Towards deeper understanding of semi-supervised learning with variational autoencoders

Karol Jurasiński

Delft University of Technology

TU Delft

Towards deeper understanding of semi-supervised learning with variational autoencoders

by

Karol Jurasiński

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday September 27, 2019 at 14:00.

Student number:	4748069	
Thesis committee:	Prof. dr. H. Hung,	TU Delft
	Prof. dr. ir. M. Loog	TU Delft, supervisor
	Prof. dr. ir. S. Verwer,	TU Delft
	T. Viering,	TU Delft, daily supervisor

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Preface

This report is the final result of my research work about semi-supervised learning with deep generative models, in the Pattern Recognition research group at the Delft University of Technology. The first part includes the main results, and for more extensive background information please refer to the supplementary material at the end.

I would like to thank Marco Loog and Tom Viering for the guidance during the entire thesis. Having realised how valuable the discussions we had were, I can only wish for such great supervisors in the future. Thank you for providing me with the mathematical insights and intuition, but also for the critical questions, support, discussions about the research community in general, and for helping me become better at research. I would also like to thank my friends for being there throughout the year, with a special mention to Ola, Daniel, Barbara and Nirmal. If not for your support, I wouldn't be able to finish this project.

Karol Jurasiński
Delft, September 2019

Abstract

Recently, deep generative models have been shown to achieve state-of-the-art performance on semi-supervised learning tasks. In particular, variational autoencoders have been adopted to use labeled data, which allowed the development of SSL models with the usage of deep neural networks. However, some of these models rely on ad-hoc loss additions for training, and have constraints on the latent space, which effectively prevent the use of recent developments in improving the posterior approximations.

In this paper, we analyse the limitations of semi-supervised deep generative models based on VAEs, and show that it is possible to drop the assumptions made on the latent space. We present a simplified method for semi-supervised learning which combines the discriminative and generative loss in a principled manner. Our model allows for straightforward application of normalizing flows and achieves competitive results in semi-supervised classification tasks.

1 Introduction

Semi-supervised learning (SSL) considers the problem of classification where only a limited number of the observations have the corresponding class labels, and the remaining data does not have labels. Such problems are common in practice, as it is often relatively simple to obtain data, but expensive to label it. One example is fake news detection [9], where labeling data involves difficult and time-consuming manual labour, while obtaining the data can be automatic and is therefore almost free. Semi-supervised learning has a wide range of applications, ranging from the aforementioned fake-news detection [9] and cancer prediction [29], through speech recognition [26], to image segmentation [22].

Semi-supervised methods have been extensively studied for many years [4], and classical examples include generative mixture models, self-learning [1] or tSVM [28]. Recently, due to the abundance of the data and the decreasing cost of computing power, many approaches on SSL for classification have been adjusted to use deep models.

In this work, we focus on the approach developed by Kingma et al. [13], which uses a deep generative model to perform classification, therefore considering the SSL problem as a missing data imputation task [16]. The approach is based on *variational autoencoder* (VAE) [12], a model whose goal is to find a lower-dimensional representation of the dataset (latent space). Kingma et al. [13] expanded the VAE by assuming an extra latent variable which is corresponding to the labels and introducing an additional discriminative loss, thus making it suitable for semi-supervised learning. Maaløe et al. [18] later improved upon that work by adding an auxiliary variable, which allowed the model to fit more complex latent distributions.

We analyse the SSL model developed by Kingma et al. [13], and show that the information meant to be contained in the extra latent variable is also captured in the original part of the latent space. We reason that it is possible to perform classification on such latent space without any further assumptions. Moreover, we show that the performance of the model is closely related to reconstruction error. We also demonstrate that the model may converge to a solution in which the classifier does not perform any meaningful task, even when trained with supplementary discriminative loss as suggested by Kingma et al. [13]. These observations lead us to hypothesize that assumptions introduced by Kingma et al. [13] can be lifted, while not degrading the performance of the model. We introduce a modified model, which optimises the *evidence lower bound* (ELBO) directly, and encompasses a more principled discriminative loss. The proposed model achieves classification accuracy of 92.3% which is comparable to that of Kingma's, and shows that it is the conditional generative setup used in both models that is the main contributor to their success, similar to mixture models [30].

1.1 Outline

In Section 2, we begin by establishing notation and introducing the necessary background material. Section 3 contains our analysis of the model proposed by Kingma et al. [13], exposing underlying assumptions and motivating the contribution. In Section 4 we state the pursued research questions and propose a new model. In Section 5 we evaluate the proposed model and analyse the experimental results. Finally, in Section 6 we reflect on the findings, provide additional insights and propose a further research direction. Appendix A contains proofs for our propositions. Hyperparameters used for all the experiments are listed in Appendix B, unless explicitly stated otherwise.

2 Preliminaries

2.1 Semi-supervised learning

Supervised learning considers the problem of finding the best prediction model $f : \mathbf{x} \to y$ that maps $\mathbf{x}$ to y, where $\mathbf{x}$ is the random sample from an unknown distribution (such as images, sequences, sets of features etc.) and y is the associated class label. The model can be found using traditional supervised learning methods by using the set of *labeled* data tuples $\mathbf{L} = (X_l, \mathrm{Y}) = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_l, y_l)\}$. In semi-supervised learning, the same problem is considered, but with an extra set of *unlabeled* data, $\mathbf{U} = (X_u) = \{\mathbf{x}_{l+1}, ..., \mathbf{x}_{l+u}\}$. This set does not contain the labels, but can be used to extract the underlying structure of the inputs $\mathbf{x}$.

A common approach to this problem is maximum weighted likelihood estimation [5], where the objective is the following:

$$\arg\max_{\Theta}(\log p(\{\mathbf{x}_i, y_i\}_{i=1}^{l}|\theta) + \lambda \cdot \log p(\{\mathbf{x}_i\}_{i=l+1}^{l+u}|\theta)),$$

where λ is a hyper-parameter that controls the relative weight of the labeled and unlabeled likelihood function.

2.2 Variational autoencoders

Variational autoencoders are a type of latent variable models of the form $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$, where $\mathbf{z}$ is the latent, $p(\mathbf{x}|\mathbf{z})$ is the posterior distribution, and $p(\mathbf{z})$ is the prior. VAEs are typically trained by maximizing the *evidence lower bound* (ELBO) objective,

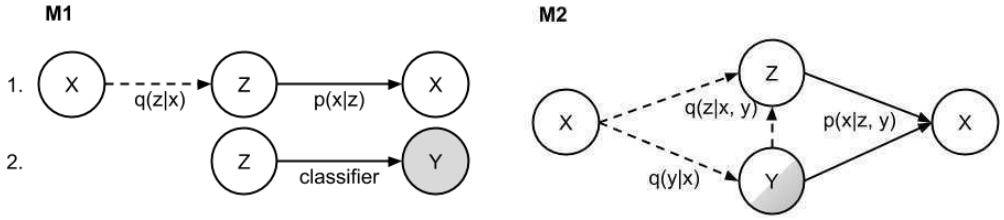
$$\log p(\mathbf{x}) \geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z}) + \log p(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})] = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] + \mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})], \quad (1)$$

introducing an approximation to the posterior distribution $q_\phi(\mathbf{z}|\mathbf{x})$. Posterior approximation $q_\phi(\mathbf{z}|\mathbf{x})$ can also be called an *inference model* or an *encoder* - it infers the hidden representation of the data. $p_\theta(\mathbf{x}|\mathbf{z})$ is the corresponding *generative model* or *decoder*, which decodes the data from the latent. The goal of variational autoencoders is usually the inference of the latent variables. The evidence lower bound is normally optimised using the reparametrization trick [12].

The loss of variational autoencoders can be interpreted as the sum of reconstruction error (first term in Equation 1) and Kullback-Leibler divergence between the approximate posterior and the prior distributions (second term in Equation 1). One of the common choices for the family of output distributions for non-binary data is multivariate Gaussian distribution, $p_\theta(\mathbf{x}|\mathbf{z}) \sim \mathcal{N}(\mathbf{x}|f(\mathbf{z};\theta), \sigma^2 I)$, with σ a hyper-parameter. This is the choice we use in this work. The mean of the normal distribution $f(\mathbf{z};\theta)$ is parameterized by neural networks with parameters θ. When choosing the normal distribution as output distribution, the first term in the ELBO is given by [7]:

$$\log p_\theta(\mathbf{x}|\mathbf{z}) = C - \frac{1}{2\sigma^2}\|\mathbf{x} - f(\mathbf{z};\theta)\|_2^2, \quad (2)$$

where C is a constant independent of σ. Notice that $\frac{1}{\sigma}$ acts as a parameter that weighs the relative cost of the reconstruction error compared to the Kullback-Leibler divergence, as appearing in the loss of variational autoencoder. KL divergence is often recognized in this context as a regularization term [12], and σ can be interpreted as the relative weight between the reconstruction error and the regularizer. A similar idea of weighing the two terms is also presented by Higgins et al. [10]. We show that the weighing of the two losses happens to be crucial to the success of the semi-supervised



(a) (b)

Figure 1: Probabilistic graphical models of M1 and M2 [13]. The dashed lines indicate inference models (encoders), the solid lines indicate the generative models (decoders). Grey background on elements y corresponds to operations only on observed labels, and split background indicates the semi-supervised setup.

models introduced by [13]. The details are discussed in Subsection 3.1. Note that q_ϕ and p_θ depend on parameters ϕ and θ. In order to simplify notation in further sections, we omit the parameter subscripts.

2.3 Semi-supervised learning with deep generative models

Kingma et al. [13] introduced two approaches to semi-supervised learning by using the variational autoencoder, which can be seen in Figure 1. The first one (called M1) consists of two steps: initially, a variational autoencoder is trained to extract a lower dimensional latent feature $\mathbf{z}$, using both labeled and unlabeled data. As a second step, a classifier is trained on the features extracted by the variational autoencoder in either fully supervised or semi-supervised setting.

