# CSC412/2506 Course Project, Variational AutoEncoder++

**Feng Chen**
University of Toronto
harrycf.chen@mail.utoronto.ca

**Fanxuan (Cami) Guo**
University of Toronto
cami.guo@mail.utoronto.ca

## Abstract

This project explores how powerful the Variational Auto Encoder (VAE) is and experimented the VAE model with other interesting settings such as training it to encode samples into a larger latent space, training the model given true labels or partial labels, training the model on a different dataset, utilizing the importance weighted autoencoder, replacing Bernoulli likelihood model with Beta likelihood model and making inference given partial digit images.

## 1 Introduction

The Variational AutoEncoder implemented in this project consists of two neural networks, the decoder network, and the encoder network. The task is to train the decoder and encoder to learn a representation of hand-written digits from 0 to 9. Specifically, The encoder takes in hand-written digits and produces a representation of that digit in another latent space in multi-dimensions. The decoder takes those encoded samples from the latent space to re-generate its corresponding hand-written digits. Intuitively speaking, the encoder compressed the digit information, and the decoder re-build digit from the compressed information. To measure the difference between two distributions, the KL Divergence was used as the loss function for optimizing the model parameters and updating them using gradient descent.

## 2 Larger Latent Space

### 2.1 Description

From A3, the encoder and decoder compressed and generated digit samples on a 2D latent space. As the representation encoded in latent space only has two dimensions which are kind of limited for containing more latent information, we try increasing the dimension of latent space to see whether there is a quality improvement of the re-generated hand-written digits.

### 2.2 Visualization of 3D latent space

Below (Figure 1) is the visualization of 3D latent space and 2D latent space. As an additional dimension is added into the latent space, we could clearly see how each group of digit is located in the 3D space which is more expressive than 2D latent space.

### 2.3 Comparison with baseline

Below (Figure 2) are five reconstructed digits the decoder decoded from random samples in 2D latent space (left column) and random samples in 3D latent space (right column). Notice that the right column of reconstructed digits is more realistic than those in the left column. For instance, the digit

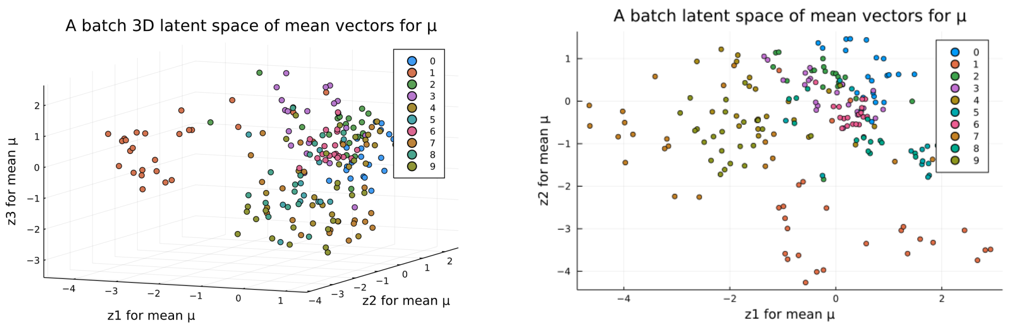


**Figure 1:** A batch 3D (left) 2D (right) latent space of mean vectors for $\mu$

0 in the right column second and fourth row are in a tilted position which seems more natural and closer to human hand-written digits. Whereas, the left column first and last row are less realistic as they only show the basic shape of digit zero without any other interesting features. Training the model in both latent spaces for 5 epochs. The training loss is shown in figure 3, notice that the loss in 3D latent space reaches a lower loss than that in 2D latent space. Therefore, we conclude that increasing the dimension of latent space results in capturing more information and the reconstructed samples show more features.

