Variational Autoencoder

Yinjie Wang

May 20, 2023

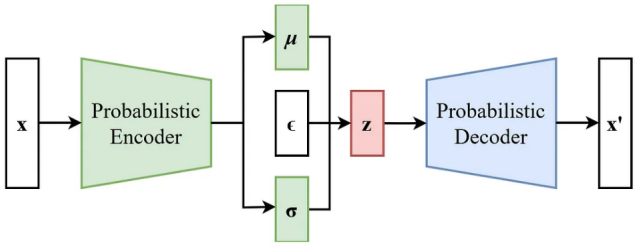
1 VAE

Variational Autoencoder (VAE) is a deep generative model with latent variables. We assume that the observed random variable X exists in a high-dimensional space and can be generated by a latent variable z in a low-dimensional space. For any given X, the first part of VAE, the encoder, transforms it into its latent space version z. Then, the decoder, on the other hand, recovers X from the input of z. However, the key aspect of VAE is its ability to map distributions as well. In fact, after the training, we aim for the output of the decoder to have the same distribution as X by randomly sampling the input of the decoder, z. We often set z to be a normal distribution, which is convenient for sampling after training. Therefore, VAE is often used for generative data augmentation, representation learning, and dimensionality reduction.

The density function of this model can be represented by:

$$p(X, z; \theta) = p(X \mid z; \theta)p(z; \theta).$$

Our goal is to maximize $\log p(x; \theta)$. In the variational inference method, we need to estimate posterior probability $p(z \mid X; \phi)$ in the E step. However, it is challenging to estimate it precisely when this distribution is complicated. Thus, we use neural network to assist us in this task.



Structure of VAE

The first part of this model, the encoder, helps us obtain the posterior probability $q(z \mid X; \phi)$, which serves as an estimator for $p(z \mid X; \phi)$. However, it's important to note that we want the distribution to be estimated accurately and z to follow a simple distribution for easy sampling after training. Therefore, we set z to follow a normal distribution $p(z)$ and configure the encoder to provide the mean $(\mu(X, \phi))$ and variance $(\sigma^2(X, \phi))$ for $z \mid X$. With them, $q(z \mid X; \phi)$ is determined since it was set to be normal distribution $N(\mu(X, \phi), \sigma^2(X, \phi))$. We

sample z based on this distribution to serve as the input for the decoder, which in turn maps it to X'. This reconstructed X' should ideally align with the original input X.

It's worth noting that the normal distribution of z doesn't compromise the flexibility of this generative model, as a complex mapping of z can still result in diverse distributions.

Now it's time to design the loss function. Note that we have:

$$log(p(X;\theta)) - KL\left[q(z \mid X;\phi)||p(z \mid X;\phi)\right] = ELBO, \tag{1}$$

$$where \quad ELBO = E_{z\sim q(z|X;\phi)}\left[\frac{p(z,X;\theta)}{q(z \mid X;\phi)}\right].$$

Then $ELBO$(evidence lower bound) is what we aim to maximize because $\log(p(X;\theta)) \geq ELBO$. Looking from a different perspective, $\log(p(X;\theta))$ is the objective we seek to maximize, while $KL\left[q(z \mid X;\phi)||p(z \mid X;\phi)\right]$ is the term we aim to minimize to enhance the estimation of $p(z \mid X;\phi)$. $ELBO$ can be written in a different form, which is our final objective function:

$$ELBO = E_{z\sim q(z|X;\phi)}\left[logp(X \mid z;\theta)\right] - KL\left[q(z \mid X;\phi)||p(z)\right] \tag{2}$$