The second model from the same authors (called M2) extends the variational autoencoder with labels, in order to use it for semi-supervised classification. The authors assume that the labels are part of the underlying latent space from which the dataset is generated, i.e. $\mathbf{z} = [y, \mathbf{z}']$[1]. Then, just like in the VAE, a posterior approximation $q(\mathbf{z}|\mathbf{x})$ is introduced, which is factorized as $q(\mathbf{z}|\mathbf{x}) = q(\mathbf{z}'|\mathbf{x}, y)q(y|\mathbf{x})$. This is done so that $q(y|\mathbf{x})$ can be used for classification during test time by inferring the class labels using this distribution. While there are no restrictions on the posterior approximation $q(\mathbf{z}'|\mathbf{x}, y)$, the posterior $q(y|\mathbf{x})$ is assumed to be in the categorical family in order to make the classification possible, $q(y|\mathbf{x}) = \text{Cat}(y|f(\mathbf{x}))$. Class specific information can be modeled through y, and $\mathbf{z}'$ is meant to capture unspecified factors of variation, which are dependent on the domain. In the example of written digits (MNIST [15]), one of the unspecified factors could be the writing style. Now, because of the semi-supervised setup, the class labels are sometimes given, and in that case the inference process for them is not necessary. The given labels can be used in the generative process immediately. This results in a different ELBO for labeled data, $\mathcal{L}(\mathbf{x}, y)$, and

[1]Note that in Kingma et al. [13] the authors denote the latent space as $\mathbf{z}$ and y. Here we use $\mathbf{z}'$ and y to indicate the difference from the latent space of traditional VAE, $\mathbf{z}$.

Figure 2: Images from MNIST dataset, along with their synthetic analogies generated by M2 [13] with the same inferred style (z), but with a changed label (y). In the left column are the original images, and each following column corresponds to analogies with labels 0-9. Note how the reconstructions on the diagonal match the original image the best.

unlabeled data, $\mathcal{U}(\mathbf{x})$ [13]:

$$\begin{aligned} \log p(\mathbf{x}, y) &\geq \mathbb{E}_{q(\mathbf{z}'|\mathbf{x},y)}[\log p(\mathbf{x}|\mathbf{z}', y) + \log p(y) + \log p(\mathbf{z}') - \log q(\mathbf{z}'|\mathbf{x}, y)] = \mathcal{L}(\mathbf{x}, y) \\ \log p(\mathbf{x}) &\geq \mathbb{E}_{q(\mathbf{z}'|\mathbf{x},y)}[\log p(\mathbf{x}|\mathbf{z}', y) + \log p(y) + \log p(\mathbf{z}') - \log q(\mathbf{z}', y|\mathbf{x})] \\ &= \sum_y q(y|\mathbf{x})(\mathcal{L}(\mathbf{x}, y)) + \mathbb{E}_{q(y|\mathbf{x})}[-\log q(y|\mathbf{x})] = \mathcal{U}(\mathbf{x}) \end{aligned} \tag{3}$$

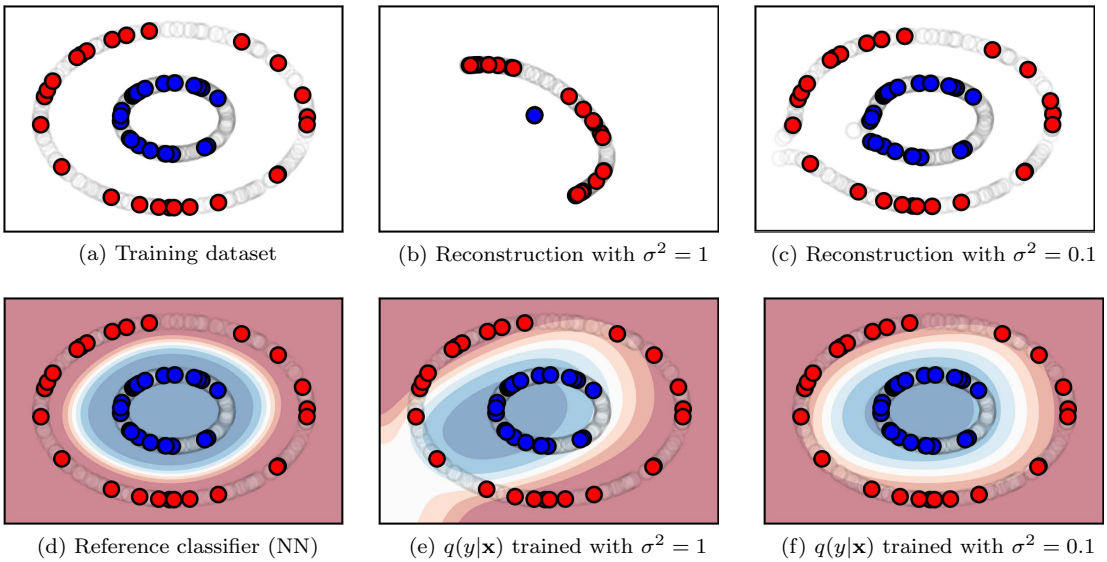
The authors also note that the discriminative term $q(y|\mathbf{x})$ which is used for classification appears only in the unlabeled loss, and introduce an explicit classification loss in order to help with training. This results in the objective of:

$$\mathcal{J}^\alpha = \mathcal{L}(\{\mathbf{x}, y\}_{i=1}^l) + \mathcal{U}(\{\mathbf{x}\}_{i=l+1}^{l+u}) + \alpha \cdot \mathbb{E}_{\widetilde{p}_l}[\log q(y|\mathbf{x})]$$

where α is a hyper-parameter controlling the relative weight of discriminative and generative learning.

3 Motivating observations

In Figure 2, we show the result of the model M2 trained on the MNIST dataset, with limited number of labeled samples (100). In the figure, we can see analogies generated by fixing the inferred $\mathbf{z}'$ part of the latent space for a given digit, and varying the class y. We can see that the model perfectly reconstructs the original digits (as seen on the diagonal), while the analogies are less sharp. Furthermore, the model treats the latent space according to our intuition, i.e. behaves like a conditional generative model - freezing inferred z and varying y for generation produces analogies of digits with the same style. When analysing the loss of M2, we can see in $\mathcal{U}(\mathbf{x})$ (Equation 3) that the classifier, $q(y|\mathbf{x})$, weighs the labeled loss when evaluated with different y (with a regularization term). Therefore, this loss is minimized when the classifier outputs the label corresponding to the



(a) Training dataset (b) Reconstruction with $\sigma^2 = 1$ (c) Reconstruction with $\sigma^2 = 0.1$

(d) Reference classifier (NN) (e) $q(y|\mathbf{x})$ trained with $\sigma^2 = 1$ (f) $q(y|\mathbf{x})$ trained with $\sigma^2 = 0.1$

Figure 3: Comparison of classifiers resulting from different settings. Red and blue points indicate data points and reconstructions from different classes. Grey points indicate unlabeled data points. The background is colored with respect to classification confidence. σ^2 can be interpreted as the trade-off parameter between reconstruction and regularization. Note that in (b) all the blue points are concentrated in one point.

best reconstruction. Intuitively, this works because the model achieves the best reconstruction when the label is correct, and so this loss is aligned with the more traditional discriminative loss - it is minimized when the classifier outputs the true label. However, this formulation introduces two implicit assumptions, namely:

1. the model is able to do reconstruction,
2. the model behaves like a conditional generative model, i.e. interprets y as the label and uses it for generation.

In the following subsections, we analyse each of the assumptions, showing that invalidating either of these leads to a classifier that performs worse than the supervised baseline. Note that these observations are mostly related to the generative loss, and the loss of M2 also contains a discriminative term weighed by parameter α. It is therefore possible that when this parameter becomes large enough, the entire model regresses into a purely discriminative setup, in which case the following observations might not hold.

3.1 Generative loss of M2 needs good reconstruction

Based on the choice of the output distribution parameters as discussed in Subsection 2.2 (σ^2 in Equation 2, or the choice of β [10]), the behaviour of the M2 model changes. In Figure 3, we show a toy dataset (concentric circles from [23], Figure 3a) on which we show the behaviour of the model. The dataset consists of 50 labeled (25 from each class), and 9950 unlabeled data points. For this experiment, we use a multi-layer perceptron (MLP) with 1 hidden layer of 50 units as a classifier,

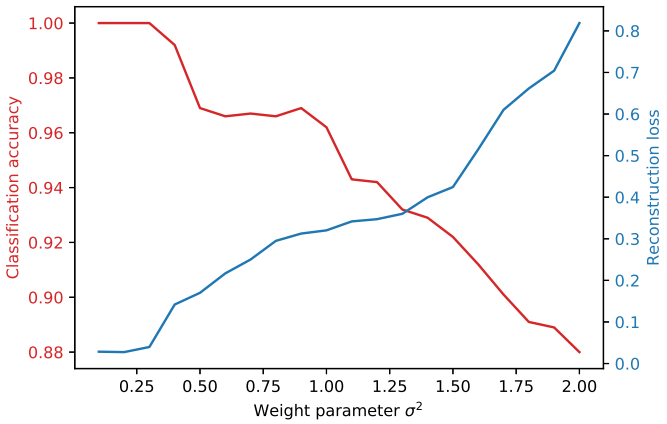


Figure 4: Reconstruction loss (blue) and classification accuracy (red) of M2 model trained with different weights σ^2. The experiment was performed with $\alpha = 0.1$.

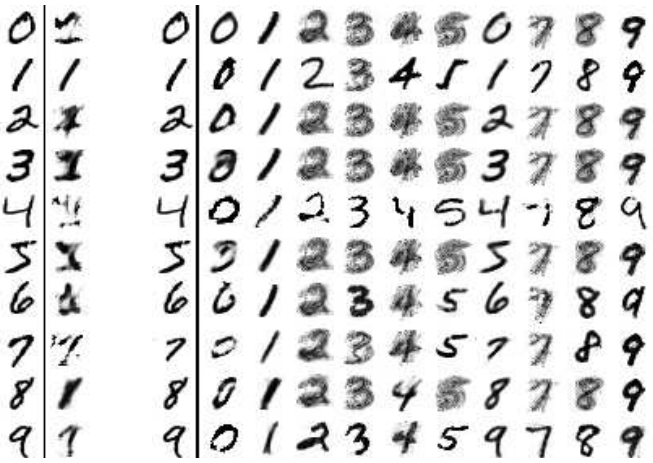
and the encoder and decoder is an MLP with 3 hidden layers of 50 units each. The latent space z' is 1-dimensional.

In Figure 3d we show the result of a neural network that was trained on the labeled data points only, as a baseline. This classifier finds the optimal classification boundary, which shows that such a simple 1 layer network is powerful enough to model the data well. In Figure 3e, the same neural network is used as a classifier, this time trained based on the M2 model loss in the semi-supervised variational autoencoder setting, with a common choice of $\sigma^2 = 1$. The model fails to reconstruct the data well (as seen in Figure 3b) due to overregularization, which further leads to a less-than-optimal classification boundary. This can be fixed with re-weighing the reconstruction loss compared to the Kullback-Leibler divergence by tuning σ^2 as shown in Figure 3c and Figure 3f. In Figure 4, we show the classification accuracy and reconstruction loss with different weighing parameters σ^2.