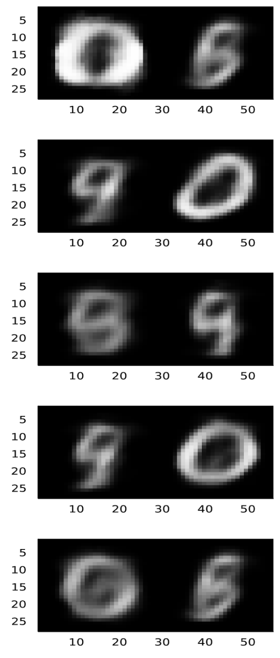
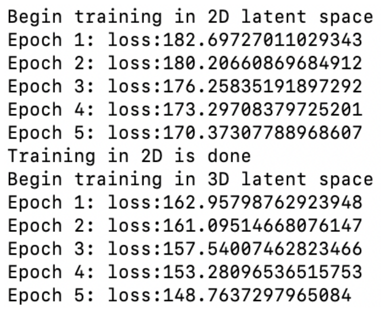


**Figure 2:** Reconstructed digits from 2D (left column) v.s. 3D (right column) latent space



```
Begin training in 2D latent space
Epoch 1: loss:182.69727011029343
Epoch 2: loss:180.20660869684912
Epoch 3: loss:176.25835191897292
Epoch 4: loss:173.29708379725201
Epoch 5: loss:170.37307788968607
Training in 2D is done
Begin training in 3D latent space
Epoch 1: loss:162.95798762923948
Epoch 2: loss:161.09514668076147
Epoch 3: loss:157.54007462823466
Epoch 4: loss:153.28096536515753
Epoch 5: loss:148.7637297965084
```

**Figure 3:** Training loss of 2D (top) v.s. 3D (bottom) latent space

### 2.4 Variance along each latent dimension

Below (Figure 4) are variance plots along each latent dimension for a batch of samples. Notice that as the number of latent dimensions increase, the variance along with each latent dimension increases.

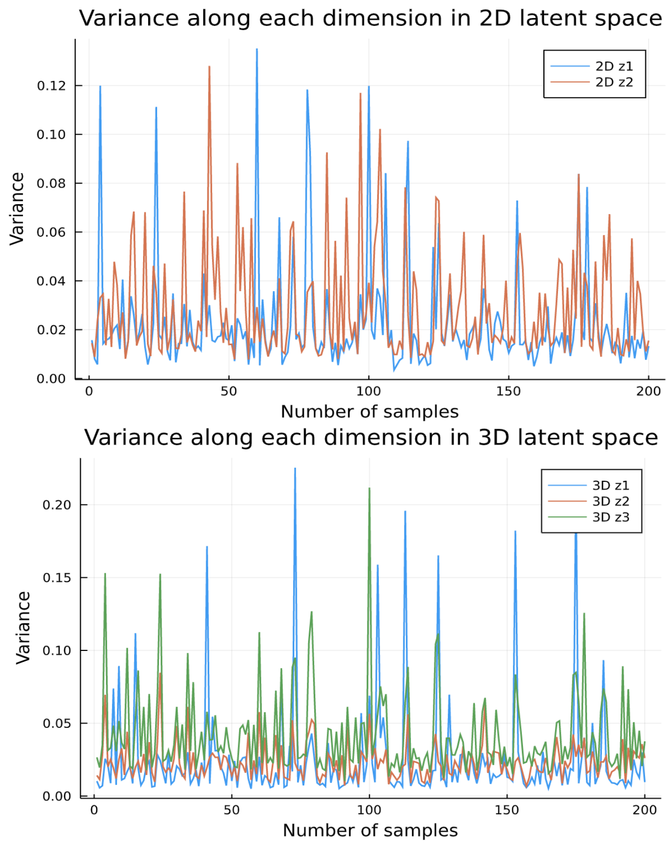


**Figure 4:** Variances in 2D (top) v.s. 3D (bottom) latent space

## 3 Condition on MNIST Digit Supervision

Made some modifications to the training data. The steps are listed below.

- Convert the categorical digit labels to one-hot vectors. For example, digit 0 is converted to a column vector [1;0;0;0;0;0;0;0;0;0].
- Vertically concatenate the one-hot vectors to the original training set
- Add 10 more connections to both input and output layers of the encoder as well as the decoder in order to facilitate the increased the dimension of the training set and dimension of latent space

Model modification is shown in figure 5 below. The new encoder and decoder are fed with true labels in one-hot vector format. Different from the figure, we feed digit 0 to both encoder and decoder.