In the training process, caculating $KL\left[q(z \mid X;\phi)||p(z)\right]$ is straightforward since $z|X\sim N(\mu(X,\phi),\sigma^2(X,\phi))$ and $z \sim N(0,I)$. In fact, the KL loss is $\frac{1}{2}(tr(\sigma^2 I) + \mu^T\mu - d - log(|\sigma^2 I|))$, where d is the dimension of latent space. But estimating $E_{z\sim q(z|X;\phi)}\left[logp(X \mid z;\theta)\right]$ needs sampling z, which lacks a gradient for caculation. So we use the reparameterization trick here. Specifically, we sample $\epsilon \sim N(0,I)$ for each X input, and set $z = \mu + \sigma * \epsilon$. Then the gradient can be derived from μ and σ. We set $X \mid z \sim N(decoder(z), \lambda I)$, where λ is a hyper-parameter. Finally, our loss function is

$$\frac{1}{2}||X - decoder(z)||^2 + \frac{\lambda}{2}(tr(\sigma^2 I) + \mu^T\mu - d - log(|\sigma^2 I|)),$$

where $\mu = \mu(X,\phi)$ and $\sigma = \sigma(X,\phi)$.

Now we apply VAE to the MNIST dataset. The encoder and decoder both consist of convolutional layers and linear fully connected layers.

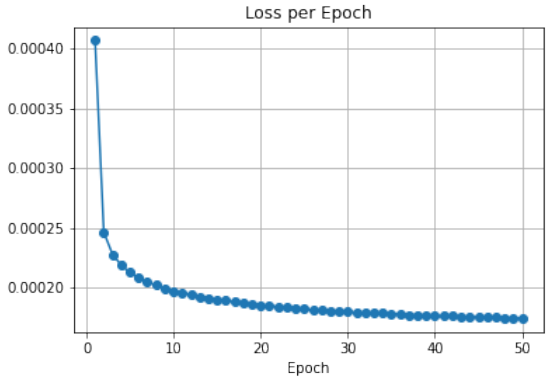


Figure 1: Training process

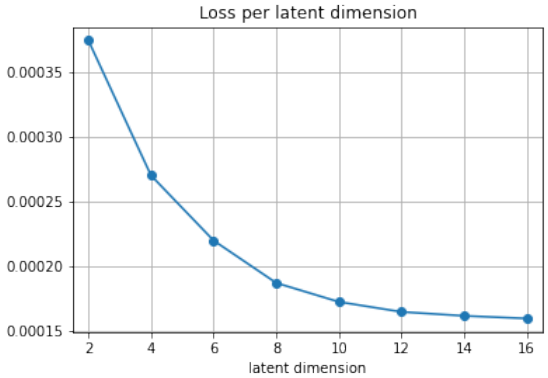


Figure 2: Influence of latent dimension

From Figure 2 above, we found that the training convergence achieved and the larger dimension of latent space will have a better performance. Then we choose to set 10 as the dimension of latent space and sampled some pictures from VAE(Figure 3). The generated pictures are somewhat vague but still recognizable as numbers.

Figure 3: Generated samples of VAE

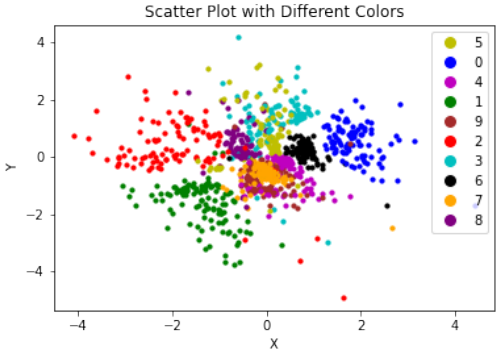


Figure 4: Latent space

Then we explore whether VAE can be used to clustering by looking into the latent space. We set 2 as the dimension and trained the model. Figure 4 indicates that the latent space does separate these numbers, but it is not good enough. It's understandable sincere VAE is designed to generate rather than classification.

We will benchmark the performance of reconstruction, running times, and the number of parameters for different VAE models at the end of this report.

2 BVAE

Beta-Variational Autoencoder(BVAE) is an improved version of VAE, which can help us achieve decoupled representation, that is, each element in the embedding corresponds to a separate influential factor.