The experiment for the case shown in Figure 3b and Figure 3e was also performed with a classifier initialized to optimal values, in order to rule out the effects of bad initialization for the setup. Even in this case, the model was drawn away from the optimal solution by the generative loss, resulting in a solution similar to the one shown in Figure 3e. What is more, the model did not stay in the optimal solution even though the training was done with the discriminative loss. This was only fixed in further experiments when the relative weight of discriminative loss to the generative loss (α) was sufficiently large. These experiments show that the reconstruction performance plays a crucial role in utilizing the generative loss for the optimal classifier.

3.2 Pretraining encoder-decoder without classifier makes the model fail

One of the scenarios where M2 does not work is when the latent variable $\mathbf{z}$ is ignored. It has been noted previously that variational autoencoders do not necessarily behave like classical autoencoders [7], and do not autoencode in general [6]. This means that there is a possibility that $p_\theta(\mathbf{x}|\mathbf{z})$ converges to the true $p(\mathbf{x})$ while ignoring the latent space $\mathbf{z}$. In that case the decoder $q_\phi(\mathbf{z}|\mathbf{x})$ simply becomes the same as the prior, $p(\mathbf{z})$, due to KL divergence term in the loss. It is clear that in such a situation, the posterior approximation does not encode any information, and so does not serve



(a) (b)

Figure 5: Illustration of the scenario in which the classifier from M2 converges to the wrong solution. In a), we show reconstruction of the digits based on the weights from pretrained VAE. In b), reconstruction from M2 after training with initialization of those weights, freezing $\mathbf{z}'$ and changing y in each column. We can see that the best reconstruction is in the column corresponding to the label 6. This suggests that the classifier converged to output the label 6 for all images, which renders it useless.

any meaningful purpose. A scenario where this happens is with a very powerful decoder which is able to model the data directly, such as an autoregressive RNN [6]. In such a scenario, the M2 model may struggle to find the desired solution. This is because part of the posterior approximation $q(y, \mathbf{z}'|\mathbf{x}) = q(y|\mathbf{x})q(\mathbf{z}'|\mathbf{x}, y)$ acts as the classifier, and when the latent space is ignored, the classifier does not get any meaningful training signal - the reconstruction is no longer based on y.

However, it might be possible that M2 does not ignore the full latent $\mathbf{z}$ and the classifier still does not perform well. The usefulness of the generative part of M2 model loss relies on the assumption that the decoder does not ignore the information contained in the part of the latent space y, and performs conditional generation. This intuitively means that if there is no available class information in the latent $\mathbf{z}'$, the optimal choice is to use the information in y. This is desirable, because a wrong inferred y allows for the classifier to learn - the reconstruction based on the wrong label in this case is a lot worse than reconstruction based on the right class label.

Yet, it might be the case that the latent $\mathbf{z}'$ *does* contain all the information necessary to reconstruct the given input, which is the other scenario when the classifier does not perform well. A practical example of this can be seen in Figure 5. In this experiment, we trained a variational autoencoder on unlabeled data, and initialized the M2 model with the corresponding weights. As a result, the model was able to reconstruct the inputs without the use of extra latent space y, and the decoder did not converge to behave like a conditional generative model. Since the reconstruction did not depend on y anymore, the classifier's output was not used and it eventually converged to a solution which was worse than the baseline supervised model, even when trained with a small added discriminative loss. This shows that if the latent $\mathbf{z}'$ contains enough information for the decoder to be able to reconstruct the data, the setup may converge to a wrong solution. This observation is crucial for

the right setup of the model, as it shows improving the flexibility of the approximations for $q(\mathbf{z}'|\mathbf{x})$ may in fact lead to worse classification performance. This is especially significant because many of the recent improvements in variational inference focus on improving the posterior approximations, e.g. by the use of normalizing flows [2, 14, 24].

We would like to note that the same experiment performed with parameter α high enough did not result in deterioration of the classifier, which suggests that the additional discriminative loss is essential for training and performance of the model. We return to this observation in Section 6.

3.3 Latent space of M2 is not disentangled

Disentanglement of the latent space means that distinct aspects of the data are encoded into separate variables in the hidden representation [3, 17, 20]. Such a representation helps with understanding of the inputs, and is one of the factors that allows for manipulation of high-level attributes of the generated data, in other words conditional generation. As shown in Subsection 3.2, successful conditional generation is also fundamental to the usefulness of the generative loss.

In Figure 2, we can see that even though the model has learned to conditionally generate the analogies of digits in most cases, some of the information about the original image remained in $\mathbf{z}'$, the part of latent space that should model the unspecified factors of variation. This can be observed clearly in the generation of digit 1, where the model did not condition on the given label and instead generated digits similar to the original. This observation suggests that the latent space of M2 is not disentangled, as the decoder was able to reconstruct digit 1 purely based on $\mathbf{z}'$. We have also performed an experiment to judge this quantitatively. We train the M2 model in the SSL setting with varying levels of the discriminative loss, and subsequently transformed the training data into the $\mathbf{z}'$ part of the latent space using all possible labels. Finally, we trained a classifier based on that encoded representation $\mathbf{z}'$ with the original labels, and tested the accuracy on the test set. If the latent space were truly disentangled, such a classifier would have accuracy close to random. Yet, the experiment has resulted in accuracy over 85%, regardless of parameter α. This means that the classifier was able to determine the original label based on the encoded representation in vast majority of the cases, suggesting that the latent space is not disentangled.

4 Proposed model

As discussed in Subsection 3.3, even though a separate part of latent space y is used to model the information about the class label, the remaining latent $\mathbf{z}'$ may also contain some of that information. As shown in the experiment described in Subsection 3.2 with the results illustrated in Figure 5, this can potentially stop the classifier from converging to the optimal solution. This motivates the following questions:

1. Given that latent space $\mathbf{z}'$ contains label information, is it possible to construct a classifier based on $\mathbf{z}'$ without using the additional part y? Furthermore, is it possible to do so while keeping conditional generation in order to use reconstruction based loss as a guide for the classifier, similar to the SSL M2 setup?
2. Conversely, is it possible to remove the label information from $\mathbf{z}'$, and is doing so beneficial for training the classifier in the M2 model setup?

In the following subsections, we answer the first question by deriving and evaluating the model based on the latent space in Subsection 4.1. We leave the second question for the discussion, with

a preliminary solution of an adversarial approach to remove the label information from $\mathbf{z}'$ proposed in Appendix E.

4.1 Semi-supervised learning based on latent space

In order to answer the first question, we start with the following proposition:

Proposition 1 *Let $\hat{\mathbf{x}} = [\mathbf{x}, y]$, the concatenation of a data point and its label. Introduce a posterior approximation $q(\mathbf{z}|\hat{\mathbf{x}})$ and assume $q(\mathbf{z}|\hat{\mathbf{x}}) \approx q(\mathbf{z}|\mathbf{x})$. Then $\log p(\hat{\mathbf{x}})$ for a single tuple $(\mathbf{x}, y)$ has an evidence lower bound (ELBO) expressed as:*

$$\log p(\hat{\mathbf{x}}) = \log p(\mathbf{x}, y) \geq \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|y, \mathbf{z}) + \log p(y|\mathbf{z}) + \log p(\mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})] = \mathcal{L}_{our}(\mathbf{x}, y)$$

This defines a lower bound which we can maximise, using the same techniques as in variational inference. Note that it includes both the conditional generative model $p(\mathbf{x}|y, \mathbf{z})$ and the discriminative term $p(y|\mathbf{z})$ as desired.

For unlabeled data, we use a similar variational bound as in variational autoencoders (Equation 1), with the only change motivated by the observation that $\log p(\mathbf{x}|z) \geq \mathbb{E}_{p(y|\mathbf{z})}[\log p(\mathbf{x}|y, \mathbf{z})]$ which is formalised in the following proposition:

Proposition 2 *The ELBO of variational autoencoders for a single data point $\mathbf{x}$ can be lower bounded by:*

$$\begin{aligned}\log p(\mathbf{x}) &\geq \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z}) + \log p(\mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})] \\ &\geq \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\mathbb{E}_{p(y|\mathbf{z})}[\log p(\mathbf{x}|y, \mathbf{z})] + \log p(\mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})] = \mathcal{U}_{our}(\mathbf{x})\end{aligned}$$

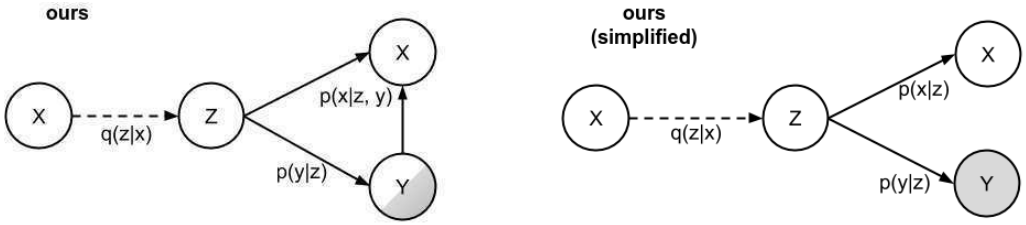
The additional inequality provides a less tight lower bound to the log-likelihood, yet maximising an objective with this inequality ($\mathcal{U}_{our}$) is aligned with the original goal of maximising ELBO. The inequality allows us to use the same $p(y|\mathbf{z})$ as in the labeled case. Such approximation is potentially beneficial as $p(y|\mathbf{z})$ acts as a classifier, and by using this additional bound we can train this classifier also in the unlabeled case. This observation also makes the model training dependent on the conditional generation, which was one of the objectives of the first research question.

Based on the propositions, we construct a model which optimizes a lower bound for likelihood of the data, and after introducing λ as a hyper-parameter to weigh labeled and unlabeled loss, the final maximisation objective becomes:

$$\mathcal{J}_{our} = \lambda \cdot \mathcal{L}_{our}(\{\mathbf{x}, y\}_{i=1}^{l}) + \mathcal{U}_{our}(\{\mathbf{x}\}_{i=l+1}^{l+u})$$

The model is shown graphically in Figure 6a. We can see that it has similar properties to M2 from Kingma et al. [13]. Indeed, we can see the classifier is based on conditional reconstruction due to $p(\mathbf{x}|y, \mathbf{z})$. Yet, unlike Kingma's model, our model does not require an ad-hoc addition of discriminative loss, as both labeled and unlabeled objectives already contain the classification model. We introduce λ to control the relative weight of labeled and unlabeled data as has been previously suggested [5].