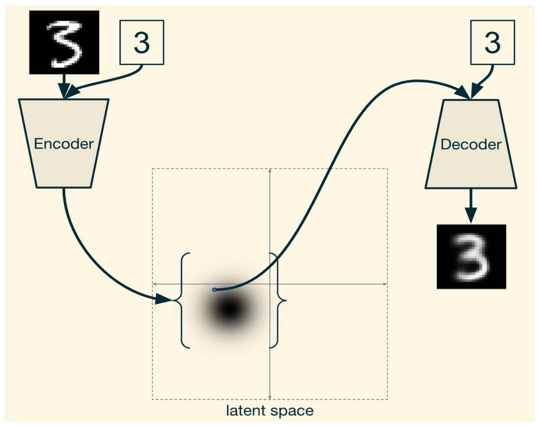


**Figure 5:** New encoder and decoder given labels [1]

Below (Figure 6) is the visualization of the 3D latent space representation given the labels

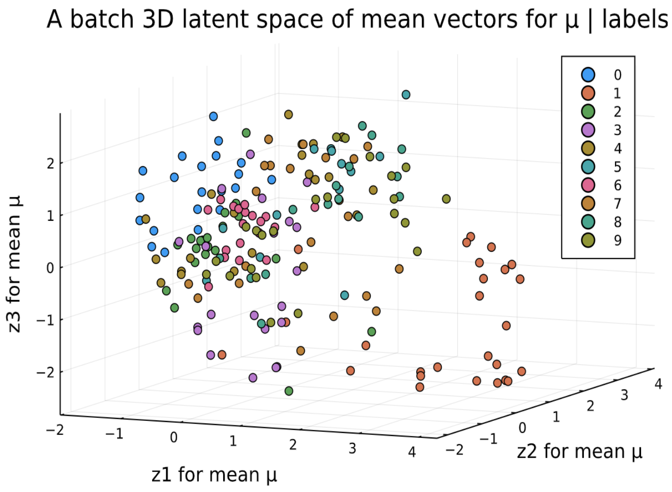


**Figure 6:** 3D latent space representation given labels

In addition to training the model given complete labels, we also experimented with incomplete labels known as "semi-supervised". To make a dataset with missing labels, we randomly choose 30000 samples and change their one-hot label to [0;0;0;0;0;0;0;0;0;0]. The reconstructed digits are shown in figure 7 below. In general, it seems that these digits are similar in shapes and style. However, outputs with complete labels (Figure 7, left five) tend to have distinguishable digit shapes, for example, digit 8 on the second left and digit 9 on the third left compared to digit 8 on the fifth right and digit on the fourth right. Therefore, we conclude that the conditional VAE generates more clear digits with complete labels and blurry digits with incomplete labels.

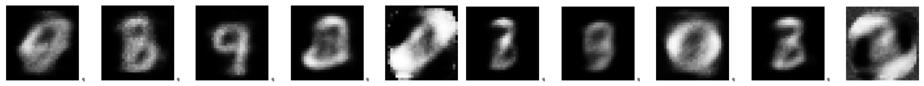


**Figure 7:** Left five: VAE with complete labels, Right five: VAE with incomplete labels

## 4 More interesting data

Training the baseline model with 2D latent space on the binarized FashionMNIST dataset. The reconstructed outputs are shown in figure 8. Training for 3 epochs. It is easy to recognize the cloth type. However, some details are missing.

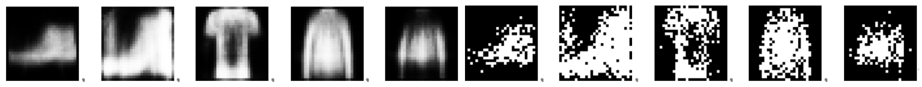


**Figure 8:** Left five: reconstructed results, Right five: random samples

## 5 Optimizing Different Divergences

Applied the Jensen-Shannon divergence [2] to measure the similarity between distributions p and q. The equation of JS divergence (equation 1) is listed below.

$$\begin{aligned} M &= (p+q)*0.5 \\ JSD(p||q) &= \frac{1}{2}D(p||M) + \frac{1}{2}D(q||M) \end{aligned} \tag{1}$$

The outputs using JS divergence are shown in figure 9. Trained the model for 3 epochs. The reconstructed digits are similar to digits generated by the baseline model using KL divergence in figure 2 left column.

