VAE

Yan Han (519030910404)

June 10, 2022

1 Introduction

Variational AutoEncoders (VAE) is a method for generative modeling, whose goal is to build the bridge between normal distribution and data distribution. Section 2 formularizes the principle of VAE. Section 3 describes some experiments on MNIST. Section 4 introduces our interesting observations. Section 5 draws a conclusion. The source code is available at `https://github.com/wolfball/Intellisense-Cognitive-Practice`.

2 Principle

The problem of "generative modeling" is to deal with models of distributions $P(X)$, where X is the datapoints in some potentially high-dimensional space $\mathcal{X}$. Take handwritten digits for example, the datapoint is the image which has a large number of pixels, and the generative model is aimed at capturing dependences between pixels and judging whether $P(X)$ is high or low. the X's which look like real images should get high probability while that look like random noise should get low probability. Suppose that X is sampled from $P_{gt}(X)$, our goal is to find P such that $P = P_{gt}$.

In the case of handwritten characters, it helps if the model first decide which character to generate before it assigns a value to any specific pixel. Thus we introduce the *latent variable z* sampled from a high-dimensional space Z with $P(Z)$. That is to say, we have a function $f : \mathcal{Z} \times \Theta \to \mathcal{X}$ to generate the images $f(z; \theta)$. With the popularity of neural networks, we use them to model this function with parameters θ. Now our goal is to optimize θ to make $f(z; \theta)$ become more similar to the data in $\mathcal{X}$.

To make the notion precise mathematically, we are aimed at maximizing the probability of each X in the training set:

$$P(X) = \int P(X|z; \theta)P(z)dz \tag{1}$$

where we replace $f(z; \theta)$ by $P(X|z; \theta)$ to build relation between X and z.

For one thing, we need to define z. In [1], we know that any d-dimention distribution can be generated by taking a set of d variables that are normally distributed and mapping them through a sufficiently complicated
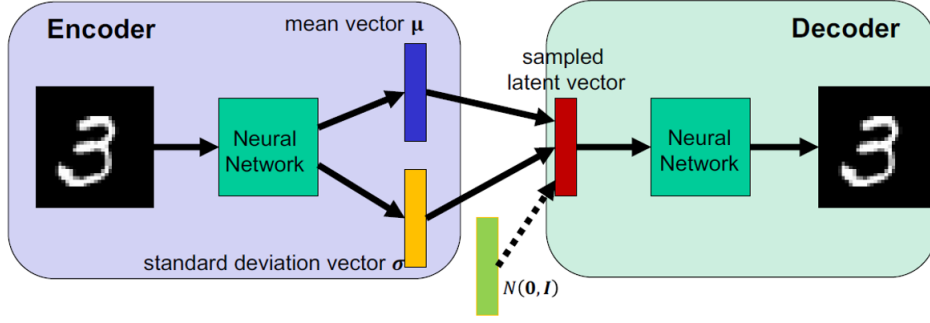
Figure 1: Pipeline

funtion. We can write

$$P(X|z;\theta) = \mathcal{N}(X|f(z;\theta), \sigma^2 I) \tag{2}$$

For another, we need to find out how to integrate over z. Since it is expensive to consider the whole space of Z, we introduce an new function $Q(z|X)$ to narrow the "interesting space". Besides, we will use KL divergence $D[Q||P]$. Then, for arbitrary distribution $Q(z)$,

$$D[Q(z)||P(z|X)] = \mathbb{E}_{z\sim Q}[\log Q(z) - \log P(z|X)] \tag{3}$$

$$= \mathbb{E}_{z\sim Q}[\log Q(z) - \log P(X|z) - \log P(z)] + \log P(X) \tag{4}$$

$$\log P(X) - D[Q(z)||P(z|X)] = \mathbb{E}_{z\sim Q}[\log P(X|z)] - D[Q(z)||P(z)] \tag{5}$$

As we are interested in $P(X)$, it is better to employ those Q that depend on X, thus:

$$\log P(X) - D[Q(z|X)||P(z|X)] = \mathbb{E}_{z\sim Q}[\log P(X|z)] - D[Q(z|X)||P(z)] \tag{6}$$

where $Q(z|X) = \mathcal{N}(z|\mu(X,\theta), \Sigma(X;\theta))$.