While the models share many similarities, the one we propose does not introduce any assumptions for the latent space, contrary to the M2. Specifically, in M2 the latent space was split in two parts,



(a) (b)

Figure 6: Probabilistic graphical models (ours). The dashed lines indicate inference models (encoders), the solid lines indicate the generative models (decoders). Grey background on elements y corresponds to operations only on observed labels, and split background indicates the semi-supervised setup.

one of which was required to be categorical. In our model, there are no such restrictions. As a result, application of modern techniques for improving the posterior approximation - such as those presented in [2, 14, 24] - is straightforward and does not require any further assumptions. These techniques have been shown to improve the flexibility of the posterior approximation, resulting in a tighter lower bound for the likelihood. Maaløe et al. [18] have shown that improving the ability to fit complex posteriors can also lead to better classification performance in the SSL setting. Such methods can also potentially improve the robustness of the model: a more flexible posterior approximation can better fit the prior, and as a result the model may be less constrained by the regularizing KL divergence term in the loss. What is more, lifting the restriction that part of the latent space is categorical allows for the use of more complicated priors, which have been shown to further improve the log-likelihood of the model [27] and can help with the problem of 'component collapse' [19].

Now, due to the observations in Subsection 3.1, we note that generative loss as seen in this model can also make the classifier perform worse. As shown in Subsection 3.2, one of the reasons for such behaviour could be the dependence of the model on conditional generation. Therefore, we propose introducing an additional assumption, $p(\mathbf{x}, y|\mathbf{z}) \approx p(\mathbf{x}|\mathbf{z})p(y|\mathbf{z})$, which simplifies the model by removing the link which supports the conditional generation. Because the conditional generation is removed, the model becomes potentially more robust against bad initializations, such as that one described in Subsection 3.2. This results in a model shown in Figure 6b, which we call *simplified*. As can be seen in the figure, it resembles M1 from Kingma et al. [13]. Yet, the classifier in our setup is also trained alongside the variational autoencoder (instead of in two steps like in M1).

5 Evaluation

5.1 Experimental setup

We evaluate the performance of our models on MNIST [15] dataset. For the experiments, we normalize intensity of the images to the range $[0, 1]$ and split the dataset into train, validation and test sets, containing 50000, 10000 and 10000 data points respectively. We use 100 randomly selected

λ or α	M2	ours	ours, simplified
0	88.6 ($\pm$ 5.0)	-	-
0.001	90.3 ($\pm$ 3.5)	72.2 ($\pm$ 3.7)	82.1 ($\pm$ 1.9)
0.1	91.4 ($\pm$ 2.2)	91.7 ($\pm$ 3.6)	80.4 ($\pm$ 0.7)
0.5	91.7 ($\pm$ 2.5)	92.3 ($\pm$ 2.8)	79.5 ($\pm$ 1.1)
1	92.0 ($\pm$ 2.6)	88.2 ($\pm$ 3.6)	78.7 ($\pm$ 0.7)
10	92.6 ($\pm$ 2.5)	78.1 ($\pm$ 1.3)	76.7 ($\pm$ 0.3)
100	93.3 ($\pm$ 2.0)	80.2 ($\pm$ 0.9)	75.9 ($\pm$ 0.8)
10000	93.7 ($\pm$ 2.5)	70.9 ($\pm$ 0.4)	72.4 ($\pm$ 0.9)

Table 1: Benchmark results of semi-supervised classification on MNIST with 100 labels (accuracy).

labeled data points from the training data set, 10 from each class. The remaining data in the training set is treated as unlabeled. The batch sizes for the training are chosen to be 100 labeled and 100 unlabeled data points. The hyperparameters of the networks are listed in Appendix B and are taken from [13]. Training was stopped after 500 epochs, and the best performance was chosen based on the validation set. As a baseline, we use a fully supervised neural network trained only on the 100 labeled data points. For M2 model we performed a grid search for the best parameter α, with the possible values coming from the set $\{0, 0.001, 0.1, 0.5, 1, 10, 100, 10000\}$. For our model, we performed a similar grid search for parameter λ.

5.2 Results

The baseline model resulted in accuracy of $74.74 \pm 1.16\%$. When evaluating M2, we found that the model highly benefits from the added discriminative learning and achieves the best performance of $93.7 \pm 2.5\%$ with $\alpha = 10000$. Our model achieved the accuracy of $92.3 \pm 2.8\%$ with $\lambda = 0.5$. We report full results in Table 1.

The models achieve similar peak performance, yet when removing the conditional generation link, the accuracy drops. Because of this, and the fact that the models share very similar structure, we believe that conditional generation is the main reason of good accuracy for both M2 and our model.

The proposed achieves the best performance with $\lambda = 0.5$, which suggests that it works best with relatively balanced labeled and unlabeled loss. This shows that the model benefits highly from discovering the underlying structure of the data based on the unlabeled samples. Yet, we would like to note that tuning this parameter may greatly change the performance of our model, more than tuning α in M2. We return to this observation in Section 6.

Considering that the best performance of M2 also comes from the highest setting of α, this further confirms that the extra discriminative loss is essential to its success. This also highlights the difference between two models: while our model benefits from modeling the underlying representation of the data, the M2 model relies more on the initial classification, making it more akin to self-learning [1].

Furthermore, we tested our model with the extra assumption, $p(\mathbf{x}, y|\mathbf{z}) \approx p(\mathbf{x}|\mathbf{z})p(y|\mathbf{z})$, the simplified version. In Table 1 we can see that the model benefits only slightly from the unlabeled data, resulting in at most $82.1 \pm 1.9\%$ with $\lambda = 0.001$. This means that the model did not improve much over the baseline. The reason for it is that with such a small amount of labeled data, the classifier overfits in the first few iterations, and later the model improves only the reconstruction ability. The peak

performance was achieved with $\lambda = 0.001$, which suggests that it is beneficial for this setup to first learn the best encoding of all the data before training a labeled classifier, similar to M1 by Kingma et al. [13].

We also performed an experiment with setup similar to the one in Subsection 3.2, where we pretrained the unlabeled part of both versions of our model. In such a scenario, the model without simplification failed to beat the baseline, yet the simplified model achieved accuracy of $81.6 \pm 1.2\%$, comparable to the scenario without pretraining. This confirms that conditional generation is one of the reasons of the good performance of our model, and the improvement over the baseline of our simplified model comes from discovering the hidden structure of the data.

6 Discussion and Conclusion

In this work, we started out by analysing the behaviour of the M2 model [13] and highlighting its strengths and weaknesses. In particular, we argued that the main reason for its performance is conditioning reconstruction of the data on the label, and as such the model needs to be able to reconstruct the data well. Next, we showed that the initialization of this model is crucial for the classifier to converge to the right solution. We also showed that the M2 model is highly dependent on the auxiliary discriminative loss.

Based on these observations, we derived a new model, which does not impose any assumptions on the latent space. The proposed model optimizes a principled lower bound to the log-likelihood, similarly to M2 benefits from conditional generation, and does not require the ad-hoc addition to the objective function.

During the experimentation, we noted that the M2 model is highly dependent on the auxiliary discriminative loss, and we showed that the model does not recover from bad initialization without the addition. What is more, the model achieves best performance with the highest tested setting of α. We note that such behaviour makes it similar to self-learning [1], perhaps a version with "soft" labels. We believe that the classifier learns to assign the true labels to the labeled data because of high discriminative loss. Then it assigns "soft" labels to unlabeled data and improves the results based on the reconstruction. As can be seen from the loss (Equation 3), the model's reconstruction loss in the unlabeled case is weighted according to the soft labels, and therefore the gradients for reconstruction of different labels are also updated in the same manner. This resembles the aforementioned self-learning setup, where the most confident predictions are considered valid, and the next iteration of algorithm is using them as true labels. In particular, generative mixture models can be considered as a special case of such "soft" self-training [30]. Interestingly, mixture models assume a model of $p(x, y) = p(y)p(x|y)$, similar to the conditional generation of M2. They may also be prone to local maxima with bad initializations [21], which we experimentally found to be the case for the Kingma's model. It would be interesting to further analyze and compare the two models, using the knowledge gathered from the extensive research about the classical method.

By introducting a new model, we showed that the assumptions made on the latent space by Kingma et al. [13] are not necessary. We would like to remark that it is possible for our model to resemble that of [13], by introducing similar assumptions. Yet, even in that case, our model does not need an ad-hoc addition to the objective function. We discuss the details in Appendix C. What is more, we argue that the M2 model does implicitly depend on a parameter λ similar to the one in our setting, and so it is more difficult to fine-tune, especially in the SSL setting. The extra parameter is due to batch sizing. Indeed, when using equal sized batches of labeled and unlabeled data in

the most common batch gradient descent setting, the optimizer effectively performs $\frac{u}{l}$ times more gradient updates based on the labeled data, where l and u are the number of labeled and unlabeled data points respectively. More generally, if n_l, n_u are the batch sizes of labeled and unlabeled data, the optimizer performs $\frac{n_l \cdot u}{n_u \cdot l}$ gradient updates based on the labeled data for 1 update based on the unlabeled data. While the effect is not exactly the same as applying a single gradient update multiplied by a constant, this suggests that different batch sizes may influence the performance of the M2 model, similarly to parameter λ of our model. This is also confirmed in an experiment we performed, results of which can be found in Appendix D.

The absence of assumptions on the latent space in our model allows for a straightforward application of recent developments in posterior approximation and different priors [27], [19]. In particular, we would like to highlight one of the recent developing fields of normalizing flows [2, 14, 24] which allow for better posterior approximations, which could be applied to our model directly. Such application could lead to richer latent representations. More complex posterior approximations have been shown to also improve the classification performance [18], yet as we reasoned in Subsection 3.2, improving posterior approximation may also lead to a deteriorated performance, because the model may be prone to local maxima.

Finally, we reasoned that one of the ways to reduce the susceptibility to bad initialization of M2 could be better disentanglement of its latent space. We raised the question of whether this would also benefit the classification ability of such model. We leave this question for future work, yet preliminary results suggesting that adversarial methods could be used, together with initial experiments can be found in Appendix E.

