Improving Variational Autoencoders

COMP 551 - Group 43

Xavier Sumba	Hafizur Rahman	Roger Madhu
260900337	260905565	260900333

April 18, 2019

Abstract

Variational autoencoders (VAE), a generative modelling approach to generate new data, is employed to learn the latent representation in a lower dimensional space. In this report, our aim is to understand, implement and develop improvements for VAE. We propose and explore different approaches to modify the baseline. First, we implemented the baseline with the original VAE model and then proposed six different extensions to check the improvements. Our results show that Denoising VAE, Conditional VAE and VAE with a CNN have the best performance in learning the true distribution, but Importance Weighted autoencoder is capable to create sharper examples.

1 Introduction

Generative models aim at learning the true distribution of an unlabelled dataset and are capable of generating new data points after training is performed. This means that these algorithms need to have a good understanding of the learned data in order to produce new data with some variations. However, in complex cases it is not possible to learn the true distribution of the data since there is an intractable posterior distribution. Therefore, an approximation has to be made. Common techniques of approximate inference are Expectation Propagation, Markov Chain Monte Carlo, and Variational Inference. This techniques usually require to derive a graphical model by hand, and it can be a laborious task when the distributions are not conjugate. In addition, using a neural network to learn the parameters of a distribution can be arduous yet variational autoencoders (VAE) [17] make this possible.

First, remember the traditional autoencoders [9], which can only capture the latent representation of the high dimensional input features into a lower dimension space. However, it is not possible to generate samples with some variability since they do not necessarily learn the distribution; hence the decoder is unable to generate samples close to the inputs. Nonetheless, VAEs are a Bayesian approach that can learn a distribution for the encoder and decoder. Additinally, this model can approximate intractable posteriors using neural networks. In other words, VAEs can train a generative and inference model in a forward pass, and scale very well to large datasets.

In this paper, we focus on VAE, as well as analyze its potential to generate new samples compared to other models introduced recently. Firstly, we reproduce the original VAE paper [17] and perform hyper-parameter tuning. Next, we see its improvement with a convolutional neural network (CNN) instead of a feed-forward neural network (FNN) as introduced in the original paper. Finally, we implement extensions of VAEs in order to get a broader insight into state-of-the-art and compare its efficacy and performance.

2 Background and Related Work

Since our work is based on VAEs, we give a brief review. Additionally, we describe the related work where we got insights for the improvement and understanding of VAEs.

VAEs [17] aims to approximate the marginal likelihood of the data by learning a latent representation $\mathbf{z}$ of an observation $\mathbf{x}$ in a Bayesian approach. The generative model $p_\theta(x \mid z)$ (also interpreted as a decoder) can be expressed with the observations given a latent variable with a prior on the latent variable $p_\theta(z)$. Additionally, an approximate distribution of the latent variable given the observations is introduced $q_\phi(z \mid x)$ or known as the encoder. We want to make the true posterior $p_\phi(z \mid x)$ as close as possible to $q_\phi(z \mid x)$, so we use Kullback-Leibler (KL) divergence to measure their closeness.

$$KL(q_\phi(z \mid x) \,||\, p_\theta(z \mid x)) = KL(q_\phi(z \mid x) \,||\, p_\theta(z)) - \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x \mid z)] + \log p_\theta(x) \tag{1}$$

The KL term on the LHS of equation 1 is intractable and our aim is to learn the true distribution of the dataset and for that we need to maximize the log-likelihood of generating the real data and minimizing the KL divergence between the true and approximate posterior. Therefore, we can construct a bound since we want the similarity between p and q to be close to zero. This term is the variational lower bound or evidence lower bound (ELBO) ($\mathcal{L}$) for $\log p_\theta(x)$.

$$\log p_\theta(x) \geq \mathcal{L}(\theta, \phi) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x \mid z)] - KL(q_\phi(z \mid x) \,||\, p_\theta(z)) \tag{2}$$

Then optimizing the ELBO, $\max_{\theta,\phi} \mathcal{L}(\theta, \phi)$, is the same as optimizing the marginal log-likelihood. The choice of the approximate distribution is important for the quality of the ELBO. In [17], the authors take a Gaussian distribution for $q_\phi(z \mid x)$ whose mean and variance are parameterized by neural networks using x as input.

Finally, since VAEs are built on top of other ideas that have been a key component in Bayesian inference, we point the reader to resources that have been useful for our understanding and and improvement of VAEs. First, variational inference is a deterministic method to approximate expectations, and an excellent review can be found in [1]. Similarly, Markov Chain Monte Carlo [24] is another popular method to perform approximations by sampling. These methods had been used for years for Bayesian inference. However, there are two techniques that have allowed this methods to scale to large datasets, the reparameterization trick [17] and REINFORCE [10]. An intuitive introduction about VAE can be found in [6].