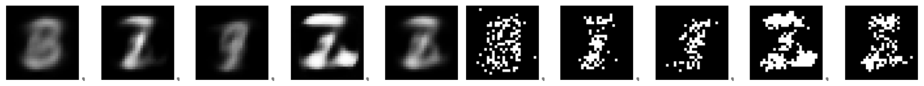


**Figure 9:** Left five: reconstructed results, right five: random samples

## 6 Importance Weighted AutoEncoder

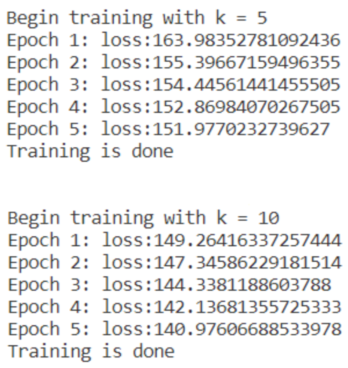
The typical VAE models have 2 strong assumptions about the posterior distribution: (a). the posterior is approximately factorial; (b). the posterior's parameters can be predicted from the observables through a nonlinear regression. VAE harshly penalizes approximate posterior samples which are unlikely to explain the data. Whereas, the importance weighted autoencoder has additional flexibility for posterior distributions that do not fit the VAE assumption [3]. The $k$-sample IWAE and VAE have the same architecture, but it is designed to maximize a different lower bound:

$$\mathcal{L}_k = \mathop{\mathbb{E}}_{z_1,\ldots,z_k \sim q(z|x)} \left[ log \frac{1}{k} \sum_{i=1}^{k} \frac{p(x,z_i)}{q(z_i|x)} \right] \tag{2}$$

To train the log-likelihood estimate, we sample $k$ different $z \sim q(z|x)$ and importance weight them. Then we use the Monte Carlo estimate when we take the gradient of equation (2):

$$\begin{aligned} w_i &= \frac{p(x,z_i)}{q(z_i|x)}, \tilde{w_i} = \frac{w_i}{\sum_{i=1}^{k} w_i} \\ \nabla\mathcal{L}_k &= \sum_{i=1}^{k} \tilde{w_i} \nabla log \frac{p(x,z_i)}{q(z_i|x)} \end{aligned} \tag{3}$$

Figure 10 shows the training loss of $k$ = 5 and $k$ = 10 for 5 epochs. The loss of $k$ = 10 is smaller than the loss of $k$ = 5, which means We get a tighter log-likelihood lower bound when we increase the value of $k$. This matches the theorems "for all $k$, $log_p(x) \geq \mathcal{L}_{k+1} \geq \mathcal{L}_k$" [3].



```
Begin training with k = 5
Epoch 1: loss:163.98352781092436
Epoch 2: loss:155.39667159496355
Epoch 3: loss:154.44561441455505
Epoch 4: loss:152.86984070267505
Epoch 5: loss:151.9770232739627
Training is done


Begin training with k = 10
Epoch 1: loss:149.26416337257444
Epoch 2: loss:147.34586229181514
Epoch 3: loss:144.3381188603788
Epoch 4: loss:142.13681355725333
Epoch 5: loss:140.97606688533978
Training is done
```

**Figure 10:** Training loss of $k$ = 5 v.s. $k$ = 10

Figure 11 is the visualization of IWAE with $k$ = 5 and $k$ = 10. For each $k$ value, we choose the same three digits, reconstruct and make one random sample using our trained model. Notice that compared to the reconstructed digits for $k$ = 5, the reconstructed digits for $k$ = 10 are clearer, cleaner and closer to the original handwritten digit, which means the model captures more features.

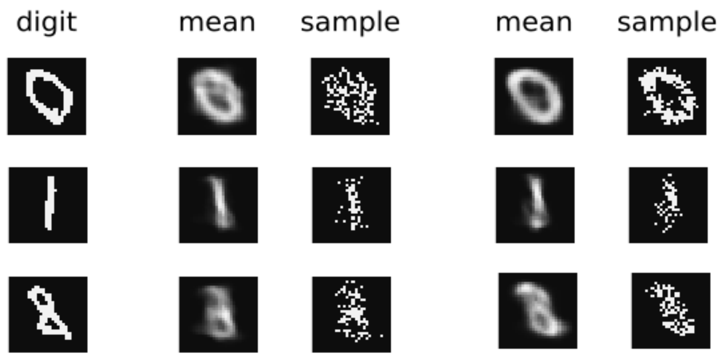


**Figure 11:** Left column: digits, middle two columns: $k$ = 5, right two columns: $k$ = 10

## 7 More Expressive Likelihood Model

Experimented the Beta likelihood model [4] with $\alpha = 2$ and $\beta = 2$ on MNIST in float data format. The beta probability density listed in equation 4,

$$\frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}$$
$$B(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)} \tag{4}$$