The full training equation is:

$$\mathbb{E}_{X\sim D}[\log P(X) - D(Q(z|X)||P(z|X))] \tag{7}$$

$$= \mathbb{E}_{X\sim D}[\mathbb{E}_{z\sim Q}[\log P(X|z)] - D(Q(z|X)||P(z|X))] \tag{8}$$

$$= \mathbb{E}_{X\sim D}[\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[\log P(X|z = \mu(X) + \Sigma^{1/2}(X)\epsilon)] - D(Q(z|X)||P(z|X))] \tag{9}$$

the equation $(8) \to (9)$ uses a trick called *reparameterization*, which makes sure that given X, ϵ, the function is deterministic and continuous over P, Q.

3 Experiment

Experiment setting   The pipeline of VAE is in Figure 1. The dataset is MNIST. In our baseline model, the encoder is implemented by an one-layer fully connected layer with ReLU activation, which can map 784d image to 400d hidden vector. And we use two Linear layers to produce μ and Σ respectively, whose dimensions are also hyper parameters called *lat_dim*. The decoder is a two-layer-MLP followed by a Sigmoid activation that can recover a 784d image from a latent point. For optimizer, we use Adam with learning rate as 1e-5. The loss consists of reconstruction loss which is BCELoss, and regularization loss which is the KL-distance between $\mathcal{N}(\mu,\sigma)$ and $\mathcal{N}(0,I)$. We train all the models for 200 epochs. And we use Tensorboard for visualizations.

Visualization   Here we mainly introduce four types of visualizations.

1. Loss curve. Take Figure 5 for example, we will plot the test loss and train KL-loss during training where the x-axis denotes epoch iteration and the y-axis represents the loss value;

2. Reconstructive visualization. Take Figure 2(b) for example, in order to visualize the quality of generated images, we feed the images in first four lines into the network, and the last four lines are the outputs;

3. Sampled visualization. Take Figure 3(a) for example, we evenly sample z from the latent space within a bounded region ([-5, 5] for each dimension), and feed z into decoder. The outputs form this graph. Note that only the latent dimension less than three can be visualized.

4. Latent space visualization. Take Figure 3(b) for example, we input all data into the encoder, which produce responding z. And we visualize these z to form these graphs. Note that only the latent dimension less four can be visualized.

Latent dimension   As the dimension of latent variable represents the capability of model, we set different latent dimension to see the performance. In particular, we visualize the result when dimension is less than 3 to better our understanding. We use 3 layers in encoder. From Figure 2.(a), we find that the higher the latent dimension is, the less the test loss will be. And Figure 2.(b) is the reconstruction result on the test data set. When latent dimension is equal to 1, most images can not reconstructed; when latent dimension is equal to 3, we can clearly see that most of them are recognized. From Figure 3.(a), we evenly sample z and feed these z's into the network, the visualization shows that the spatial information of the latent variable is learned, which is more vivid in Figure 3.(b).

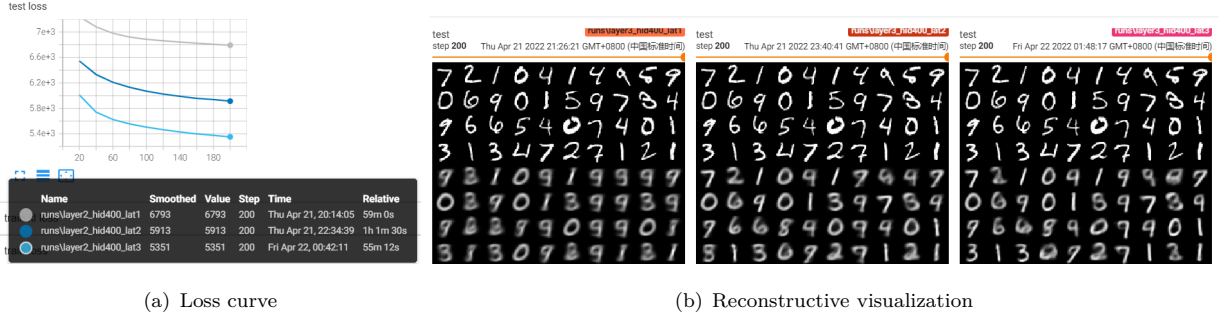(a) Loss curve   (b) Reconstructive visualization