3 Dataset and Setup

It is a common practice to use the MNIST dataset [19] for generative models whether to test new algorithms or define a benchmark. MNIST consists of a set of images of 28x28 pixels containing individual handwritten digits in black and white. The dataset contains 60,000 training examples and 10,000 test examples. Additionally, we binarized the MNIST dataset since it contains continuous values. We used a custom approach, which consists in setting to zero any pixel greater than zero. We also tested the with the binarized MNIST[1] proposed in [18]. However, we decided to use the custom approach since in the original VAE paper [17] the authors use an inversion of colors; for instance, background white and digit black. However, the binarization method does not have a significant impact on the results.

[1] `http://www.dmi.usherb.ca/~larocheh/mlpython/_modules/datasets/binarized_mnist.html`

Similar to the original VAE paper, we did experiment with the Frey Face[2] dataset which contains almost 2,000 images of Brendan Frey's face. The images are taken from a sequential frame of a video and have a size of 20x28 pixels. It contains continuous values for each image. We normalize the pixel values to be in the range between $[0, 1]$. We hold-out 10% of the images for testing.

4 Proposed Approach

This section describes the baseline, which is the implementation and reproduction of the results of the VAE paper [17]. Additionally, we describe the extensions and improvements done. First, we develop simple improvements by fine-tuning parameters and changing the architecture of the neural network. Finally, since we wanted to take advantage of this project to have an idea of the state-of-the-art in the research of VAE, we implement extensions or improvements from other research papers. For each paper, we give a brief description of the contribution and point the reader for more details found in the original publication.

4.1 Baseline

We implement the VAEs paper [17] on its fully for both discrete and continuous data. We use the same settings as in the original publication with a fewer number of passes over the data. In all our experiments with MNIST dataset, we use 100 epochs due to computational limitations. Additionally, the neural networks for the encoder-decoder distributions have an equal number of hidden units. Finally, we execute experiments for different dimensionalities of the latent space.

Specifically, for the experimentation using continuous data, we use the Frey Faces dataset that contains continuous pixel values ranging between $[0-1]$. We used the same settings as proposed in the paper. For instance, for the generative model and variational approximation use an isotropic Gaussian distribution. Samples are taken from a standardized Gaussian distribution. On the side of the neural network, we use a FNN with 200 hidden units for the encoder-decoder with *tanh* activations. In the output of the decoder, we use a *sigmoid* activation for the mean and a *tanh* activation for the variance. Additionally, we take mini-batches of 100 and since the dataset is small, we give 3000 passes over the data taking one sample for each example $L = 1$.

The training of the experiment with continuous data is as follows: i) the weights of the neural network are randomly initialized; ii) we use Adagrad [7] optimizer with no weight decay; iii) first, we train the model with a small number of epochs to select the right learning rate (approximately 50) and once found we train the model for longer (3000 epochs). Since we are training the model using different latent spaces, $N_z = \{2, 5, 10, 20\}$, we search for the appropriate learning rate in the first passes. The values used for the initial learning rate are $\{1 \cdot 10^{-1}, 1 \cdot 10^{-2}, 2 \cdot 10^{-2}\}$. This training procedure is similar for all the experiments with a difference in the optimizer or learning rate values.

There is a difference in our implementation with the original paper; we added a *tanh* activation for the output of the variance in the decoder. This decision was because we encountered gradients exploding. This is because we wanted to compare the true value of the ELBO, so only a MSE loss cannot be used (the full derivation of the ELBO can be found in Appendix A.1). As training progress, the value of the variance increases, and this results in gradients exploding. In order to solve this problem, first, we used gradient clipping but did not help. So, we constrained the values of variance with a *tanh* activation.

Further, we use the binarized MNIST for the experimentation with discrete data. The settings are similar to the ones used in the continuous experimentation. Except that the encoder-

[2] `https://cs.nyu.edu/~roweis/data.html`

decoder FNN use a 500 hidden unit layer, and the decoder uses a Bernoulli distribution with a *sigmoid* activation function. As stated before, training proceeds with mini-batches of 100 and 100 passes over the data. We compare 5 latent spaces, $N_z = \{3, 5, 10, 20, 200\}$.

For the rest of this work, we continue the experimentation and improvements using only the discrete data, but the same procedure translates to the continuous data. This decision was done due we can cover a broader number of ideas proposed in research papers and experiment with them. Additionally, in some of those papers, the full derivations are not included, so we needed to derive them by hand.

4.2 Extensions and Improvements

4.2.1 Fine-tunning

For comparison purposes, we did not change the number of hidden units in the encoder-decoder. We only aim at fine-tuning the baseline and look to answer two questions: i) is there any improvement? ii) when does the model start to overfit?

