative model (blue) and when generating the model follows a normal autoregressive sampling flow (red) while also using the stochastic latent variables $\mathbf{z} = z_1, ..., z_L$. Both the variational approximation and the generative model follow a top-down hierarchical structure which enables precision weighted stochastic variables in the variational approximation.

attempts to combine VAEs with PixelCNNs (Gulrajani et al., 2016; Chen et al., 2017). This results in a model where some global structure of the data is learned in the stochastic latent variables of the VAE and the local structure is learned in the PixelCNN. However, despite the additional complexity and potential extra expressiveness, these models do not outperform a simple autoregressive model (van den Oord et al., 2016a; Salimans et al., 2017).

In this paper we present the Feature Map Variational Auto-Encoder (FAME) that combines the top-down variational approximation presented in the Ladder Variational Auto-Encoder (LVAE) (Sønderby et al., 2016) with a spatial (feature map) representation in the deterministic layers and an autoregressive decoder. We show that (i) FAME outperforms previously state-of-the-art log-likelihood on MNIST and OMNIGLOT (ii) FAME learns a deep hierarchy of stochastic latent variables without inactivated latent units, (iii) by removing the autoregressive decoder FAME performs close to previous state-of-the-art log-likelihood suggesting that it is possible to get good quality generation with just one forward pass.

2 Feature Map Variational Auto-Encoder

The VAE (Rezende et al., 2014; Kingma, 2013) is a generative model with a hierarchy of stochastic latent variables:

$$p_\theta(x, \mathbf{z}) = p_\theta(x|z_1)p_\theta(z_L)\prod_{i=1}^{L-1} p_\theta(z_i|z_{i+1}) \,, \tag{1}$$

where $\mathbf{z} = z_1, ..., z_L$, θ denotes the parameters, and L denotes the number of stochastic latent variable layers. The stochastic latent variables are usually modelled as conditionally independent Gaussian distributions with a diagonal covariance:

$$p_\theta(z_i|z_{i+1}) = \mathcal{N}\Big(z_i; \mu_{\theta,i}(z_{i+1}), \mathrm{diag}(\sigma^2_{\theta,i}(z_{i+1}))\Big), \qquad p_\theta(z_L) = \mathcal{N}\Big(z_L; 0, I\Big) \,. \tag{2}$$

Since the posterior $p(\mathbf{z}|x)$ often is intractable we introduce a variational approximation $q_\phi(\mathbf{z}|x)$ with parameters ϕ. In the original VAE formulation, $q_\phi(\mathbf{z}|x)$ is decomposed as a bottom-up inference path

through the hierarchy of the stochastic layers:

$$q_\phi(\mathbf{z}|x) = q_\phi(z_1|x) \prod_{i=2}^{L} q_\phi(z_i|z_{i-1}) \,, \tag{3}$$

$$q_\phi(z_1|x) = \mathcal{N}\Big(z_1; \mu_{\phi,1}(x), \mathrm{diag}(\sigma^2_{\phi,1}(x))\Big) \,, \tag{4}$$

$$q_\phi(z_i|z_{i-1}) = \mathcal{N}\Big(z_i; \mu_{\phi,i}(z_{i-1}), \mathrm{diag}(\sigma^2_{\phi,i}(z_{i-1}))\Big) \,. \tag{5}$$

We optimize an *evidence lower-bound* (ELBO) to the log-likelihood $\log p_\theta(x) = \log \int_{\mathbf{z}} p_\theta(x, \mathbf{z}) d\mathbf{z}$. Burda et al. (2015a) introduced the importance weighted bound:

$$\log p(x) \geq \mathbb{E}_{q_\phi(\mathbf{z}^1|x)}, ..., \mathbb{E}_{q_\phi(\mathbf{z}^K|x)} \left[\log \sum_{k=1}^{K} \frac{p_\theta(x, \mathbf{z}^k)}{q_\phi(\mathbf{z}^k|x)} \right] \equiv \mathcal{L}_K(\theta, \phi; x) \tag{6}$$

and proved that $\mathcal{L}_K(\theta, \phi; x) \geq \mathcal{L}_L(\theta, \phi; x)$ for $K > L$. For $K = 1$ the bound co-incides with the standard ELBO: $\mathcal{L}(\theta, \phi; x) = \mathcal{L}_1(\theta, \phi; x)$. The hierarchical structure of both the variational approximation and generative model give the VAE the expressiveness to learn different representations of the data throughout its stochastic variables, going from local (e.g. edges in images) to global features (e.g. class specific information). However, we can apply as recursive argument Maaløe et al. (2017) to show that when optimizing with respect to the parameters θ and ϕ the VAE is regularized towards $q_\phi(z_L|z_{L-1}) = p_\theta(z_L) = \mathcal{N}(z_L; 0, I)$. This is evident if we rewrite Equation 6 for $K = 1$:

$$\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z_{1:L-1}|x)} \left[\frac{p_\theta(x, z_{1:L-1}|z_L)}{q_\phi(z_{1:L-1}|x)} \right] - \mathbb{E}_{q_\phi(z_{1:L-1}|x)} \left[KL\big(q_\phi(z_L|z_{L-1})||p_\theta(z_L)\big) \right] \,.$$