Convert the MNIST to float format instead of binary format. Then train the baseline model with 2D latent space for 5 epochs. The reconstructed outputs are shown in figure 12 below. We can see that the reconstructed digits are blurry and hard to distinguish. Due to the pixel values are continuous floating numbers, it is reasonable to choose a continuous probability distribution like beta distribution. Furthermore, with two parameters $\alpha$ and $\beta$, we could choose the shape of distribution depending on the dataset. Here, for calculation simplicity, we experimented with $\alpha = \beta = 2$ which gives a symmetric shape density centered at $\frac{1}{2}$ with 0 skewness.

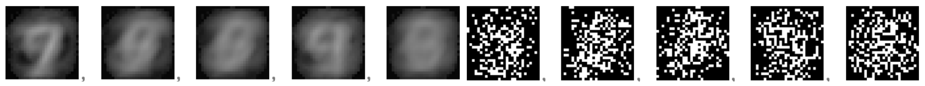


**Figure 12:** Left five: reconstructed results, Right five: random samples

## 8 Inference Questions

Use the baseline model to infer the bottom part given the top part of the digit. Follow the guidance in question 4 of previous year A3 for this course [5]. First, we compute $p(z, top)$ which requires having $p(top|z)$ and the prior $p(z)$. Then we could get the joint density $logp(z, top)$. To approximate $p(z|top)$, use the stochastic variational inference. Randomly initialize the parameter $\phi_\mu$ and $\phi_{log\sigma}$ for the variational distribution $q(z|top)$. Then optimize both parameters $\phi_\mu$ and $\phi_{log\sigma}$ using ELBO. Finally, with optimum parameters, sample z from the approximate posterior. Use the baseline decoder to decode the sample z and convert the result logits means to Bernoulli means of the bottom part. Finally, concatenate it with the original top part. We take digit 0 as an example and the result is shown in figure 13 below. Notice that the bottom part shape seems to follow the shape of the top part. The tilted position of the digit 0 is also easy to recognize.

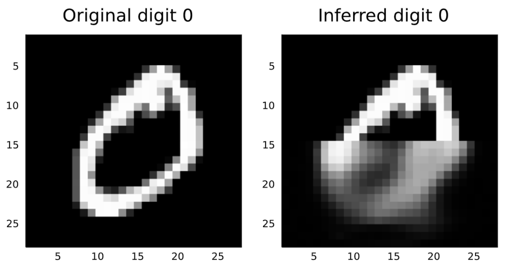


**Figure 13:** Infer the bottom part given the top part

## 9 Discussion & Limitations

**Model architecture selection**: Designing and building the structure of a neural network plays an important role in reaching a better performance and improving the training process. In this project, we kept the structures of the encoder and decoder unchanged. Besides simply adding fully connected dense layers, we could add convolutional layers to extract more important features and feed activation signals to dense layers for the encoder. Once the samples are compressed into latent space. The decoder inverses the process by feeding the compressed representations to dense layers first then signals go through deconvolutional layers. The reconstructed outputs probably capture more features if we implement this model architecture. Below (Figure 14) is the comparison between our FashionMNIST results with Aleco's FashionMNIST results [6]. Aleco utilizes convolutional and deconvolutional layers in the encoder and decoder models which produces better-reconstructed results.