We first replaced all the *tanh* activations with ReLU [23] activations. The Adagrad optimizer was also changed. We use Adam [16] optimizer, and we selected an initial learning rate in the range between $[1 \cdot 10^{-1}, 1 \cdot 10^{-5}]$, $\beta_1 = 0.9$, $\beta_2 = 0.99$ and no weight decay. Additionally, we tried two alternatives to weight initialization, Xavier [8] and He initialization [11], but there was not a significant improvement. For the latent spaces $N_z = \{5, 10, 20, 200\}$ an initial learning rate of $1 \cdot 10^{-3}$ was chosen and a learning rate of $1 \cdot 10^{-2}$ for the latent space $N_z = 3$. As we will see in section 5, this configuration slightly improves the values of the ELBO.

Additionally, we executed the model for 1,600 passes over the data to determine when it will start overfitting. We noticed that the test error was improving along with the training error. There was not overfitting, but there was not a compelling improvement in the reconstruction and generation of examples. Our intuition is that since VAE is an unsupervised task there is always a room for improvement. Additionally, this might due the cost function introduces an expected reconstruction error that plays the role of a regularizer. In other words, it makes sure that the posterior is not way too different from the prior (see equation 2).

4.2.2 CNN VAE

Since we are experimenting with images, we replaced the FNN with a CNN architecture to take advantage of the local connectivity and parameter sharing that they have. In other words, we use a CNN architecture to exploit the nature of the images. We tried different combinations with max pooling, unpooling, and batch normalization. We only added two convolutional layers in the encoder and two deconvolutional layers in the decoder. We tried different combinations between activations, by alternating Leaky ReLU [11] or ReLU.

After some experimentation, we added batch normalization between each convolutional and deconvolutional layers. For the encoder, we use Leaky ReLUs activations and for the decoder we use ReLUs activations. The final output of the model has a sigmod activation function. Find the complete description of the architecture (i.e. filter sizes, strides, padding) in Appendix A.2. We used the following learning rates $\{1 \cdot 10^{-2}, 1 \cdot 10^{-2}, 1 \cdot 10^{-2}, 1 \cdot 10^{-2}, 1 \cdot 10^{-3}\}$ for each latent space $\{3, 5, 10, 20, 200\}$ respectively. The rest of the settings and hyperparameters are the same as the ones detailed in section 4.2.1.

4.2.3 Conditional Variational Auto-Encoders (CVAE)

In VAE, we do not have the control over the data generation process. Conditioning VAE with specific label can solve this problem. CVAE models the latent variables and data at the same

time conditioned on some random variables [27]. Hence, CVAE can be taken as an exploration of traditional VAE with guided outputs. ELBO calculation for CVAE is similar to VAE with an exception of a label value y which we are conditioning on in our model.

$$\mathcal{L}(\theta, \phi; z|x, y) = -KL(q_\phi(z|x, y)||p_\theta(z|x)) + \mathbb{E}_{q_\phi(z|x,y)}[\log p_\theta(y|x, z)] \tag{3}$$

We trained our model to maximize the conditional log-likelihood using the framework of stochastic gradient variational bayes (SGVB). The training process was done using the same settings from section 4.2.1. Inspired by [27], we examined our baseline VAE with a condition to get structured outputs. For MNIST digits there are 10 labels. Each image has 784 features and additionally 10 more one-hot encoded features are added, making a total of 794 features. For example, for generating samples of 1 we passed $c = [0, 1, 0, 0, 0, 0, 0, 0, 0, 0]$ to the generator network along the 784 features for each epoch. In CVAE, the lower bound is comparatively lower than traditional VAE as we are training the model under specific label which has its own distinctive distribution. For the same reason, the reconstructed and sampled images (see A.5) are better than the baseline model.

4.2.4 Importance Weighted Autoencoders (IWAE)

One of the problems of VAE is that makes a strong assumption since the variational posterior can approximate a single mode. However, VAE models need a functional posterior which can be predictable with a linear network, for example. Importance Weighted Autoencoders [3] relax this assumption using a tighter lower bound on the marginal likelihood. The new bound looks like the following expression $\mathbb{E}[\log \frac{1}{k} \sum_i w_i]$, where the w_i are the unnormalized importance weights of the joint distribution. The bigger the value of k (more samples), the tighter the bound. The traditional VAE can be derived when $k = 1$.

In our implementation of IWAE, we use two stochastic layers with $h_1 = 200$ and $h_2 = 100$ hidden units, and analyze different latent space representations $N_z = \{3, 5, 10, 20, 200\}$. We add *tanh* activations throughout the neural network and a *sigmoid* activation function in the output of the decoder (for more details see the complete architecture in the Appendix C of [3]). We use the settings reported in the IWAE paper that have higher log-likelihood results. For instance, we use 50 samples (k) and a learning rate of $1 \cdot 10^{-3}$. Additionally, we follow their advice in replicating each training example k times within a mini-batch to scale the number of operations linearly.