$KL\big(q_\phi(z_L|z_{L-1})||p_\theta(z_L)\big) = 0$ is a local maxima and learning a useful representation in z_L can therefore be disregarded throughout the remainder of the training. The same argumentation can be used for all subsequent layers $z_{2:L}$, hence the VAE has a tendency to collapse towards not using the full hierarchy of latent variables. There are different ways to get around this tendency, where the simplest is to down-weight the KL-divergence with a temperature term (Bowman et al., 2015; Sønderby et al., 2016). This term is applied during the initial phase of optimization and thereby downscales the regularizing effect. However, this only works for a limited number of hierarchically stacked latent variables (Sønderby et al., 2016).

Formulating a deep hierarchical VAE is not the only cause of inactive latent variables, it also occurs when the parameterization of the decoder gets too powerful (Krishnan et al., 2015; Fraccaro et al., 2016; Chen et al., 2017). This can be caused by using autoregressive models such as $p(x, \mathbf{z}) = \prod_j p(x^j|x^{<j}, \mathbf{z})p(\mathbf{z})$. Chen et al. (2017) circumvent this by introducing the Variational Lossy Auto-Encoder (VLAE) where they define the architecture for the VAE and autoregressive model such that they capture global and local structures. They also utilize the power of more expressive posterior approximations using inverse autoregressive flows (Rezende & Mohamed, 2015; Kingma et al., 2016). In the PixelVAE, Gulrajani et al. (2016) takes a similar approach to defining the generative model but makes a simpler factorizing decomposition in the variational approximation $q_\phi(\mathbf{z}|x) = \prod_i^L q_\phi(z_i|x)$, where the terms have some degree of parameter sharing. This formulation results in a less flexible model.

In Kingma et al. (2016); Gulrajani et al. (2016); Chen et al. (2017) we have seen that VAEs with simple decompositions of the stochastic latent variables and a powerful autoregressive decoder can result in good generative performance and representation learning. However, despite the additional cost of learning a VAE we only see improvement in the log-likelihood over the PixelCNN for small gray-scale image datasets (Salimans et al., 2017). We propose FAME that extends the VAE with a top-down variational approximation similar to the LVAE (Sønderby et al., 2016) combined with spatial stochastic latent layers and an autoregressive decoder, so that we ensure expressive latent stochastic variables learned in a deep hierarchy (cf. Figure 1).

2.1 Top-Down Variational Approximation

The LVAE (Sønderby et al., 2016) does not change the generative model but changes the variational distribution to be top-down like the generative model. Furthermore the variational distribution

shares parameters with the generative model which can be viewed as a precision-weighted (inverse variance) combination of information from the prior and data distribution. The variational approximation is defined as:

$$q_\phi(\mathbf{z}|x) = q_\phi(z_L|x) \prod_{i=1}^{L-1} q_\phi(z_i|z_{i+1}, x) \ . \tag{7}$$

The stochastic latent variables are all fully factorized Gaussian distributions and are therefore modelled by $q_\phi(z_i|z_{i+1}, x) = \mathcal{N}(z_i|\mu_i, \text{diag}(\sigma_i^2))$ for layers $i = 1, ..., L$. Instead of letting q and p have separate parameters (as in the VAE), the LVAE let the mean and variance be defined in terms of a function of x (the bottom-up data part) and the generative model (the top-down prior):

$$\mu_i = \frac{\mu_{\phi,i}\sigma_{\phi,i}^{-2} + \mu_{\theta,i}\sigma_{\theta,i}^{-2}}{\sigma_{\phi,i}^{-2} + \sigma_{\theta,i}^{-2}} \tag{8}$$

$$\sigma_i = \frac{1}{\sigma_{\phi,i}^{-2} + \sigma_{\theta,i}^{-2}} \ , \tag{9}$$

where $\mu_{\phi,i} = \mu_{\phi,i}(x)$ and $\mu_{\theta,i} = \mu_{\theta,i}(z_{i+1})$ and like-wise for the variance functions. This precision weighted parameterization has previously yielded excellent results for densely connected networks (Sønderby et al., 2016).