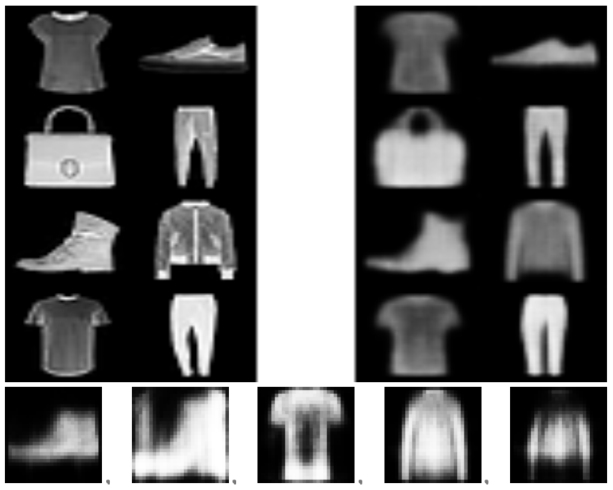


**Figure 14:** Alceo's results: original(top left) and reconstructed(top right), Our results: reconstructed(bottom five)

**latent space dimension**: The dimension of latent space should be treated as a hyperparameter of the VAE model. Normally, we could tune it by running a grid search on the number of dimensions, for example, training multiple models which encode samples into 3, 4, 5 dimensional latent spaces and make comparison with all the generated results. Select the best dimension value based on the reconstruction quality. Due to limited time and restricted computing resources for model training, we only experimented with the 3D latent space and the results are shown in figure 2. Intuitively speaking, if the dataset is more complex, for instance, the FashionMNIST dataset. Then we should choose a higher dimension value to capture more information. So that the reconstructed results will indicate more interesting features based on the encoded representation.

**Choice of likelihood model:** Depending on what type of data format, for instance, binary or floating-point formats, choosing an appropriate likelihood function to model the data distribution is crucial to final model performance. From our experiment, we choose the beta likelihood function on the floating-point format. It seems the reconstructed digits are blurry compared with baseline results in figure 11. And those random samples drawn from the distribution are even hard to recognize. Thus, the choice of likelihood function and type of data format should be carefully considered.

## 10 Conclusion

In this project, we experimented the variational autoencoder with various settings and compared every new reconstructed result with those produced by the baseline model. First, we found out that increasing the dimension of latent space from 2D to 3D would generate more realistic hand-written digits. Second, models conditioned on true labels tend to be easy to recognize in shape and writing style. Third, we also experimented the baseline model on other datasets like the FashionMNIST. The reconstructed results suggest that the baseline model is less capable of generating more complex data samples. Then, we utilize other divergence measurement metrics like the Jensen-Shannon divergence instead of the KL divergence for estimating the difference between the true distribution and the approximated variational distribution of the pixel values. And found out both metrics produce similar results. After this, we tried to implement $k$-sample importance weighted autoencoder that maximizes a different lower bound. And discovered that the performance of IWAEs improved with increasing $k$ value. We also replaced the Bernoulli likelihood function with the Beta likelihood function on floating-point format data, found that the reconstructed results using the Beta likelihood function are blurry. Lastly, we use the trained decoder of the baseline model and optimize $\phi_{\mu}$ and $\phi_{log\sigma}$ using ELBO to make inference of the bottom part of the digit given the top part of the digit. The result is reasonable and acceptable.

## Appendices

**Project code link:** `https://github.com/harrychenfeng/CSC412-2506_Project`

## References

[1] Dykeman, I. (2016, December 21). Conditional variational autoencoders. Retrieved April 23, 2021, from `https://ijdykeman.github.io/ml/2016/12/21/cvae.html#:~:text=A%20variational%20autoencoder%20(VAE)%20is,synthetic%20images%20of%20handwritten%20digits.`

[2] Reis, T. (2020, September 20). Measuring the Statistical similarity between two samples using Jensen-Shannon, Kullback-Leibler ... Retrieved April 23, 2021,from `https://medium.com/datalab-log/measuring-the-statistical-similarity-between-two-samples-using-jensen-shannon-and-kullback-leibler-8d05af514b15`

[3] Yuri Burda, Roger Grosse Ruslan Salakhutdinov, "Importance Weighted Autoencoders", 2015.

[4] Beta distribution. (2021, April 16). Retrieved April 23, 2021, from `https://en.wikipedia.org/wiki/Beta_distribution`

[5] Duvenaud, D., amp; Bettencourt, J. (2020, May 27). Assignment 3: Variational Autoencoders. Retrieved April 23, 2021, from `https://probmlcourse.github.io/csc412/assignments/assignment_3/A3.pdf`

[6] Alecokas. (n.d.). Alecokas/flux-vae. Retrieved April 23, 2021, from `https://github.com/alecokas/flux-vae`