IWAE relies on computing the gradient of the log unnormalized weights for the forward and backward passes in backpropagation. This gradients are normalized and averaged between the k samples. We need to make a distinction that only the log weights contribute to the gradient, so when computing the ELBO this has to be considered. We add the full derivation of $\log w$ in Appendix A.3. A problem encountered during the implementation of the IWAE is that in some cases the $\log p(x \mid h_1)$ goes to infinity since zero-probabilities were obtained hence this caused the gradient to vanish. To diminish this problem, we tried clipping the gradient, but it worked for some passes over the data and the problem reappeared. Therefore, since the $p(x \mid h_1)$ is a Bernoulli distribution and is equivalent to a negative binary cross-entropy loss, we use the implementation of cross-entropy from PyTorch that assigns zeros (i.e. $\log 0 = 0$).

4.2.5 Denoising Variational Autoencoder (DVAE)

Traditional denoising autoencoders [29] add noisy inputs and evaluate a cost function between the output and the original input, so the neural network is able to reconstruct the original input when given a corrupted input. This idea can be translated to VAEs using the denoising criterion

[26]; however, the lower bound obtained is intractable. The Denoising Variational Autoencoder [13] explores this idea by introducing the denoising variational lower bound. Additionally, DVAE creates a tighter lower bound, and as a result of this, the posterior can capture multiple modes; similar to IWAE. However, the DVAE is sensible to the noise distribution chosen. In general, the DVAE adds noise to the input and the stochastic layer while VAE adds noise only to the stochastic layer, the denoising lower bound is expressed in equation 4, which can be approximated with Monte Carlo sampling.

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{q_\phi(z|\tilde{x})}\mathbb{E}_{p_\theta(\tilde{x}|x)}[\log \frac{p_\theta(x, z)}{q_\phi(z \mid \tilde{x})}] \tag{4}$$

Hence, by looking at the denoising variational lower bound, we can notice that training a DVAE is similar to training a traditional denoising autoencoder. We provide a corrupted input $\tilde{x}$, sample a latent representation z, and try to reconstruct the original input x. This procedure translates to training a VAE with a corrupted input $\tilde{x}$.

We trained the DVAE using the settings from section 4.2.1. As proposed in the DVAE paper, we add a *salt and pepper* noise to the original inputs. We experimented with different noise levels for the noise distribution chosen $\{0.05, 0, 1, 0.15, 0.25, 0.4, 0.5\}$. We notice that the performance of the model is sensible to the injected noise level. Values greater than 0.2 harm the performance of the model while values between 0.05 and 0.15 overcome the results obtained by traditional VAE. In general, 0.05 gives a better performance in all the latent spaces experimented (i.e. $\{3, 5, 10, 20, 200\}$).

4.2.6 Beta Variational Auto-Encoders (β-VAE)

β-VAE, introduced by [12], is a modification of the VAE that focuses on the disentangled latent factors. When the latent variable is highly sensitive to one generative factor and invariant to other factors then the representation is disentangled. An adjustable parameter β is incorporated to balance the latent channel capacity and independence constraints with reconstruction accuracy as shown in equation 5.

$$\mathcal{L}(\theta, \phi; \mathbf{x}, \mathbf{z}, \beta) \approx \mathbb{E}_{q_\phi(z|x)}[\log p(z \mid x)] - \beta KL(q_\phi(z \mid x) \mid\mid p(z)) \tag{5}$$

Setting the hyperparameter $\beta = 1$ represents the original VAE proposed by [17]. In order to learn the disentangled representations, [12] proposed to set $\beta > 1$ which limits the learning capacity of $\mathbf{z}$. This restriction combined with the compulsion to maximize the log-likelihood of the training data $\mathbf{x}$, enables the model to learn the most efficient learning representation of the data [4]. For training the model with the MNIST dataset [19], we used the same settings as the original VAE model with a change in the loss function according to 5. As indicated in [12], we set the value of hyperparameter, $\beta = 4$. $\beta-$VAE, being state of the art generative model [5] was unable to get better accuracy than the traditional VAE. One possible reason for that might be the simplicity of our dataset where the model was incapable to learn the disentanglement. However, it is proven that for complex datasets like in [20], this model beats the traditional model [12]. Because of time constraints and lack of computational power, we restricted ourselves to MNIST digits dataset for the model.

5 Results

This section presents the results obtained by performing an extensive hyper-parameter tuning and implementing extensions or improvements of VAE. First, we give the results for the continuous experimentation, and later the obtained results for the discrete case.

The Frey Faces dataset used for the continuous experimentation is capable to reconstruct and generate examples. However, due to the arbitrarily decision of using a *tanh* in the variance of the decoder, the values of the likelihood lower bound are scaled differently to those of the original paper. The values obtained for the likelihood lower bound are $\{236.67, 241.69, 243.72, 245.74\}$ for the latent space $\{2, 5, 10, 20\}$ respectively. Some examples of the images reconstructed and generated can be found in the Appendix A.4. We notice that the model has problems with faces on which Frey is blinking or smiling.