2.2 Convolutional Deterministic Layers and Autoregressive Decoding

We have seen multiple contributions (e.g. Gulrajani et al. (2016)) where VAEs (and similar models) have been parameterized with convolutions in the deterministic layers h_j^i, for $j = 1, ..., M$, and M is the number of layers connecting the stochastic latent variables z_i. The size of the spatial feature maps decreases towards higher latent representations and transposed convolutions are used in the generative model. In FAME we propose to add convolutional layers in the lateral architecture in a similar way:

$$\begin{aligned} h_{M,i} &= \texttt{CNN}(h_{<M,i}) \\ \mu_{\phi\vee\theta,i} &= \texttt{Linear}(\texttt{DENSE}(h_{M,i})) \\ \sigma_{\phi\vee\theta,i} &= \texttt{Softplus}(\texttt{DENSE}(h_{M,i})) \ , \end{aligned}$$

where `CNN` denote a convolutional neural network and `DENSE` a densely connected layer.

From van den Oord et al. (2016b;a); Salimans et al. (2017) we have seen that the PixelCNN architecture is very powerful in modelling a conditional distribution between pixels. In FAME we introduce a PixelCNN in the input dimension of the generative model $p_\theta(x|\mathbf{z})$ (cf. Figure 1). During training we concatenate the input with the reconstruction data in the channel dimension and propagate it through the PixelCNN, similarly to what is done in Gulrajani et al. (2016). When generating samples we fix a sample from the stochastic latent variables and generate the image pixel by pixel autoregressively.

3 Experiments

We test FAME on images from which we can compare with a wide range of generative models. First we evaluate on gray-scaled image datasets: statically and dynamically binarized MNIST (LeCun et al., 1998) and next OMNIGLOT (Lake et al., 2013). The OMNIGLOT dataset is of particular interest due to the large variance amongst samples. When modelling the gray-scaled images we assume a Bernoulli $\mathcal{B}$ distribution using a Sigmoid activation function as the output. We evaluate the performance with $\mathcal{L}_{5000}$ (cf. Equation 6).

We use a hierarchy of 5 stochastic latent variables where the stochastic latent layers are dense with sizes 64, 32, 16, 8, 4 (equivalent to Sønderby et al. (2016)). We apply batch-normalization (Ioffe & Szegedy, 2015) and ReLU activation functions as the non-linearity between all hidden layers $h_{i,j}$ and use a simple PixelCNN as in van den Oord et al. (2016b) with 4 residual blocks.

Because of the concatenation in the autoregressive decoder (cf. Figure 1), generation is a cumbersome process that scales linearly with the amount of pixels in the input image. Therefore we have

	GRAY-SCALED IMAGES 28X28
$h_{:,1}$	1 X CONV F=5X5, K=32, S=2 1 X CONV F=3X3, K=64, S=1
z_1	1 X DENSE D=64 64 FEATURE VECTOR
$h_{:,2}$	1 X CONV F=3X3, K=64, S=2 1 X CONV F=3X3, K=64, S=1
z_2	1 X DENSE D=32 32 FEATURE VECTOR
$h_{:,3}$	1 X CONV F=3X3, K=64, S=2 1 X CONV F=3X3, K=64, S=1
z_3	1 X DENSE D=16 16 FEATURE VECTOR
$h_{:,4}$	1 X CONV F=3X3, K=64, S=2 1 X CONV F=3X3, K=64, S=1
z_4	1 X DENSE D=8 8 FEATURE VECTOR
$h_{:,5}$	1 X CONV F=3X3, K=64, S=2 1 X CONV F=3X3, K=64, S=1
z_5	1 X DENSE D=4 4 FEATURE VECTOR

Table 1: The convolutional layer (Conv), filter size (F), depth (K), stride (S), dense layer (Dense) and dimensionality (D) used in defining FAME for gray-scaled. The architecture is defined such that we ensure dimensionality reduction throughout the hierarchical stochastic layers. The autoregressive decoder is a PixelCNN (van den Oord et al., 2016b) with a mask *A* convolution F=7x7, K=64, S=1 followed by 4 residual blocks of convolutions with mask *B*, F=3x3, K=64, S=1. Finally there are three non-residual layers of convolutions with mask *B* where the last is the output layer with a sigmoid activation function.

	NLL
IWAE (BURDA ET AL., 2015A)	82.90
LVAE (SØNDERBY ET AL., 2016)	81.74
CAGEM (MAALØE ET AL., 2017)	81.60
DVAE (ROLFE, 2017)	80.04
VGP (TRAN ET AL., 2016)	79.88
IAF VAE KINGMA ET AL. (2016)	79.10
VLAE CHEN ET AL. (2017)	78.53
FAME NO CONCATENATION	78.73
FAME	**77.82**

	NLL
DRAW (GREGOR ET AL., 2015)	80.97
DVAE (ROLFE, 2017)	81.01
IAF VAE (KINGMA ET AL., 2016)	79.88
PIXELRNN (VAN DEN OORD ET AL., 2016B)	79.20
VLAE (CHEN ET AL., 2017)	79.03
PIXELVAE (GULRAJANI ET AL., 2016)	**79.02**
FAME	79.30

Table 2: Negative log-likelihood performance on dynamically (left) and statically (right) binarized MNIST in nats. For the dynamically binarized MNIST results show the results for the FAME No Concatenation that has no dependency on the input image. The evidence lower-bound is computed with 5000 importance weighted samples $\mathcal{L}_{5000}(\theta, \phi; x)$.

defined a slightly changed parameterization denoted FAME No Concatenation, where the concatenation with the input is omitted. The generation has no dependency on the input data distribution and can therefore be performed in one forward-pass through the generative model.