Figure 2: Loss curve and reconstructive visualization with different latent dimensions. In (a), the curves represent latent dimension with 1, 2, and 3 from top to bottom. In (b) the images represent latent dimension with 1, 2 and 3 from left to right.

Number of encoder layers   As the number of encoder layers reflects the capability of extracting feature, we set different encoder layers to see the difference. We set latent dimension as 2. From Figure 4, we find that the deeper network in encoder can improve the performance of VAE. The reason may be that the better encoder can capture more robust features and dependences of the images.

Hidden size   Hidden size determines the intermediate vector in Encoder and Decoder, which shows the potential ability of modeling. With one layer encoder and 20d latent vector, we have tried 100, 400, 1000 and 2000 hidden size. Figure 5 shows the results. Lager hidden size results in lower test loss, but higher training KL-loss.

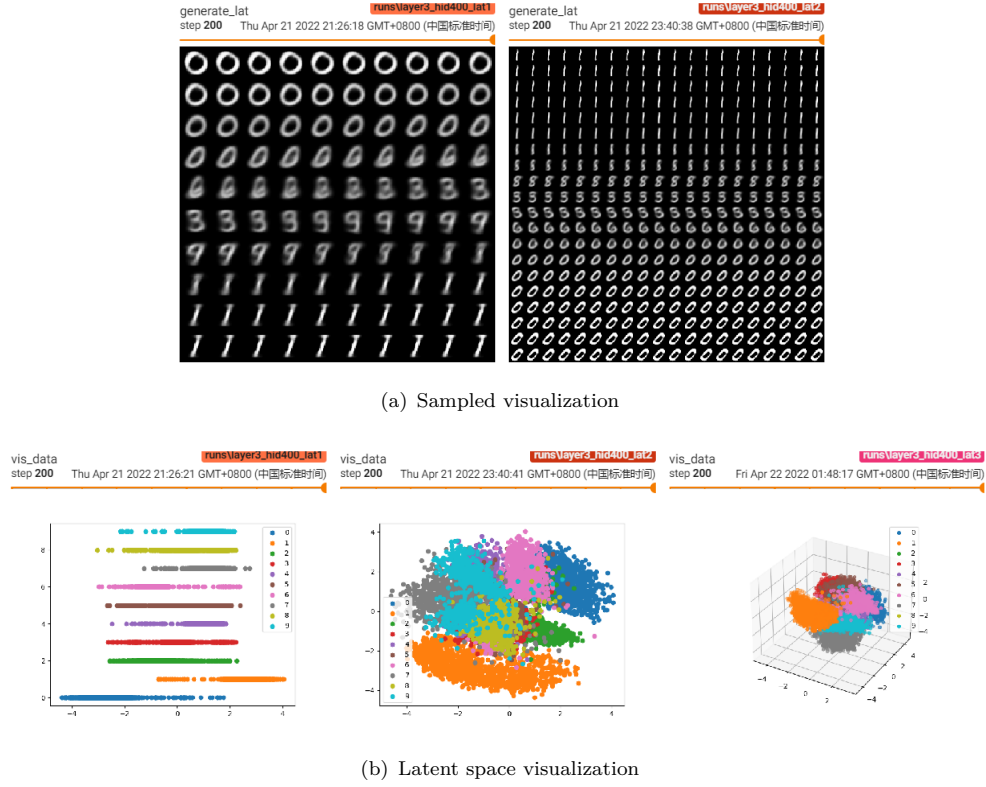(a) Sampled visualization

(b) Latent space visualization

Figure 3: Sampled and latent space visualization with different latent dimensions.