A Proofs

Proof of proposition 1:

We start with the usual variational bound for data, defining $\hat{x}$ as concatenation of the data with the label, i.e. $\hat{\mathbf{x}} = [\mathbf{x}, y]$.

$$
\begin{aligned}
\log p(\hat{\mathbf{x}}) &= \log \int p(\hat{\mathbf{x}}, \mathbf{z}) d\mathbf{z} \\
&= \log \int p(\hat{\mathbf{x}}|\mathbf{z}) p(\mathbf{z}) d\mathbf{z} \\
&= \log \int \frac{q(\mathbf{z}|\hat{\mathbf{x}})}{q(\mathbf{z}|\hat{\mathbf{x}})} p(\hat{\mathbf{x}}|\mathbf{z}) p(\mathbf{z}) d\mathbf{z} \\
&= \log \mathbb{E}_{q(\mathbf{z}|\hat{\mathbf{x}})}[\frac{p(\hat{\mathbf{x}}|\mathbf{z})p(\mathbf{z})}{q(\mathbf{z}|\hat{\mathbf{x}})}] \\
&= \log \mathbb{E}_{q(\mathbf{z}|\mathbf{x},y)}[\frac{p(\mathbf{x}, y|\mathbf{z})p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x}, y)}]
\end{aligned}
\tag{4}
$$

Now, using the assumption that $q(\mathbf{z}|\mathbf{x}, y) \approx q(\mathbf{z}|\mathbf{x})$, we have:

$$\begin{aligned}
\log \mathbb{E}_{q(\mathbf{z}|\mathbf{x},y)}[\frac{p(\mathbf{x}, y|\mathbf{z})p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x}, y)}] &\approx \log \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\frac{p(\mathbf{x}, y|\mathbf{z})p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x})}] \\
&= \log \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\frac{p(\mathbf{x}|\mathbf{z}, y)p(y|\mathbf{z})p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x})}] \\
&\geq \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log \frac{p(\mathbf{x}|\mathbf{z}, y)p(y|\mathbf{z})p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x})}] \\
&= \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z}, y) + \log p(y|\mathbf{z}) + \log p(\mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})] = \mathcal{L}_{our}(\mathbf{x}, y)
\end{aligned} \tag{5}$$

using Jensen's inequality.

Proof of proposition 2:

We note that

$$\begin{aligned}
p(\mathbf{x}|\mathbf{z}) &= \int p(\mathbf{x}, y|\mathbf{z})dy = \int p(\mathbf{x}|y, \mathbf{z})p(y|\mathbf{z})dy \\
&= \mathbb{E}_{p(y|\mathbf{z})}[p(\mathbf{x}|y, \mathbf{z})]
\end{aligned} \tag{6}$$

Using Jensen's inequality, we have $\log p(\mathbf{x}|\mathbf{z}) \geq \mathbb{E}_{p(y|\mathbf{z})}[\log p(\mathbf{x}|y, \mathbf{z})]$. The result follows naturally by substituting $\log p(\mathbf{x}|z)$ in the ELBO as appears in [12].

B Hyperparameters for experiments

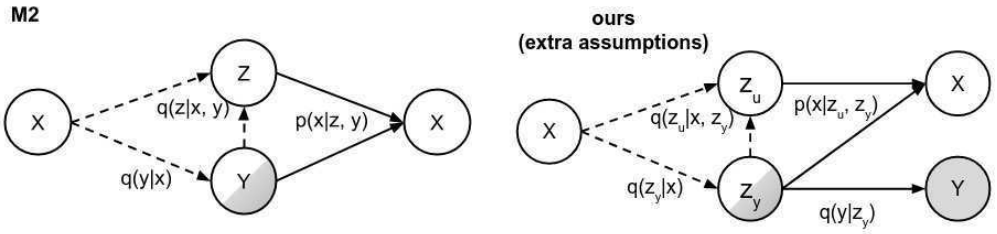
For all of the experiments, we used the hyperparameters listed in this section, unless specified otherwise. We used Adam optimizer [11] with a learning rate of 0.001, and the weights of the networks were initialized with Xavier initializer [8]. Each of the networks that was part of the generative and inference models was a fully connected neural network (MLP) with a single hidden layer of 500 units, using softplus activation functions. We used a 50-dimensional latent space. Parameter α was set to 1.

C Special case of our model is similar to the M2 model

Kingma et al. [13] assume that the latent space $\mathbf{z}$ consists of two parts, $\mathbf{z}'$ and y, corresponding to unspecified factors of variation and factors of variation defining the class label. Our model does not have such assumptions, yet by introducing $\mathbf{z} = [\mathbf{z_u}, \mathbf{z_y}]$, assuming $q(\mathbf{z_u}, \mathbf{z_y}|\mathbf{x}) = q(\mathbf{z_u}|\mathbf{z_y}, \mathbf{x})q(\mathbf{z_y}|\mathbf{x})$, $p(\mathbf{x}, y|\mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(y|\mathbf{z})$ and restricting $p(y|\mathbf{z})$ to $p(y|\mathbf{z_y})$ we can see that the objective becomes:

$$\begin{aligned}
\mathcal{L}_{our}(\mathbf{x}, y) &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|y, \mathbf{z}) + \log p(y|\mathbf{z}) + \log p(\mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})] \\
&= \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z}) + \log p(y|\mathbf{z_y}) + \log p(\mathbf{z}) - \log q(\mathbf{z_u}|\mathbf{z_y}, \mathbf{x}) - \log q(\mathbf{z_y}|\mathbf{x})]
\end{aligned} \tag{7}$$

While the connection to the M2 model is not immediately obvious, it might be better understood by looking at Figure 7. We can see that similar links for both the inference and generation are supported. One of the differences is the lack of generative model for y in M2 (which was fixed with auxiliary loss), and the other can be seen in the classifier, which in this special case for our model becomes $p(y|\mathbf{z_y})q(\mathbf{z_y}|\mathbf{x})$, i.e. contains a stochastic unit. This suggests it could potentially be more robust to noise in the inputs, yet we leave the evaluation for future work.



(a) (b)

Figure 7: Probabilistic graphical models of M2 [13] and ours with multiple extra assumptions. The dashed lines indicate inference models (encoders), the solid lines indicate the generative models (decoders). Grey background on elements y corresponds to operations only on observed labels, and split background indicates the semi-supervised setup.

D Batch size setting may influence the performance

In order to show quantitatively that batch size settings may influence the performance of the model, we have trained the M2 model in two different settings: with 100 labeled and 100 unlabeled data points in a batch; and 1 labeled and 499 unlabeled data points in a batch. The results can be seen in Figure 8. We can see that the model's performance is changing depending on the ratio of labeled data points to unlabeled data points in a batch.

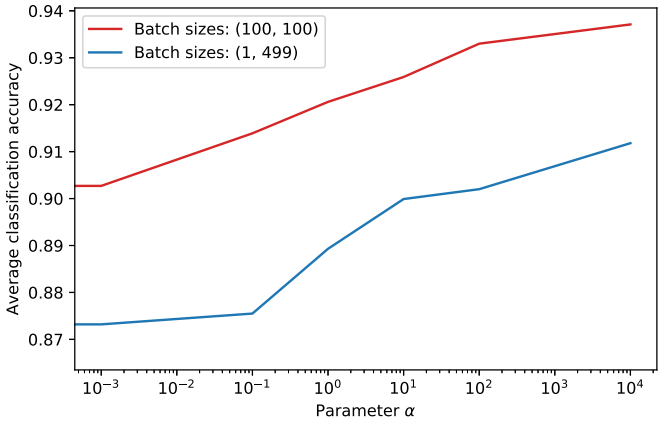


Figure 8: Performance of the M2 model with different batch size settings. The red line corresponds to the setting of 100 labeled and 100 unlabeled points in a batch, and the blue to 1 labeled and 499 unlabeled points in the batch.

E Adversarial semi-supervised learning with deep generative models

In order to answer the second question in Section 4 we introduce an adversarial classifier $f_\psi(y|\mathbf{z}')$ to the M2 model setup, with the goal of removing the label information from the part of the latent space which is meant to model the unknown factors of variation, $\mathbf{z}'$. We define the objective as

$$\min_{\psi} \max_{\theta,\phi} \mathcal{J}^{\alpha}_{\theta,\phi} + \mathbb{E}_{q_\phi(\mathbf{z}'|\mathbf{x},y)}[-\log f_\psi(y|\mathbf{z}')],$$

where $\mathcal{J}^{\alpha}_{\theta,\phi}$ is the objective of the M2 model (with ϕ, θ parameters of encoder and decoder respectively) and $\mathbb{E}_{q_\phi(\mathbf{z}'|\mathbf{x},y)}[-\log f_\psi(y|\mathbf{z}')]$ is the objective of adversarial classifier. This loss can be better understood as a game, where the networks in the M2 model try to find the best ELBO approximation, while the adversarial classifier is trying to detect whether there is any information about the label in $\mathbf{z}'$. The model is shown in Figure 9. We can see that the equilibrium could be found by removing the label information from $\mathbf{z}'$ and instead modeling that part of data through y. In other words, this setup may in theory force a disentangled representation, which in turn may alleviate the problem shown in Subsection 3.2.

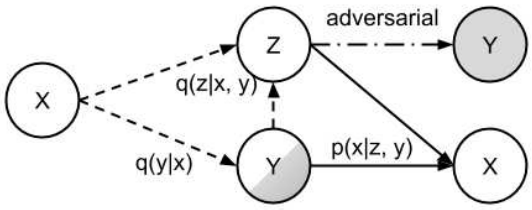


Figure 9: M2 model [13] with added adversarial connection. The dashed lines indicate inference models (encoders), the solid lines indicate the generative models (decoders), and the dashed-dotted line indicates an adversarial classifier. Grey background on elements y corresponds to operations only on observed labels, and split background indicates the semi-supervised setup.

We performed preliminary experiments using this extended loss, yet we were not able to find any difference in the model's performance. However, we note that adversarial setups are known to be difficult to train [25] and so we believe further analysis is necessary in order to determine whether or not the extended loss may improve the classification performance of the model.

References

[1] Steven Abney. Understanding the yarowsky algorithm. *Computational Linguistics*, 30(3):365–395, 2004.

[2] Rianne van den Berg, Leonard Hasenclever, Jakub M Tomczak, and Max Welling. Sylvester normalizing flows for variational inference. *arXiv preprint arXiv:1803.05649*, 2018.

[3] Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in beta-vae. *arXiv preprint arXiv:1804.03599*, 2018.

[4] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[5] Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010. ISBN 0262514125, 9780262514125.