For optimization we apply the Adam optimizer (Kingma & Ba, 2014) with a constant learning rate of 0.0003. We use 1 importance weighted sample and temperature (Sønderby et al., 2016) scaling from .3 to 1. during the initial 200 epochs. All models are trained using the same optimization scheme.

3.1 GENERATIVE PERFORMANCE

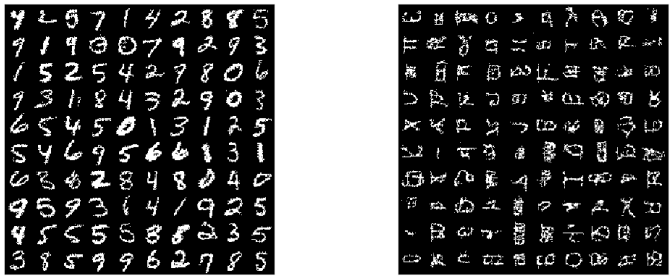
The MNIST dataset serves as a good sanity check and has a myriad of previously published generative modelling benchmarks. We experienced much faster convergence rate on FAME compared to training a regular LVAE. On the dynamically binarized MNIST dataset we see a significant improvement (cf. Table 2). However, on the statically binarized MNIST, the parameterization and current optimization strategy was unsuccessful in achieving state-of-the-art results

	NLL
IWAE (BURDA ET AL., 2015A)	103.38
LVAE (SØNDERBY ET AL., 2016)	102.11
RBM (BURDA ET AL., 2015B)	100.46
DVAE (ROLFE, 2017)	97.43
DRAW (GREGOR ET AL., 2015)	96.50
CONV DRAW (GREGOR ET AL., 2016)	91.00
VLAE CHEN ET AL. (2017)	89.83
FAME	**82.54**

Table 3: Negative log-likelihood performance on OMNIGLOT in nats. The evidence lower-bound is computed with 5000 importance weighted samples $\mathcal{L}_{5000}(\theta, \phi; x)$.

(cf. Table 1). In Figure 2a we see random samples drawn from a $\mathcal{N}(0, I)$ distribution and propagated through the decoder parameters θ. We also trained the FAME No Concatenation which performs nearly on par with the previously state-of-the-art VLAE model (Chen et al., 2017) that in comparison utilizes a skip-connection from the input distribution to the generative decoder: $p_{\text{local}}(x|z) = \prod_i p(x_i|z, x_{\text{WindowAround}(i)})$. This proves that a better parameterization of the VAE improves the performance without the need of tedious autoregressive generation. There was no significant difference in the $KL\big(q(\mathbf{z}|x)||p(\mathbf{z})\big)$ between FAME and FAME No Concatenation. FAME use 10.85 nats in average to encode images, whereas FAME No Concatenation use 12.29 nats.

OMNIGLOT consists of 50 alphabets of handwritten characters, where each character has a limited amount of samples. Each character has high variance which makes it harder to fit a good generative model compared to MNIST. Table 3 presents the negative log-likelihood of FAME for OMNIGLOT and demonstrates significant improvement over previously published state-of-the-art. Figure 2b shows generated samples from the learned θ parameter space.



(a) (b)

Figure 2: Random samples drawn from a $\mathcal{N}(0, I)$ distribution and propagated through the generative model of FAME for the dynamically binarized MNIST (a) and OMNIGLOT (b) dataset.

From Sønderby et al. (2016) we have seen that the LVAE is able to learn a much tighter $\mathcal{L}_1$ ELBO compared to the VAE. For the MNIST experiments, the $\mathcal{L}_1$ ELBO is at 80.11 nats compared to the $\mathcal{L}_{5000}$ 77.82 nats. Similarly the OMNIGLOT $\mathcal{L}_1$ ELBO is 86.62 nats compared to 82.54 nats. This shows significant improvements when using importance weighted samples and indicates that the parameterization of the FAME can be done in a way so that the bound is even tighter. We also find that the top-most latent stochastic layer is not *collapsing* into its prior, since the $KL\big(q(z_5|x)||p(z_5)\big)$ is 5.04 nats for MNIST and 3.67 nats for OMNIGLOT.

In order to analyze the contribution from the autoregressive decoder we experimented on masking the contribution from either the concatenated image or the output of the FAME decoder before feeding it into the PixelCNN layers (cf. Figure 1). In Figure 3a we see the results of reconstructing MNIST images when masking out the contribution from the stochastic variables and in Figure 3b we mask out the contribution from the concatenated input image.