The drop in test loss can be regarded as the improvement of the capacity of the model, and the increase in KL-loss may be to blame for the complexity of data distribution (not normal distribution).

CNN While convolution neural network(CNN) can capture the local information in an image, we use CNN to replace MLP in the baseline. Both networks use one layer encoder and set latent dimension as 1. See Figure 6 for results. CNN performs better than MLP, but results in higher KL-loss. Besides, VAE with CNN shows more diversity(See line 7 in Figure 6(c)).
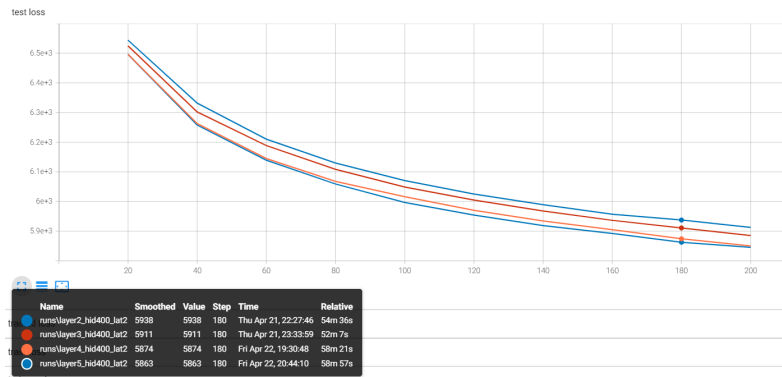

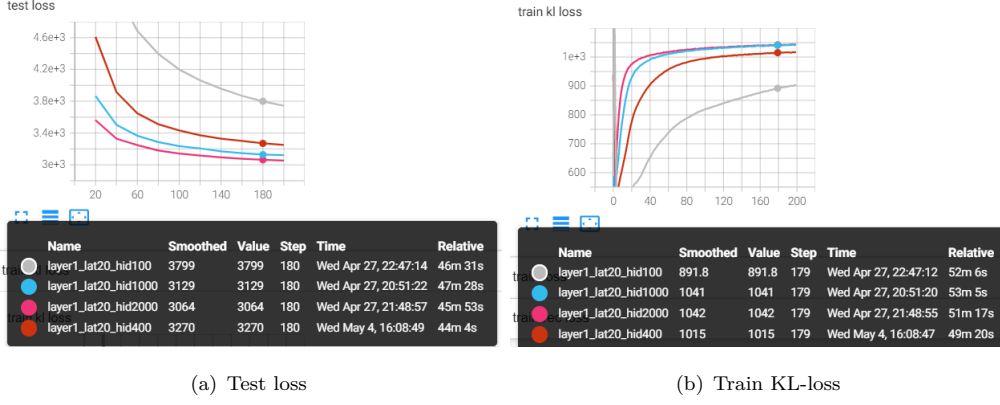Figure 4: Test loss curve with different layers.

(a) Test loss

(b) Train KL-loss

Figure 5: Test loss and train KL-loss of experiments with different hidden dimensions.