[6] Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. *CoRR*, abs/1611.02731, 2016.

[7] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[8] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[9] Gisel Bastidas Guacho, Sara Abdali, Neil Shah, and Evangelos E Papalexakis. Semi-supervised content-based detection of misinformation via tensor embeddings. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 322–325. IEEE, 2018.

[10] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR*, 2(5):6, 2017.

[11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[12] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[13] Diederik P. Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *NIPS*, pages 3581–3589, 2014.

[14] Durk 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*, pages 4743–4751, 2016.

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

[16] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*, volume 793. John Wiley & Sons, 2019.

[17] Francesco Locatello, Stefan Bauer, Mario Lucic, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. *CoRR*, abs/1811.12359, 2018.

[18] Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Improving semi-supervised learning with auxiliary deep generative models. In *NIPS Workshop on Advances in Approximate Bayesian Inference*, 2015.

[19] Eric Nalisnick and Padhraic Smyth. Stick-breaking variational autoencoders. *arXiv preprint arXiv:1605.06197*, 2016.

[20] Siddharth Narayanaswamy, T Brooks Paige, Jan-Willem Van de Meent, Alban Desmaison, Noah Goodman, Pushmeet Kohli, Frank Wood, and Philip Torr. Learning disentangled representations with semi-supervised deep generative models. In *Advances in Neural Information Processing Systems*, pages 5925–5935, 2017.

[21] Kanal Paul Nigam. *Using Unlabeled Data to Improve Text Classification*. PhD thesis, Pittsburgh, PA, USA, 2001. AAI3040487.

[22] George Papandreou, Liang-Chieh Chen, Kevin P. Murphy, and Alan L. Yuille. Weakly- and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[24] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[25] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016.

[26] Samuel Thomas, Michael L. Seltzer, Kenneth Church, and Hynek Hermansky. Deep neural network features and semi-supervised training for low resource speech recognition. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, May 2013. doi: 10.1109/icassp.2013.6638959.

[27] Jakub M Tomczak and Max Welling. Vae with a vampprior. *arXiv preprint arXiv:1705.07120*, 2017.

[28] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[29] Yawen Xiao, Jun Wu, Zongli Lin, and Xiaodong Zhao. A semi-supervised deep learning method based on stacked sparse auto-encoder for cancer prediction using rna-seq data. *Computer methods and Programs in Biomedicine*, 166:99–105, 2018.

[30] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2005.

Supplementary Material

This supplementary material is meant to complement the main thesis form. It is written in a more informal way, aimed to improve the understanding and intuition behind the ideas presented previously. In the first section, we include more information about latent variable models and variational autoencoders, the knowledge of which should be helpful to understand the main text. The second part contains an explanation of normalizing flows, which we investigated prior to the main question answered in the thesis. While they do not play a central role in the main part of the report, they form the foundation of the motivation for the ideas researched. Finally, we include a crude section with plots that did not fit into the main paper but are also interesting, and we include some partially explored ideas.

1 Extended background

1.1 Latent variable models

Latent variable models are statistical models, which assume that the data (observable variables) were generated from a set of hidden variables, called *latent variables*. In the case of very-high dimensional data (such as images), these latent variables are usually assumed to be coming from a lower dimensional distribution. In other words, the latent variables can be understood as the "high-level" characteristics of the data. One of the (indirect) goals of latent variable models can be to discover such characteristics, i.e. find the *posterior distribution* of the latent variables. This may be desired for example to better understand the data, or boost the performance of further operations on the data by obtaining a lower dimensional representation. Usually, latent variable models are optimized to maximize the likelihood of the data, i.e. find the best fit to the marginal distribution $p(x)$ over the observable variables, which often makes it easy to retrieve the posterior.

More formally, latent variable models introduce a probability distribution $p_\theta(x, z)$ over the set of observable variables x and latent variables z with some parameters θ. Such models are commonly factorized into a conditional likelihood $p_\theta(x|z)$ and a prior $p_\theta(z)$:

$$p_\theta(x, z) = p_\theta(x|z)p_\theta(z) \tag{1}$$

A classical example of latent variable models which may help with their understanding is a Gaussian Mixture Model (GMM). In this model, the observations are assumed to be coming from a mixture of Gaussian distributions. Each of the components of the mixture $p_\theta(x|z)$ is assumed to be Gaussian, and they are combined in a convex combination to form the distribution of the data. An illustration of the Gaussian mixture can be seen in Figure 1. Intuitively, knowing the posterior distribution is useful, as given a particular value of an observation x, we can answer the question of "which of the underlying mixtures did this observation likely come from?", and therefore perform classification (or "soft" clustering). In the case of Gausian Mixture Models, the analytical form of the posterior $p_\theta(z|x)$ is known (it is a categorical distribution), and therefore the parameters θ can be optimized by classical methods such as expectation-maximization (EM) [10].

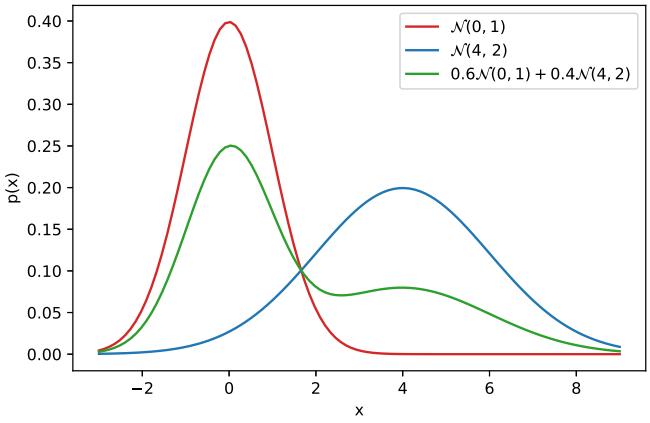


Figure 1: The probability density function of a mixture of Gaussian distributions and its two components, $\mathcal{N}(0, 1)$ and $\mathcal{N}(4, 2)$. The weighing is chosen to be 0.6 for the first component, and 0.4 for the second.

However, not all the latent variable models can be optimized with EM. For example, this algorithm cannot be applied when it is impossible to compute the posterior in a closed form. This happens for example in the case when $p_\theta(x|z)$ are parameterized with neural networks with at least one non-linear hidden layer.

1.2 Variational autoencoders

Variational autoencoders are another type of latent variable model which has the form of $p_\theta(x, z) = p_\theta(x|z)p_\theta(z)$. In contrast to GMM, the true posterior distribution is unknown, and so the optimization by traditional EM can no longer be applied. One way of solving the problem of finding such posterior distribution is to introduce a family $\mathcal{D}$ of densities over the latent variables, and perform variational inference (sometimes also called variational EM) [2].

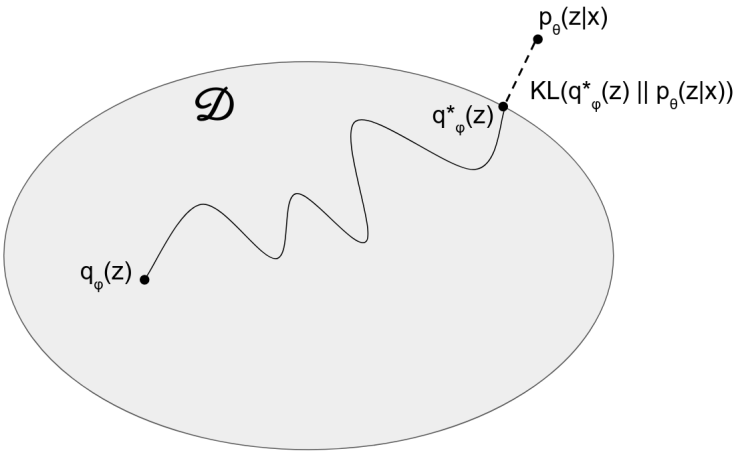


Figure 2: Illustration of variational inference. Instead of computing the true posterior $p_\theta(z|x)$, we introduce a variational family of distributions $\mathcal{D}$, and fit the variational parameters ϕ to make $q_\phi(z) \in \mathcal{D}$ close to the true posterior.

After introducing parametric inference model $q_\phi(z)$, which comes from family $\mathcal{D}$, we try to optimize its parameters in order to find the member of that family such that it minimizes the Kullback-Leibler

divergence to the exact posterior:

$$q_\phi^*(z) = \arg\min_{q_\phi(z)\in\mathcal{D}} \mathrm{KL}[q_\phi(z)\|p_\theta(z|x)] \tag{2}$$

This inference model in the end approximates the true posterior. This is visualised in Figure 2.

However, objective in Equation 2 is also not computable directly, due to dependence on the unknown evidence $p_\theta(x)$, which can be seen by applying Bayes' Theorem:

$$p_\theta(z|x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(x)} \tag{3}$$

Because of that, variational inference optimizes an alternative objective that is equivalent to KL divergence up to a constant:

$$\mathrm{ELBO}(q) = \mathbb{E}_{q_\phi(z)}[\log p_\theta(x,z) - \log q_\phi(z)] = \log p_\theta(x) - \mathrm{KL}[q_\phi(z)\|p_\theta(z|x)] \tag{4}$$

This function is called the evidence lower bound (ELBO), because it is optimizing a lower bound for $\log p_\theta(x)$, as can be seen in the equation. Note how maximizing ELBO is equivalent to minimizing the $\mathrm{KL}[q_\phi(z)\|p_\theta(z|x)]$ which appears in Equation 2 - this is because the evidence $\log p_\theta(x)$ does not depend on parameters ϕ, and so during such maximization, it remains constant.

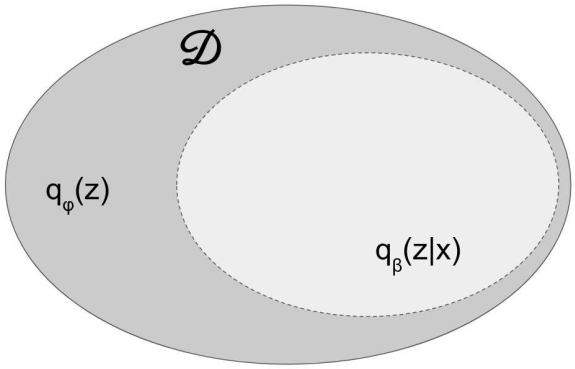
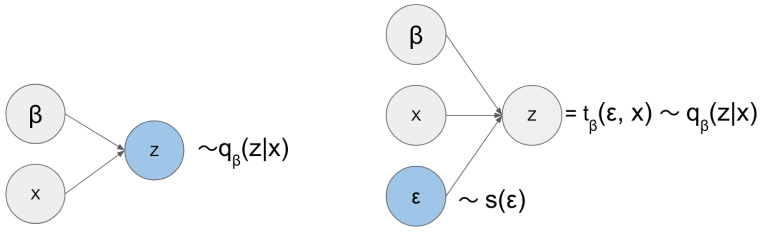


Figure 3: Illustration of amortized inference. This assumption limits the family of distributions that we assumed initially, but it makes the optimization faster. The size of the smaller class depends on the flexibility of f [1].