Further, our first contribution to the discrete experiment is the improvement of VAE by hyper-parameter tuning and using a CNN architecture. In general, the use of a CNN for the encoder and decoder seem to work pretty well since in all the latent spaces there was a significant improvement. So, using a more powerful neural network that allows to encode certain properties of a image translates to obtaining a better representation of the latent space. Equally, there is an improvement when performing hyper-parameter tuning. Figure 1 illustrates the improvement of the ELBO as the model sees more data during training. We can notice that in the improvements implemented outperform the traditional VAE.

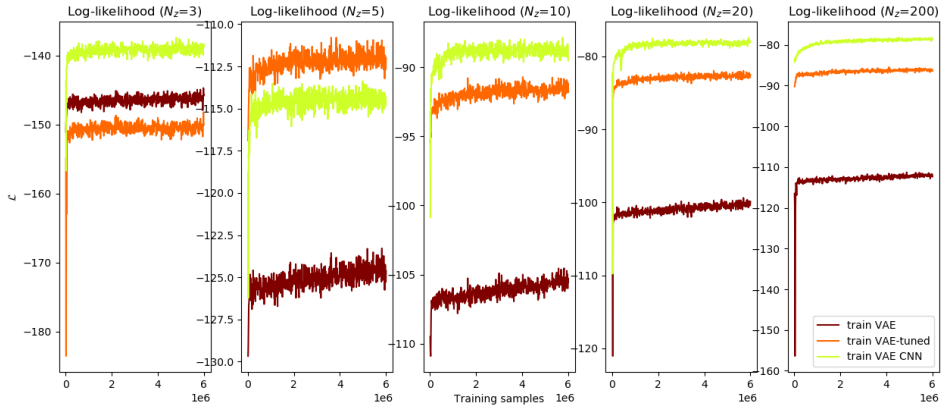

Figure 1: Comparison of the lower bound for VAE, VAE-tuned, and VAE CNN.

Finally, table 1 shows a comparison of all the improvements and extensions implemented by comparing the values of the variational lower bound. We can notice that the ELBO of our implementation is really close to the original implementation of VAE. This difference is due to the number of passes over the data and the binarization method since the authors do not specify the method chosen. We can additionally notice that the values of IWAE[3] are a bit high even though we used a high number of samples ($k = 50$). This might be a result of cutting some probabilities with values $\log 0 = 0$. We can observe that VAE CNN capture better the representations with a high latent capacity and DVAE performs better at capturing lower latent capacities. Addtionally, CVAE performs better than the other models, and our intuition is that showing more features to the model, helps to construct better latent representations. The reconstructed and generated images for CVAE are a bit sharper and our conclusion is that, since we are training the model with a condition on the data label for each instance, the model learns a better distribution when trained in a semi-supervised manner.

***Discussion*:** In general, our implementations are capable to capture lower latent capacities effectively; for an example see the figure in the Appendix A.6. However, we wonder what would be the behavior if we increase the latent capacity to the point of the input size or even bigger

[3]Al alternative explanation is that we did a mistake in the derivation of the variational lower bound, see the resulting expression in Appendix A.3.

Model / N_z	**3**	**5**	**10**	**20**	**200**
VAE original	-136.00	-115.00	-99.00	-95.00	-98.00
VAE implementation	-137.64	-116.93	-95.22	-87.95	-90.31
VAE tuned	-140.68	-110.32	-89.78	-80.97	-83.41
VAE CNN	-134.99	-113.61	-94.41	-77.78	**-75.47**
IWAE	-124.89	-122.45	-177.95	-114.72	-107.02
DVAE	**-109.48**	-102.58	-98.26	-92.89	-94.50
$\beta-$VAE	-154.17	-141.21	-136.14	-137.06	-138.79
CVAE	-116.27	**-98.99**	**-80.85**	**-75.43**	-76.39

Table 1: Log-likelihoods on binarized MNIST for the improvements and extensions done (the lower, the better).

Our intuition is that the model will be able to retain more information but more likely to overfit.

We also present some examples of the reconstruction and generation of data in the Appendix A.5. We notice that most of the images generated and reconstructed are blurry. However, the IWAE is able to create nice reconstructions and this is due to the number of samples used ($k = 50$) but it cannot generate examples that good. Models also seem to confuse 8s with 3s and 4s with 9s or vice-versa. In that sense, we wonder if there will be an improvement by doing data augmentation. But in general, our experimentation show that since the models are not expressive enough, they cannot capture values of the true distribution, and as a consequence images are blurry. We think that this can be improved by selecting a loss function that is able to capture this small variations or using more expressive models.