(a) (b)

Figure 3: MNIST reconstructions when masking the output from the FAME stochastic variables (a) and the concatenated input image (b) prior to feeding them to the autoregressive PixelCNN. It is interesting to see how the edge information comes from the autoregressive dependency on the input image.

4 CONCLUSION

We have presented FAME, an extension to the VAE that significantly improve state-of-the-art performance on standard benchmark datasets. By introducing feature map representations in the deterministic layers in addition to top-down inference we have shown that the model is able to capture representations of image distributions while utilizing a powerful autoregressive architecture as a decoder.

In order to analyze the contribution from the VAE as opposed to the autoregressive model, we have presented results without concatenating the input image when reconstructing and generating. This parameterization shows on par results with the previously state-of-the-art results without depending on the time-consuming autoregressive generation.

Further directions for FAME is to (i) test it on natural image datasets with convolutions in the stochastic layers, (ii) expand the model to capture other data modalities such as audio and text, (iii) combine the model in a semi-supervised framework.

REFERENCES

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8), 2013.

S.R. Bowman, L. Vilnis, O. Vinyals, A.M. Dai, R. Jozefowicz, and S. Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance Weighted Autoencoders. *arXiv preprint arXiv:1509.00519*, 2015a.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Accurate and conservative estimates of mrf log-likelihood using reverse annealing. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2015b.

Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational Lossy Autoencoder. In *International Conference on Learning Representations*, 2017.

Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. In *Advances in Neural Information Processing Systems*. 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*. 2014.

Karol Gregor, Ivo Danihelka, Alex Graves, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

Karol Gregor, Rezende Danilo Jimenez Besse, Fredric, Ivo Danihelka, and Daan Wierstra. Towards conceptual compression. *arXiv preprint arXiv:1604.08772*, 2016.

Ishaan Gulrajani, Kundan Kumar, Faruk Ahmed, Adrien Ali Taiga, Francesco Visin, David Vazquez, and Aaron Courville. PixelVAE: A latent variable model for natural images. *arXiv e-prints*, 1611.05013, November 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of International Conference on Machine Learning*, 2015.

Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 12 2014.

Diederik P. Kingma, Danilo Jimenez Rezende, Shakir Mohamed, and Max Welling. Semi-Supervised Learning with Deep Generative Models. In *Proceedings of the International Conference on Machine Learning*, 2014.

Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*. 2016.

Max Kingma, Diederik P; Welling. Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*, 12 2013.

Rahul G Krishnan, Uri Shalit, and David Sontag. Deep Kalman filters. *arXiv:1511.05121*, 2015.

Brenden M Lake, Ruslan R Salakhutdinov, and Josh Tenenbaum. One-shot learning by inverting a compositional causal process. In *Advances in Neural Information Processing Systems*. 2013.

Yann LeCun, Lon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2278–2324, 1998.

Lars Maaløe, Casper K. Sønderby, Søren K. Sønderby, and Ole Winther. Auxiliary Deep Generative Models. In *Proceedings of the International Conference on Machine Learning*, 2016.

Lars Maaløe, Marco Fraccaro, and Ole Winther. Semi-supervised generation with cluster-aware generative models. *arXiv preprint arXiv:1704.00637*, 2017.

Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, 2015.

Danilo J. Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *arXiv preprint arXiv:1401.4082*, 04 2014.

Danilo Jimenez Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. In *Proceedings of the International Conference on Machine Learning*, 2015.

Jason Tyler Rolfe. Discrete variational autoencoders. In *Proceedings of the International Conference on Learning Representations*, 2017.

Tim Salimans, Andrej Karparthy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint:1701.05517, 2017*, 2017.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *Advances in Neural Information Processing Systems 29*. 2016.

Dustin Tran, Rajesh Ranganath, and David M Blei. Variational Gaussian process. In *Proceedings of the International Conference on Learning Representations*, 2016.

Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. *arXiv preprint arXiv:1606.05328*, 2016a.

Aaron van den Oord, Kalchbrenner Nal, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 01 2016b.

Bibliography

Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; S. Corrado, G.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Jozefowicz, R.; Kaiser, L.; Kudlur, M.; Levenberg, J.; Mané, D.; Monga, R.; Moore, S.; Murray, D.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Talwar, K.; Tucker, P.; Vanhoucke, V.; Vasudevan, V.; Viégas, F.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Agakov, F. V.. and Barber, D. An Auxiliary Variational Method. In *Neural Information Processing*, volume 3316 of *Lecture Notes in Computer Science*, pages 561–566. Springer Berlin Heidelberg, 2004.

Ali, M. H.; Rabhi, A.; H., A. E., and Tina, G. M. Real time fault detection in photovoltaic systems. *Procedia Energy*, pages 914–923, 2017.

Alzahrani, Ahmad; Shamsi, Pourya; Dagli, Cihan, and Ferdowsi, Mehdi. Solar irradiance forecasting using deep neural networks. *Procedia Computer Science*, pages 304–313, November 2017.