Variational autoencoders optimize ELBO, making use of two extra techniques in order to make this process scalable and faster to train. First, VAEs assume the parameters ϕ of q_ϕ are based on the data, $q_\phi(z) = q_\beta(z|x;\phi = f_\beta(x))$. Without this assumption, stochastic optimization would be required for each data point. Yet, by introducing a mapping $f_\beta(x)$ from x to ϕ, the number of parameters is kept constant. This assumption is often referred to as "amortizing inference" [3, 4]. Effectively, by making such assumption, we shrink the class of variational approximations and so this can be considered a computational-statistical tradeoff [1]. This is depicted in Figure 3.



(a) (b)

Figure 4: Illustration of reparametrization trick [6]. Blue nodes correspond to random operations, the grey nodes are deterministic.

Secondly, VAEs use what is known as the "reparametrization trick" [6], in order to reduce the variance of the gradients. This is shown in Figure 4. In short, this technique allows for faster training due to the reduced variance of the gradients, but not without drawbacks. The technique requires the variational approximation to have the property that we can obtain the sample from $q_\phi(z)$ by transforming noise ϵ drawn from a noise distribution $s(\epsilon)$ with some differentiable transformation t_ϕ, in other words:

$$z = t_\phi(\epsilon) \text{ for } \epsilon \sim s(\epsilon) \text{ implies } z \sim q_\phi(z)$$

An example of such transformation is the reparametrization of the Gaussian distribution: starting with noise from the standard normal distribution, $\epsilon \sim \mathcal{N}(0, 1)$, we can obtain any other Gaussian distribution by applying the transformation $t_\phi(\epsilon) = \mu_\phi + \sigma_\phi \epsilon$, where μ_ϕ and σ_ϕ are for example retrieved by applying a deterministic function approximated by neural networks using the parameters ϕ.

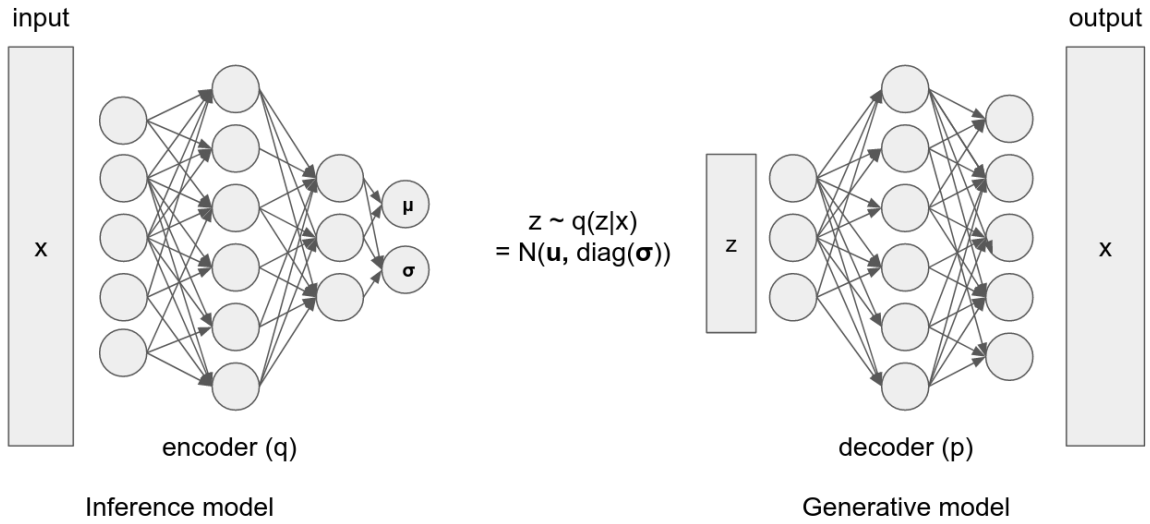


Figure 5: An example of a variational autoencoder. The neural networks are shown as a directed graph. Note that there is no requirement for these neural networks, and almost any deep network architecture can be applied. Note μ and σ are presented as two nodes for simplicity, yet they are usually multidimensional and so the networks output more than two scalars. The setup can be trained end-to-end using standard modern optimization techniques.

Finally, VAEs usually use mean-field approximation, that is assume that all the dimensions of the latent space are independent given the parameters, $q_\phi(z|x) = \Pi_i q_\phi(z_i|x)$. All these assumptions

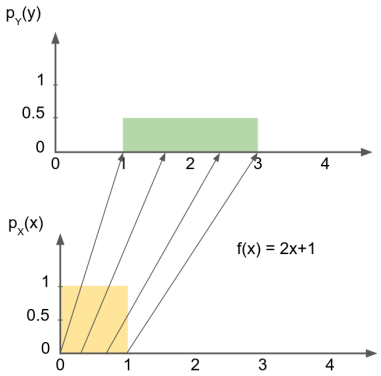


Figure 6: An example of a transformation of one density to another, from $\mathcal{U}(0,1)$ in yellow to $\mathcal{U}(1,3)$ in green, by applying the mapping $f(x) = 2x + 1$.

lead to a model which resembles a traditional autoencoder, shown in Figure 5, with a probabilistic encoder and a probabilistic decoder parameterized by neural networks. Usually, the encoder is a factorized Gaussian encoder, $q_\phi(z|x) = \mathcal{N}(z; \mu(x), \text{diag}(\sigma^2(x)))$, where $\mu(x)$ and $\sigma^2(x)$ are functions approximated with neural networks. The decoder is often chosen to be a Gaussian, Bernoulli or a categorical distribution, depending on whether the observations are continuous or discrete.

2 Extended motivation

2.1 Normalizing flows

The assumptions and techniques used in variational autoencoder which allowed them to scale to large datasets, also made the posterior approximation simpler. This is because the true posterior might not necessarily be captured in the chosen posterior family $\mathcal{D}$, and the amortization of inference could shrink the search space even more. Also, the mean-field approximation further reduced the space from which the posterior approximation could come from. While not all of the assumptions are necessary and alternative optimization methods exist (e.g. black-box variational inference [12]), very often these methods do not scale as well as variational autoencoders. Therefore, a lot of research focuses on improving the posterior approximation, without removing the assumptions. In the literature, two ideas of improving the posterior appear frequently: methods based on auxiliary variables and methods based on change-of-variables [1]. The addition of auxiliary variables to semi-supervised learning with deep generative models was already shown to improve the performance of such models [9], and therefore our attention was brought to change-of-variable methods.

The idea of improving the posterior approximation may be based on the change of variables theorem from statistics:

$$p_Y(y) = p_X(x) \left| \det \frac{df^{-1}}{dy} \right| = p_X(x) \left| \det \frac{df}{dx} \right|^{-1} \tag{5}$$

where $p_Y(y)$ and $p_X(x)$ are densities, and f is a bijective, differentiable mapping. Using this theorem, one may transform any density to any other density, provided that f exists. An example of such transformation is shown in Figure 6.

First shown by Rezende and Mohamed [13], a *normalizing flow* "describes the transformation of a density by a sequence of multiple such mappings", and often these mappings are parameterized and learnt by neural networks. Applying long chains of density changes allows for increasingly complex and multi-modal distributions, as illustrated in Figure 7.

The key question in normalizing flows is how to define the transformation function: it has to be bijective and differentiable. Also, since normalizing flows were designed to fix the simplifications of posterior due to computational complexity as discussed in subsection 1.2, the implicit requirement

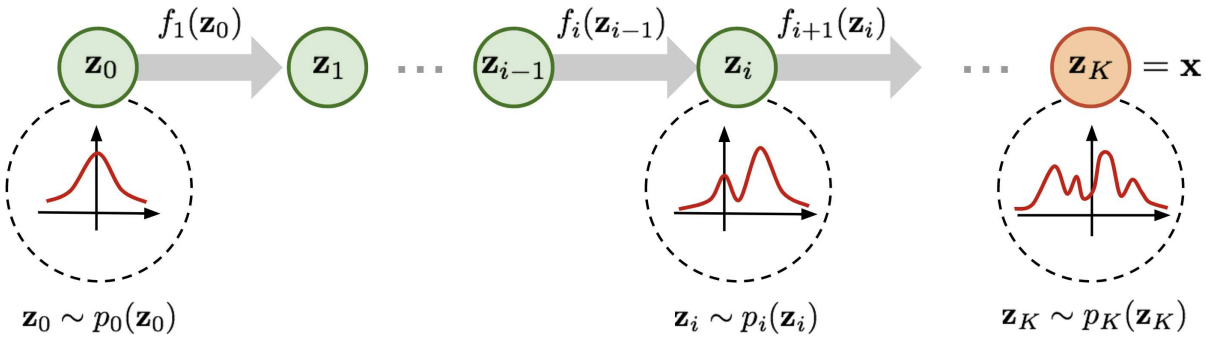


Figure 7: By applying multiple transformations on the density, a more complex density can be retrieved. Image taken from [15].

is that they do not increase the asymptotic computational complexity. This is currently a very active area of research, and most of the recent advances in this field are based on autoregressive transformations [5, 8, 14].

Unfortunately, due to the requirement of the transformation being bijective, normalizing flows cannot be straightforwardly applied to discrete distributions. In the context of semi-supervised learning with deep generative models as shown by Kingma et al. [7], this means that these techniques cannot be implemented without changing the model - since part of the latent space which corresponds to the labels is assumed to be categorical. Therefore, it could be beneficial to adopt the model for semi-supervised learning in such a way that the assumption of categorical distribution is no longer needed.

3 Additional materials

3.1 Semi-supervised learning with generative loss from M2

Below we present some of the experimentation results with the M2 model from [7].