6 Conclusions and Future Work

After emerging as a promising generative model, VAE is being researched and expanded by many researchers. Our aim was to implement as many models as possible to improve the baseline rather than selecting computationally expensive architecture or dataset.

There is still a lot of room for improvement since the research done in VAEs is extensive. Furthermore, we point other recent works that we think it might improve our results, see the Semi-amortized variational autoencoder [15], Adversarial Variational Bayes [22], VAE with a VampPrior [28], and From Variational to Deterministic Autoencoders [25]. Specially combining VAE with Generative Adversial Networks [21] can allow VAEs to create sharper images. Additionally, this project has inspired us to improve the Dirichlet VAE [14] using a distribution with more degrees of freedom and as a consequence improving the model and allowing us to create extensions to models such as the Correlated Topic Model [2].

7 Statement of Contributions

Xavier contributed to the experimentation, research, writing and editing. Hafiz contributed to the experimentation, research and writing. Roger contributed to the experimentation.

References

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

[2] Blei, D. M., Lafferty, J. D., et al. A correlated topic model of science. *The Annals of Applied Statistics 1*, 1 (2007), 17–35.

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

[4] Burgess, C. P., Higgins, I., Pal, A., Matthey, L., Watters, N., Desjardins, G., and Lerchner, A. Understanding disentangling in β-vae. *CoRR abs/1804.03599* (2018).

[5] Chen, T. Q., Li, X., Grosse, R., and Duvenaud, D. Isolating sources of disentanglement in variational autoencoders, 2018.

[6] Doersch, C. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908* (2016).

[7] Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research 12*, Jul (2011), 2121–2159.

[8] Glorot, X., and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (2010), pp. 249–256.

[9] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

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

[11] He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 1026–1034.

[12] Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR* (2017), OpenReview.net.

[13] Im, D. I. J., Ahn, S., Memisevic, R., and Bengio, Y. Denoising criterion for variational auto-encoding framework. In *Thirty-First AAAI Conference on Artificial Intelligence* (2017).

[14] Joo, W., Lee, W., Park, S., , and Moon, I.-C. Dirichlet variational autoencoder, 2019.

[15] Kim, Y., Wiseman, S., Miller, A. C., Sontag, D., and Rush, A. M. Semi-amortized variational autoencoders. *arXiv preprint arXiv:1802.02550* (2018).

[16] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. In *ICLR* (2015).

[17] Kingma, D. P., and Welling, M. Auto-encoding variational bayes. In *ICLR* (2014).

[18] Larochelle, H., and Murray, I. The neural autoregressive distribution estimator. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (2011), pp. 29–37.

[19] LeCun, Y. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/* (1998).

[20] Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)* (2015).

[21] Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., and Frey, B. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644* (2015).

[22] Mescheder, L., Nowozin, S., and Geiger, A. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (2017), JMLR. org, pp. 2391–2400.

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

[24] Neal, R. M. Probabilistic inference using markov chain monte carlo methods.

[25] Partha Ghosh, Mehdi S. M. Sajjadi, A. V. M. B. B. S. From variational to deterministic autoencoders. *arXiv preprint arXiv:1903.12436* (2019).

[26] Seung, H. S. Learning continuous attractors in recurrent networks. In *Advances in neural information processing systems* (1998), pp. 654–660.

[27] Sohn, K., Lee, H., and Yan, X. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 3483–3491.

[28] Tomczak, J. M., and Welling, M. Vae with a vampprior. *arXiv preprint arXiv:1705.07120* (2017).

[29] Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning* (2008), ACM, pp. 1096–1103.

A Appendix

A.1 ELBO for Gaussian-VAE

The marginal likelihood lower bound for a Gaussian encoder and a Gaussian decoder is shown in the equation 6. Additionally, we are doing one Monte Carlo estimate (i.e. $L = 1$) from the expectation $\mathbb{E}[\log p(z \mid x)]$. J is the dimensionality of the latent variable (z) and M is the dimensionality of the observation (x).

$$\mathcal{L}(\theta, \phi; \mathbf{x}^{(i)}) \approx \frac{1}{2}\sum_{j=1}^{J}(1+\log((\sigma_j^{(i)})^2)-(\mu_j^{(i)})^2-(\sigma_j^{(i)})^2)-\frac{1}{2}\sum_{k=1}^{M}(\log(2\pi)+\log((\sigma_k^{(i)})^2)+\frac{(x_k^{(i)}-\mu_k^{(i)})^2}{(\sigma_k^{(i)})^2}) \tag{6}$$

A.2 CNN Architecture

We use batch normalization between each layer. In the encoder, we use Leaky ReLU activations and in the decoder, we use ReLU activations with a sigmoidal activation function in the output.