Amodei, D.; Ananthanarayanan, S.; Anubhai, R.; Bai, J.; Battenberg, E.; Case, C.; Casper, J.; Catanzaro, B.; Cheng, Q.; Chen, G.; Chen, J.; Chen, J.; Chen, Z.; Chrzanowski, M.; Coates, A.; Diamos, G.; Ding, K.; Du, N.; Elsen, E.; Engel, J.; Fang, W.; Fan, L.; Fougner, C.; Gao, L.; Gong, C.; Hannun, A.; Han, T.; Johannes, L. V.; Jiang, B.; Ju, C.; Jun, B.; LeGresley, P.; Lin, L.; Liu, J.; Liu, Y.; Li, W; Li, X.; Ma, D.; Narang, S.; Ng, A.; Ozair, S.; Peng, Y.; Prenger, R.; Qian, S.; Quan, Z.; Raiman, J.; Rao, V.; Satheesh, S.; Seetapun, D.; Sengupta, S.; Srinet, K.; Sriram, A.; Tang, H.; Tang, L.; Wang, C.; Wang,

J.; Wang, K.; Wang, Y.; Wang, Z.; Wang, Z.; Wu, S.; Wei, L.; Xiao, B.; Xie, W.; Xie, Y.; Yogatama, D.; Yuan, B.; Zhan, J., and Zhu, Z. Deep speech 2: End-to-end speech recognition in english and mandarin. In *Proceedings of the International Conference on Machine Learning*, pages 173–182, 2016.

Bach-Andersen, M. *A Diagnostic and Predictive Framework for Wind Turbine Drive Train Monitoring*. PhD thesis, Technical University of Denmark, 2017.

Bach-Andersen, Martin; Rømer-Odgaard, Bo, and Winther, Ole. Deep learning for automated drivetrain fault detection. *Wind Energy*, 21:29–41, October 2017.

Bastien, Frédéric; Lamblin, Pascal; Pascanu, Razvan; Bergstra, James; Goodfellow, Ian J.; Bergeron, Arnaud; Bouchard, Nicolas, and Bengio, Yoshua. Theano: new features and speed improvements. In *Deep Learning and Unsupervised Feature Learning, workshop at Neural Information Processing Systems*, 2012.

Basu, S.; Banerjee, A., and Mooney, R. J. Semi-supervised clustering by seeding. In *Proceedings of the International Conference on Machine Learning*, 2002.

Bengio, Y.; Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8), 2013.

Bishop, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006.

Blundell, C.; Cornebise, J.; Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural networks. In *Proceedings of the International Conference on Machine Learning*, pages 1613–1622, 2015.

Bowman, S. R.; Vilnis, L.; Vinyals, O.; Dai, A. M.; Jozefowicz, R., and Bengio, S. Generating sentences from a continuous space. In *Proceedings of the Conference on Computational Natural Language Learning*, pages 10–21, 2016.

Burda, Y.; Grosse, R., and Salakhutdinov, R. Importance Weighted Autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Carlini, N. and Wagner, D. Audio Adversarial Examples: Targeted Attacks on Speech-to-Text. *arXiv preprint arXiv:1801.01944*, 2017.

Chapelle, O.; Schlkopf, B., and Zien, A. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010.

Chen, X.; Kingma, D. P.; Salimans, T.; Duan, Y.; Dhariwal, P.; Schulman, J.; Sutskever, I., and Abbeel, P. Variational Lossy Autoencoder. In *International Conference on Learning Representations*, 2017.

Dayan, P. and Hinton, G. E. Varieties of helmholtz machine. *Neural Networks*, 9(8):1385–1403, November 1996.

Deng, J.; Dong, W.; Socher, R.; Li, L. J.; Li, K., and Fei-fei, L. Imagenet: A large-scale hierarchical image database. In *CVPR09*, 2009.

Dieleman, S.; Schlüter, J.; Raffel, C.; Olson, E.; Sønderby, S. K.; Nouri, D.; van den Oord, A., and Battenberg, E. and. Lasagne: First release., August 2015.

Fraccaro, M.; Sønderby, S. Kaae; Paquet, U., and Winther, O. Sequential neural models with stochastic layers. In *Advances in Neural Information Processing Systems*. 2016.

Gal, Yarin. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

Glorot, X.; Bordes, A., and Bengio, Y. Deep sparse rectifier neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2011.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems*. 2014.

Goodfellow, I.; Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016.

Gulrajani, I.; Kumar, K.; Ahmed, F.; Taiga, A. A.; Visin, F.; Vazquez, D., and Courville, A. PixelVAE: A latent variable model for natural images. *arXiv e-prints*, 1611.05013, November 2016.

Gumbel, E. J. Statistical theory of extreme values and some practical applications: a series of lectures. *Number 33. US Govt. Print. Office*, 1954.