(a) Test loss

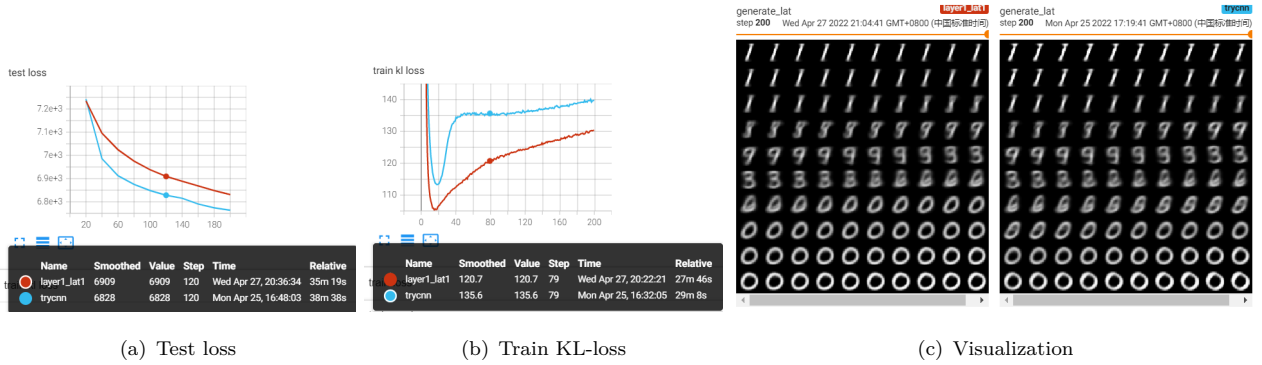

(b) Train KL-loss

(c) Visualization

Figure 6: Experiments of replacing MLP with CNN.

4 Interesting Observations and Thoughts

4.1 Trends in Curve

We find that when VAE works well in our experiments, the reconstruction loss always decreases while the KL-loss descends first and rises later (See Figure 8). The possible causes may be that at first, the KL-divergence helps the sampled points representing different numbers to cluster, and later, since the data distributions are actually not standard gaussian distribution, the improvement in reconstruction will result in the drop in similarities between data distribution and gaussian distribution. Finally, these two losses will convergence to comparative orders of magnitude (Here is 3000 and 1000).

4.2 β-VAE

Inspired by [2], while the total loss of VAE consists of two parts: reconstruction loss and regularization loss (KL loss), we will naturally consider adding one parameter β to balance these two items. That is to say, redesign the loss function:

$$\mathcal{L} = \mathcal{L}_{BCE} + \beta \cdot \mathcal{L}_{KL} \tag{10}$$

For $\beta \in \{0.1, 0.5, 1.0, 2.0, 4.0\}$, we do some experiments with 3-layer-encoder and 2d latent vector. Higher β will push the network to focus more on KL-loss, thus resulting in less KL-loss but higher reconstruction loss. As reconstruction loss is larger than Kl loss, the total loss increases with increasing β (See Figure 7). And the
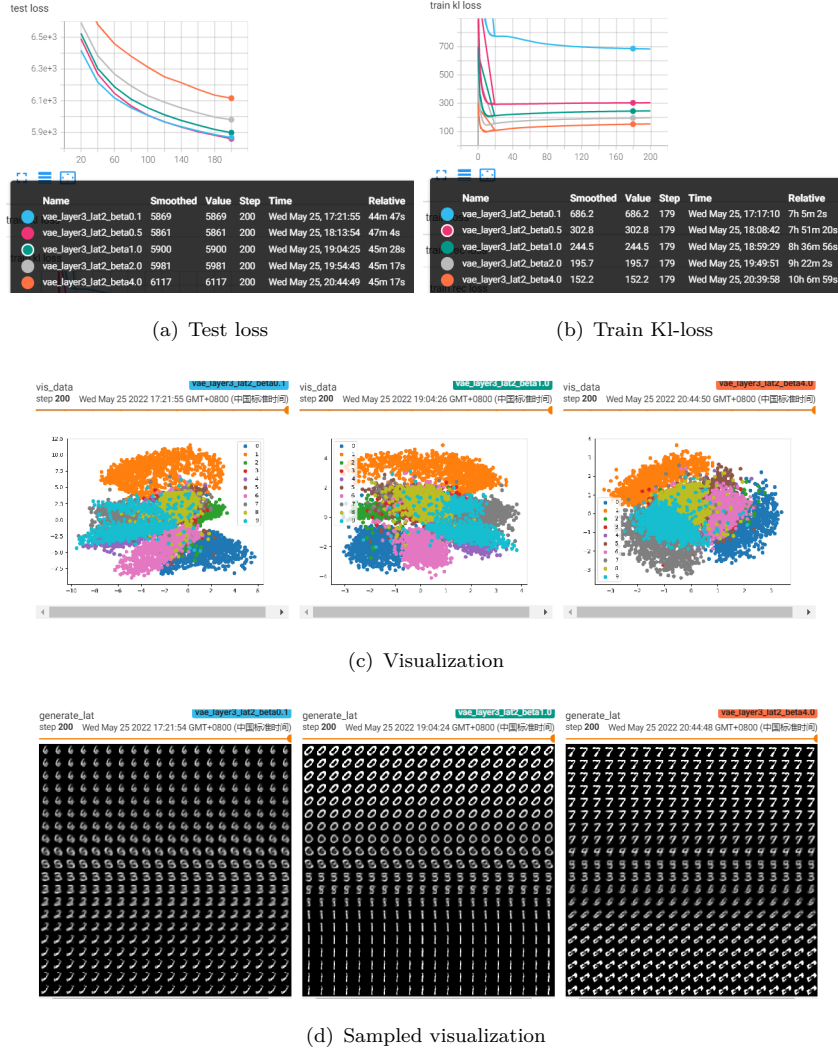
(a) Test loss