In this experiment, we trained the model with only generative loss, in order to better understand the principles of semi-supervised classification of such model. During training we used KL divergence warmup, i.e. increased the relative importance of KL divergence in the loss linearly, until it was fully incorporated at epoch 100. In Figure 8 we can see the results of the reconstruction and the classifier after 70 epochs, when KL divergence was not yet fully incorporated. In the last panel, we show the classification confidence based on the reconstruction loss, which were obtained after performing *softmax* function on negative reconstruction losses. This may be intuitively interpreted as "soft" clustering, similar to GMM. We can see that the both of the classifiers perform relatively well, and the classification boundaries approximately coincide with the original densities.

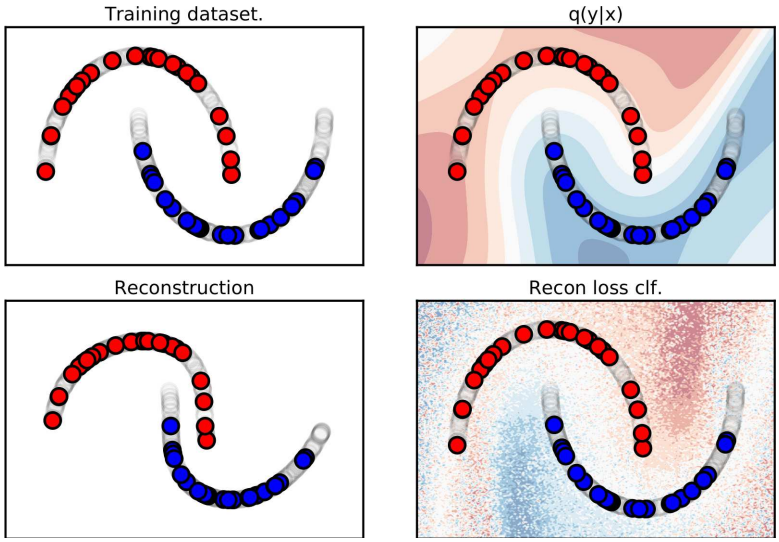
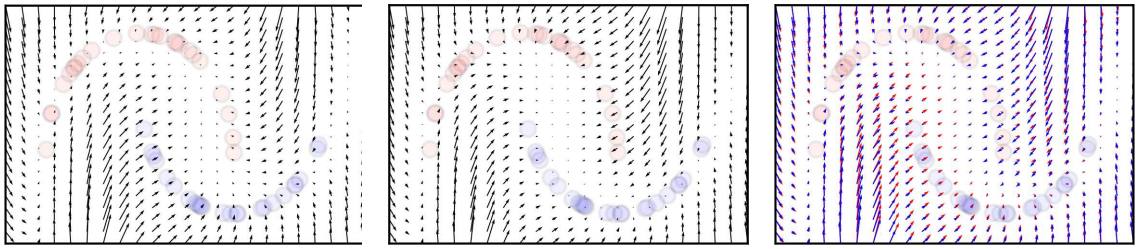


Figure 8: Two moons dataset, along with a reconstruction from M2 model. $q(y|z)$ shows the classification - the background is colored with respect to the classification confidence. The last figure shows the classification based on the reconstruction loss. The figure was created during training with annealed KL divergence, and the classifier achieves almost accuracy of nearly 100%.

In order to better understand the reconstruction loss based classifier, we show the inner workings of the reconstructions for each class in Figure 9, by presenting a vector field. Each vector corresponds to a reconstruction of the datapoint at its base. In a) we assume that the datapoint belongs to the red class, and in b) to blue class. In c) we show the figures combined, with respective colors representing the assumed class. In the background, we plotted the original dataset as a point-of-reference.

In the plots, the direction of the arrow shows the direction of the reconstruction. The length of the arrow is proportional to the reconstruction loss, yet the arrows do not have the exact scale of reconstruction error, in order to keep the figure readable. Note that dots appearing in those figures mean that reconstruction was (close to) perfect. In the panel c) we show the combined reconstructions - note that the longer arrows correspond to higher error.



(a) Red class reconstruction (b) Blue class reconstruction (c) Reconstruction combined

Figure 9: Reconstruction of the dataset during training, with annealed KL divergence. The direction of the arrows indicates the direction of the reconstruction, and the length of each arrow is proportional to the reconstruction error (Euclidean distance) between the original and reconstructed point.

Below, we present similar figures, this time with fully incorporated KL divergence (regularizer). We can see that the classification boundaries dramatically change. The figures also confirm that the generative loss leads to a classifier which makes a "soft" assignment based on the reconstruction loss.

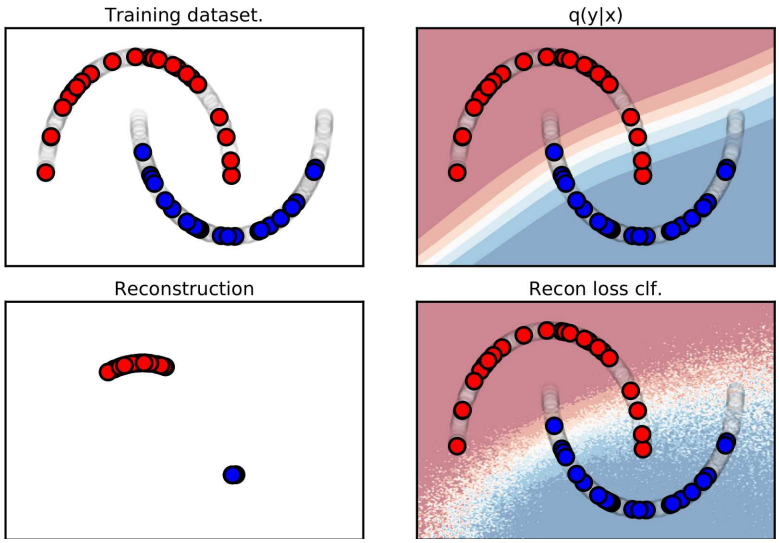
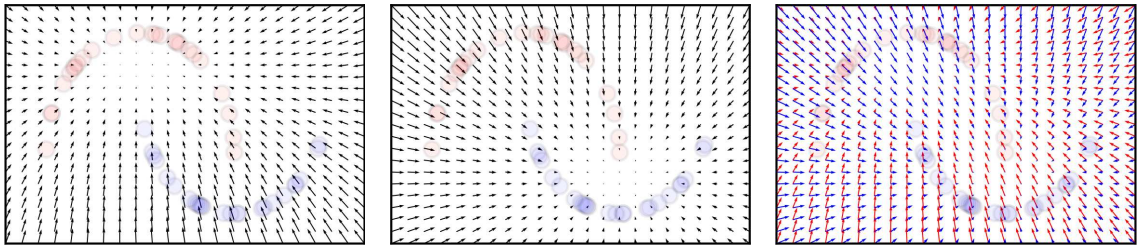


Figure 10: Two moons dataset, along with a reconstruction from M2 model. $q(y|z)$ shows the classification - the background is colored with respect to the classification confidence. The last figure shows the classification based on the reconstruction loss.



(a) Red class reconstruction (b) Blue class reconstruction (c) Reconstruction combined

Figure 11: Reconstruction of the dataset after training, with KL divergence fully incorporated in the loss. The direction of the arrows indicates the direction of the reconstruction, and the length of each arrow is proportional to the reconstruction error (Euclidean distance) between the original and reconstructed point.

3.2 Realistic evaluation of SSL models

In the semi-supervised learning setup, we usually do not have access to many labels. Therefore, as noted by Oliver et al. [11], validation sets cannot be as large as in the usual fully supervised setups. We performed some experiments in which we selected the classifier based on the lowest achieved loss, as opposed to the best performance on the validation set. We present the results in Figure 12.

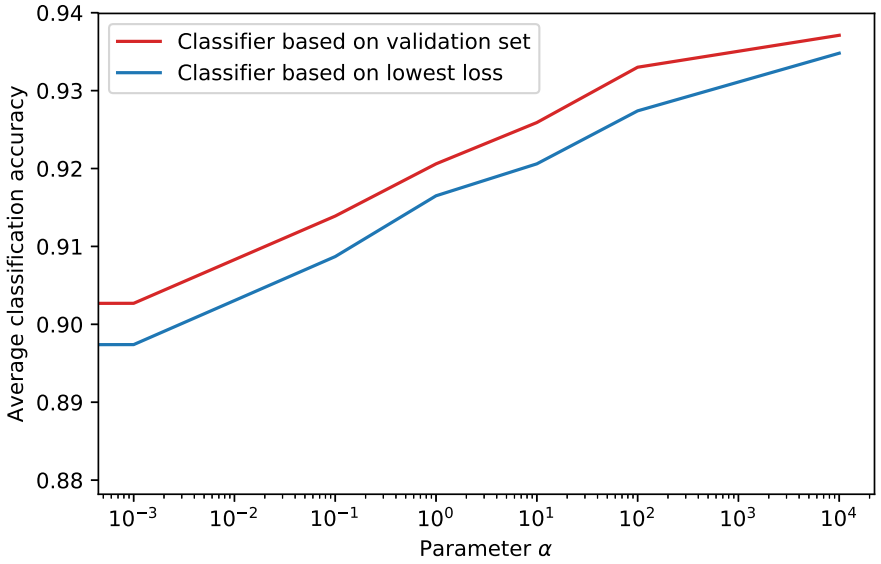


Figure 12: Accuracy comparison of classifiers chosen based on the validation set compared to classifiers chosen based on the lowest loss.

We can see that model selection based on the validation (early stopping) does improve the accuracy, yet the difference in performance isn't large. What is more, the two methods of selection similarly benefit from the large discriminative loss. It could be interesting to see when such a model overfits, i.e. the classification performance drops even though the model's loss is also becoming lower.

References

[1] David Blei, Rajesh Ranganath, and Shakir Mohamed. Variational inference: Foundations and modern methods.

[2] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

[3] Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

[4] Samuel Gershman and Noah Goodman. Amortized inference in probabilistic reasoning. In *Proceedings of the annual meeting of the cognitive science society*, volume 36, 2014.

[5] Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224, 2018.

[6] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[7] Diederik P. Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *NIPS*, pages 3581–3589, 2014.

[8] 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*, pages 4743–4751, 2016.

[9] Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Improving semi-supervised learning with auxiliary deep generative models. In *NIPS Workshop on Advances in Approximate Bayesian Inference*, 2015.

[10] Todd K Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13 (6):47–60, 1996.

[11] Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. In *Advances in Neural Information Processing Systems*, pages 3235–3246, 2018.

[12] Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *Artificial Intelligence and Statistics*, pages 814–822, 2014.

[13] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[14] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *ICML*, 2016.

[15] Lilian Weng. Flow-based deep generative models. 2018.