He, K.; Zhang, X.; Ren, S., and Sun, J. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

He, K.; Zhang, X.; Ren, S., and Sun, J. Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

Hinton, G. E. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, August 2002.

Hinton, G. E. and Salakhutdinov, R. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 2006.

Hinton, G. E. and Zemel, R. S. Autoencoders, minimum description length and helmholtz free energy. In *Advances in Neural Information Processing Systems*, pages 3–10, 1993.

Hinton, G. E.; Osindero, S., and Teh, Y. W. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November 1997.

Huang, G.; Liu, Z., and Weinberger, K. Q. Densely connected convolutional networks. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2261–2269, 2017.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of International Conference on Machine Learning*, 2015.

Jang, E.; Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Jiang, L. L. and Maskell, D. L. Automatic fault detection and diagnosis for photovoltaic systems using combined artificial neural network and analytical based methods. In *Proceedings of the IEEE International Joint Conference on Neural Networks*. IEEE Computer Society, 2015.

Jordan, M. I.; Ghahramani, Z.; Jaakkola, T. S., and Saul, L. K. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, November 1999.

Kalchbrenner, N. and Blunsom, P. Recurrent continuous translation models. 2013.

King, D. L.; Quintana, M. A.; Kratochvil, J. A.; Ellibee, D. E., and Hansen, B. R. Photovoltaic module performance and durability following long-term field exposure. *Progress in Photovoltaics: Research and Applications*, 8(2): 241–256, 2000.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 12 2014.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations*, 2014.

Kingma, D. P.; Rezende, D. J.; Mohamed, S., and Welling, M. Semi-Supervised Learning with Deep Generative Models. In *Proceedings of the International Conference on Machine Learning*, 2014.

Kingma, D. P.; Salimans, T.; Jozefowicz, R.; Chen, X.; Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*. 2016.

Köntges, M.; Kurtz, S.; Packard, C.; Jahn, U.; Berger, K. A.; Kato, K.; Friesen, T.; Liu, H., and Van Iseghem, M. Review of failures of photovoltaic modules. Report, International Energy Agency, March 2014 2014.

Krizhevsky, A.; Ilya, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105. 2012.

Lake, Brenden M; Salakhutdinov, Ruslan R, and Tenenbaum, Josh. One-shot learning by inverting a compositional causal process. In *Advances in Neural Information Processing Systems*. 2013.

Larsen, A. B. L.; Sønderby, S. K.; Larochelle, H., and Winther, O. Autoencoding beyond pixels using a learned similarity metric. In *Proceedings of the International Conference on Machine Learning*, pages 1558–1566, 2016.

Laukamp, H.; Schoen, T., and Ruoss, D. Reliability study of grid connected pv systems, field experience and recommended design practice. Report, International Energy Agency, 2002.

LeCun, Y.; Bottou, L.; Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2278–2324, 1998.

LeCun, Y.; Haffner, P.; Bottou, L., and Bengio, Y. Object recognition with gradient-based learning. In *Shape, Contour and Grouping in Computer Vision*, pages 319–, London, UK, 1999. Springer-Verlag.

LeCun, Y.; Huang, F. J., and Bottou, L. Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 97–104. IEEE Computer Society, 2004.

LeCun, Y.; Bengio, Y., and Hinton, G. E. Deep learning. *Nature*, 521(7553): 436–444, 2015.

Maaløe, L. and Winther, O. Feature map variational auto-encoders. In *To be submitted*, 2018.

Maaløe, L.; Arngren, M., and Winther, O. Deep belief nets for topic modeling. In *International Conference on Machine Learning Workshop on Knowledge-Powered Deep Learning for Text Mining*, 2015a.

Maaløe, L.; Sønderby, S. K.; Sønderby, C. K., and Winther, O. Improving semi-supervised learning with auxiliary deep generative models. In *Neural Information Processin Systems Workshop on Approximate Bayesian Inference*, 2015b.

Maaløe, L.; Sønderby, C. K.; Sønderby, S. K., and Winther, O. Auxiliary Deep Generative Models. In *Proceedings of the International Conference on Machine Learning*, 2016.

Maaløe, L.; Fraccaro, M., and Winther, O. Semi-supervised generation with cluster-aware generative models. In *Neural Information Processin Systems Workshop on Approximate Bayesian Inference*, 2017.

Maaløe, L.; Spataru, S. V.; Sera, D., and Winther, O. Condition monitoring in photovoltaic systems by semi-supervised machine learning. In *Submitted to IEEE Transactions of Industrial Informatics*, 2018.

Miyato, Takeru; Maeda, Shin-ichi; Koyama, Masanori; Nakae, Ken, and Ishii, Shin. Distributional Smoothing with Virtual Adversarial Training. *arXiv preprint arXiv:1507.00677*, 7 2015.