(b) Train Kl-loss

(c) Visualization

(d) Sampled visualization

Figure 7: Experiments on different β.

distribution is more like normal distribution under $\beta = 4.0$.

Besides, we find that different β will help different numbers to be "outstanding". For instance, we can hardly see "2" (green points) when $\beta = 4.0$ while it is vivid in low values of β. And "5" (brown points) is on the opposite. Plus, the numbers on the boundary are also different.

5 Conclusion

In summary, we introduce the principle of VAE and realize it using Encoder-Decoder architecture. The visualization when latent dimension is one and two is available in this report. Besides, we have designed various experiments to research on the impact of the parameters. Moreover, we find it interesting to use one parameter β to balance between the reconstruction loss and the regularization loss. The lowest test loss is reached under three-layer-encoder, 100d-latent-vector and 2000d-hidden-size, which is 3002. And the quality of reconstructed images is shown in Figure 9.

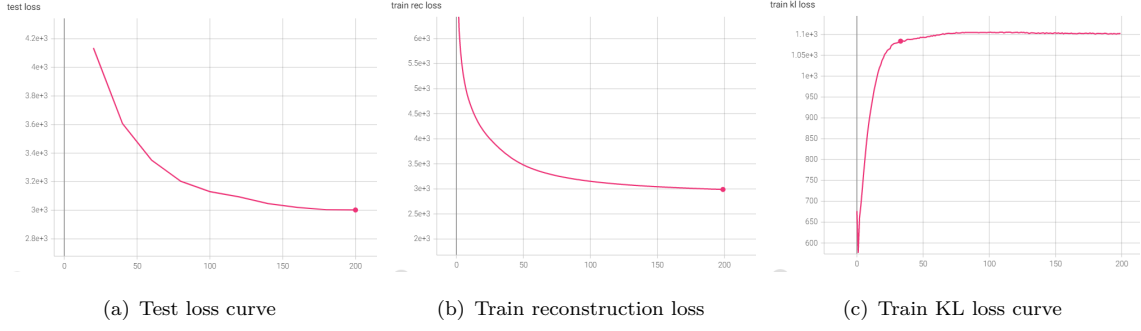| (a) Test loss curve | (b) Train reconstruction loss | (c) Train KL loss curve |

Figure 8: The loss curve of the best model, who has three-layer-MLP-encoder, 100d-latent-vector and 2000d-hidden-size.
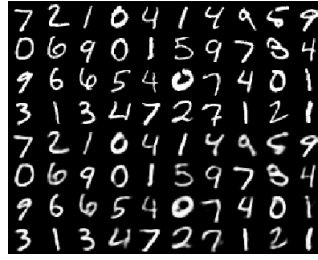

Figure 9: The reconstructive visualization of the best model, who has three-layer-MLP-encoder, 100d-latent-vector and 2000d-hidden-size.

References

[1] Carl Doersch. Tutorial on variational autoencoders, 2016.

[2] Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.