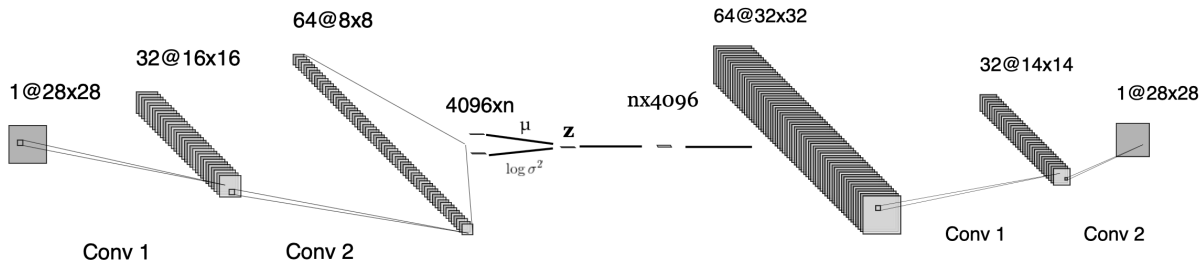

Figure 2: CNN architecture used with VAE.

A.3 ELBO for IWAE

The ELBO for importance weighted auto-encoder is normally defined as follows:

$$\mathcal{L}(\theta, \phi; \mathbf{x}) \approx \mathbb{E}[\log \frac{1}{k}\sum_{i=1}^{k}\frac{p(x, h_i)}{q(h_i \mid x)}] \tag{7}$$

After applying the reparameterization trick, the gradient of the ELBO is the expectation of the average of the normalized importance weights times the derivative of the logarithm of the importance weights as expressed, see equation 8.

$$\nabla\mathcal{L}(\theta, \phi; \mathbf{x}) \approx \mathbb{E}[\sum_{i=1}^{k}\hat{w_i}\nabla \log w] \tag{8}$$

In this work, we use the same distributions provided in IWAE [3]. For instance, zero-mean and unit-variance Gaussian distribution for priors, conditional distributions are Gaussians with a diagonal covariance for continuous data and Bernoulli distribution for binary data. in equation 9, we derive the logarithm of the importance weights ($\log w$) needed to obtain the ELBO.

$$
\begin{aligned}
\log w &= -\log \mathcal{N}(h_1 \mid \mu_{q1}, \sigma_{q1}^2) - \log \mathcal{N}(h_2 \mid \mu_{q2}, \sigma_{q2}^2) \\
&+ \log \mathcal{N}(h_1 \mid \mu_{p1}, \sigma_{p1}^2) + \log \mathcal{B}(x \mid p_x) \\
&+ \log \mathcal{N}(h_1 \mid 0, I) + \log \mathcal{N}(h_2 \mid 0, I) \\
&= \frac{1}{2} \sum_i (\log 2\pi + \log \sigma_{q1_i}^2 + \frac{(h_{1_i} - \mu_{q1_i})^2}{\sigma_{q1_i}^2}) \\
&+ \frac{1}{2} \sum_i (\log 2\pi + \log \sigma_{q2_i}^2 + \frac{(h_{2_i} - \mu_{q2_i})^2}{\sigma_{q2_i}^2}) \\
&- \frac{1}{2} \sum_i (\log 2\pi + \log \sigma_{p1_i}^2 + \frac{(p_{1_i} - \mu_{p1_i})^2}{\sigma_{p1_i}^2}) \\
&+ \sum_i (x_i \log p_{x_i} + (1 - x_i) \log(1 - p_{x_i})) \\
&- \frac{1}{2} \sum_i (\log 2\pi h_{1_i}^2) - \frac{1}{2} \sum_i (\log 2\pi h_{2_i}^2)
\end{aligned} \tag{9}
$$

A.4 Reconstruction and generation of faces of Frey Faces dataset

Figure 3: Reconstruction of faces using the implementation of VAE with continuous inputs using a latent space $N_z = 5$. First row: original input. Second row: reconstruction.

Figure 4: Generation of new examples using the traditional VAE implemented. Images belong to a latent space of $N_z = 5$.