Mnih, A. and Gregor, K. Neural variational inference and learning in belief networks. In *Proceedings of the International Conference on Machine Learning*, pages 1791–1799, 2014.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

Murphy, K. P. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. 2011.

Nguyen, A. M.; Yosinski, J., and Clune, J. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436. IEEE Computer Society, 2015.

Ranganath, R.; Tran, D., and Blei, D. M. Hierarchical Variational Models. In *Proceedings of the International Conference on Machine Learning*, 2016.

Ranzato, M. A. and Szummer, M. Semi-supervised Learning of Compact Document Representations with Deep Networks. In *Proceedings of the International Conference on Machine Learning*, pages 792–799, 2008.

Rasmus, A.; Berglund, M.; Honkala, M.; Valpola, H., and Raiko, T. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, 2015.

Rezende, D. J. and Mohamed, S. Variational Inference with Normalizing Flows. In *Proceedings of the International Conference on Machine Learning*, 2015.

Rezende, D. J.; Mohamed, S., and Wierstra, D. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *arXiv preprint arXiv:1401.4082*, 2014.

Rolfe, J. T. Discrete variational autoencoders. In *Proceedings of the International Conference on Learning Representations*, 2017.

Ronneberger, O.; Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. *arXiv preprint arXiv:1505.04597*, 2015.

Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386, 1958.

Rumelhart, D. E.; Hinton, G. E., and Williams, R. J. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

Salakhutdinov, R. and Hinton, G. E. Semantic hashing. *International Journal Approximate Reasoning*, 50(7):969–978, July 2009.

Salakhutdinov, R. and Larochelle, H. Efficient learning of deep boltzmann machines. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 693–700, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; Chen, X., and Chen, X. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*. 2016.

Salimans, T.; Karparthy, A.; Chen, X., and Kingma, D. P. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint:1701.05517, 2017*, 2017.

Schmidhuber, J. Deep learning in neural networks: An overview. *arXiv preprint arXiv:1404.7828*, 2014.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T.; Leach, M.; Kavukcuoglu, K.; Graepel, T., and Hassabis, D. Mastering the

game of Go with deep neural networks and tree search. *Nature*, 529(7587): 484–489, jan 2016.

Silvestre, S.; Aissa, C., and Karatepe, E. Automatic fault detection in grid connected pv systems. 94:119–127, 06 2013.

Sønderby, C. K.; Raiko, T.; Maaløe, L.; Sønderby, S. K., and Winther, O. Ladder variational autoencoders. In *Advances in Neural Information Processing Systems*. 2016.

Spataru, S. V.; Gavriluta, A.; Sera, D.; Maaløe, L., and Winther, O. Automatic fault detection and diagnosis for photovoltaic systems using combined artificial neural network and analytical based methods. In *Proceedings of the IEEE Energy Conversion Congress and Exposition*. IEEE Computer Society, 2016.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, January 2014. ISSN 1532-4435.

Srivastava, R. K.; Greff, K., and Schmidhuber, J. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

Sutskever, I.; Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

Tenenbaum, J. B.; Griffiths, T. L., and Kemp, C. Theory-based Bayesian models of inductive learning and reasoning. *Trends in cognitive sciences*, 10(7):309–318, July 2006.

Tran, D.; Ranganath, R., and Blei, D. M. Variational Gaussian process. In *Proceedings of the International Conference on Learning Representations*, 2016.

Vahdat, A.; Macready, W. G.; Zhengbing, B., and Khoshaman, A. Dvae++: Discrete variational autoencoders with overlapping transformations. *arXiv preprint arXiv:1802.04920*, 2018.

van den Oord, A.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, N.; Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016a.

van den Oord, A.; Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 01 2016b.

van den Oord, A.; Kalchbrenner, N.; Vinyals, O.; Espeholt, L.; Graves, A., and Kavukcuoglu, K. Conditional image generation with pixelcnn decoders. *arXiv preprint arXiv:1606.05328*, 2016c.

Vincent, P.; Larochelle, H.; Bengio, Y., and Manzagol, P. A. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the International Conference on Machine Learning*, pages 1096–1103, 2008.

Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y., and Manzagol, P. A. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11: 3371–3408, December 2010.

Xiong, W.; Wu, L.; Alleva, F.; Droppo, J.; Huang, X., and Stolcke, A. The microsoft 2017 conversational speech recognition system. *arXiv preprint arXiv:1708.06073*, 2017.

Xu, F. and Tenenbaum, J. B. Word learning as Bayesian inference. *Psychological Review*, 114(2):245–272, apr 2007.

Yang, B. B.; Sorensen, N. R.; Burton, P. D.; Taylor, J. M.; Kilgo, A. C.; Robinson, D. G., and Granata, J. E. Reliability model development for photovoltaic connector lifetime prediction capabilities. In *39th IEEE Photovoltaic Specialists Conference (PVSC)*, pages 0139–0144, 2013.