A.5 Reconstruction and generation of digits of MNIST dataset

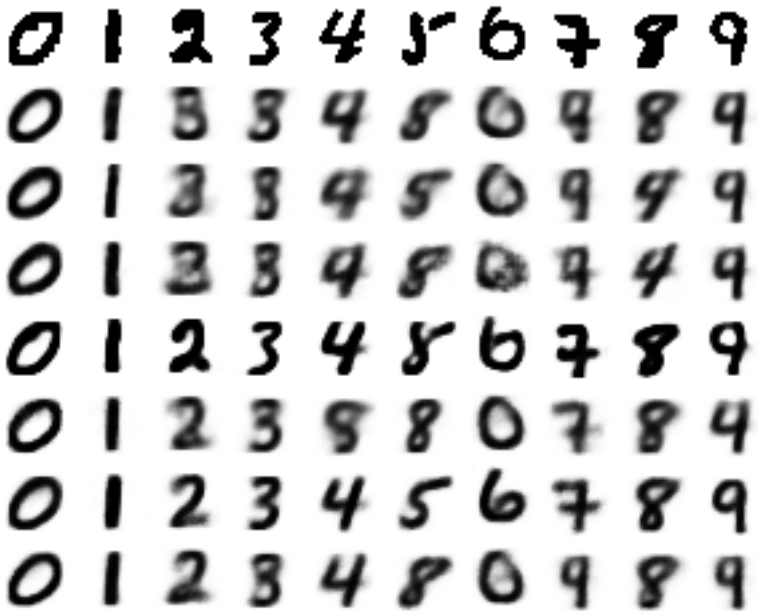

Figure 5: Reconstruction of images with a latent space of $N_z = 3$. On top is the original input and each of following rows is the result of VAE, VAE-tuned, VAE CNN, IWAE, DVAE, CVAE, and β-VAE respectively.

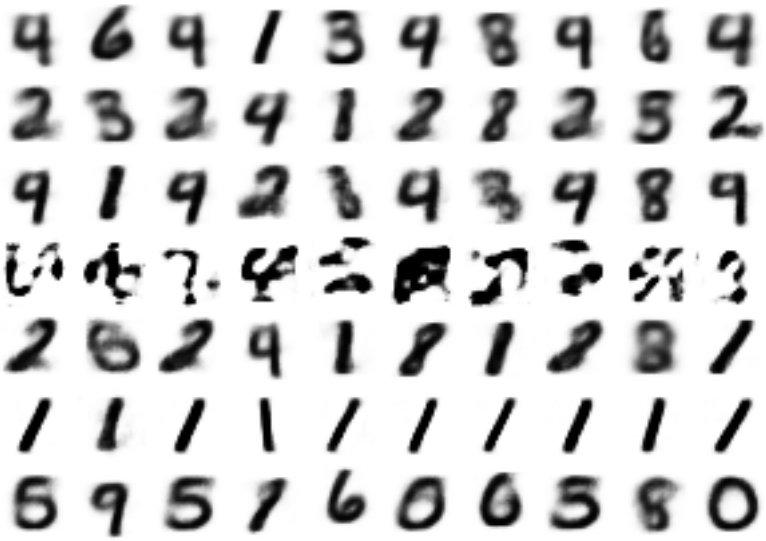

Figure 6: Generation of new digits using latent space of $N_z = 3$. Each of rows represent a sampling with a different implementation of VAEs in the following order: VAE, VAE-tuned, VAE CNN, IWAE, DVAE, CVAE, and β-VAE. We condition CVAE to generate only ones.

A.6 Assignments of latent space $N_z = 3$ by VAE CNN

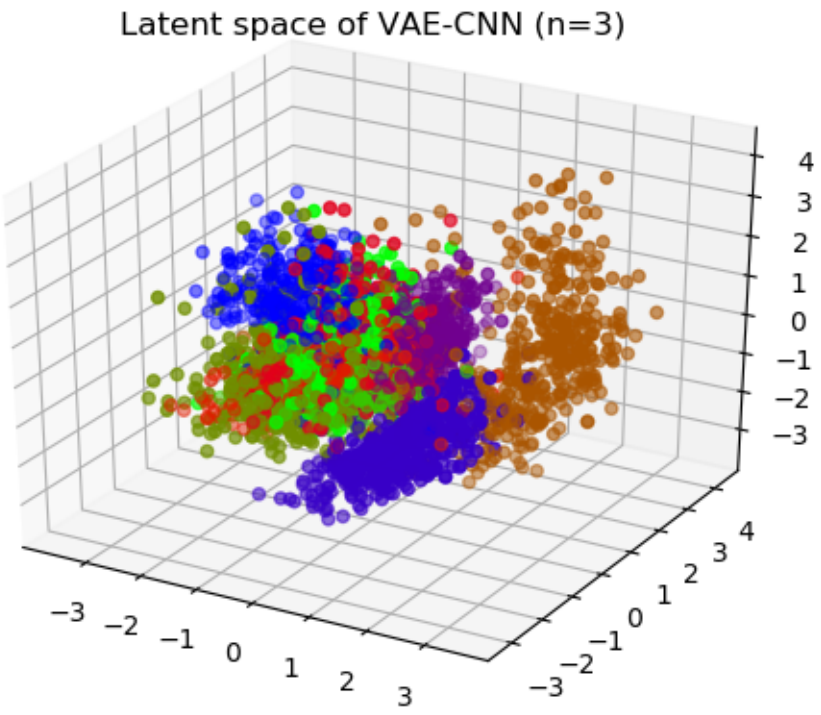

Figure 7: Assignment of latent space captured by VAE CNN using a dimensionality $N_z = 3$.