The objective of BVAE is

$$max_{\theta,\phi} E_{z\sim q(z|X;\phi)}\left[logp(X \mid z;\theta)\right], \quad st.\, KL\left[q(z \mid X;\phi)||p(z)\right] \leq \epsilon.$$

Condition $KL\left[q(z \mid X;\phi)||p(z)\right] \leq \epsilon$ requires the posterior probability $q(z \mid X;\phi)$ to align with $p(z)$. Once converted into Lagrangian form, the objective function becomes

$$E_{z\sim q(z|X;\phi)}\left[logp(X \mid z;\theta)\right] - \beta\left(KL\left[q(z \mid X;\phi)||p(z)\right] - \epsilon\right).$$

When $\beta = 1$, it's just VAE. When β becomes larger, $q(z \mid X;\phi)$ will become simpler, and z will require less information to transmit from X. The original paper states that: If the algorithm can reconstruct images effectively while transmitting a small amount of information, then it has certainly learned a good decoupled representation.

Figure 5: VAE

Figure 6: BVAE

To compare the performance of BVAE and VAE, we set $d = 10$. For each element of $z \in R^d$, z_i, where $1 \leq i \leq d$, we shift the values of z_i and generate the corresponding images (like each row of Figure 5). We observe that the shifts in VAE are not interpretable for humans, whereas the shifts in BVAE are much clearer. The images generated by BVAE tend to transition smoothly from one number to another as each element of z is shifted."

From Figure 7, the generated pictures of BVAE are just like which of VAE ($d = 10$). And the clustering in latent space performed in the same way ($d = 2$).

Figure 7: Generated samples of BVAE

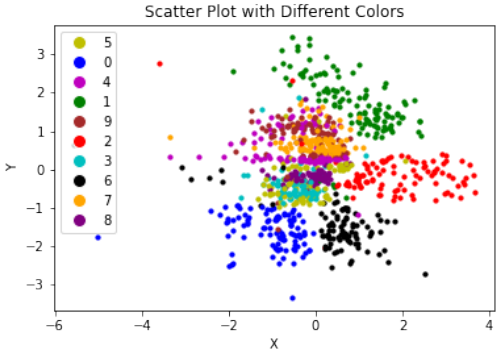


Figure 8: Latent space of BVAE

3 CVAE

The conditional-Variational Autoencoder(CVAE) is kind of VAE that can generate based on the conditions(c) provided. For example, the label of number in MNIST dataset can be seen as a condition.

Notice that

$$logp(X \mid c) \tag{3}$$

$$= E_{z\sim q(z|X,c)}\left[\frac{p(X,z \mid c)}{q(z \mid X,c)}\right] + KL(q(z \mid X,c)||p(z \mid X,c)) \tag{4}$$

$$\geq E_{z\sim q(z|X,c)}\left[\frac{p(X,z \mid c)}{q(z \mid X,c)}\right] = ELBO,$$

where the $ELBO$ of CVAE is

$$E_{z\sim q(z|X,c;\phi)}\left[logp(X \mid z,c;\theta)\right] - KL\left[q(z \mid X,c;\phi)||p(z \mid c)\right].$$

We often assume that the latent variables are independent with the conditions. Therefore $p(z \mid c)$ is just $p(z)$ here. What we need to change in the neural network is simply to include the conditions as part of the input for both the encoder and decoder.

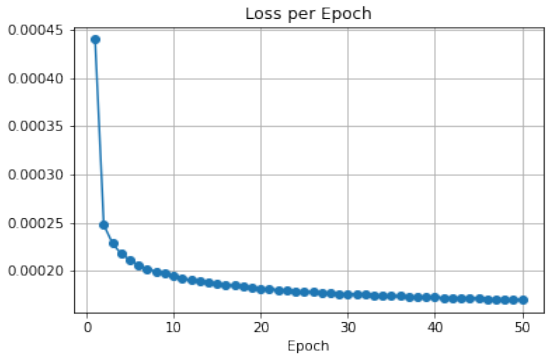


Figure 9: Training process

Figure 10: Generated samples by condition 8

From Figure 10, we observe that most of the generated pictures depict the number 8. The provided condition did contribute to the generation process. However, there are still some pictures of 9 and 7, indicating that the influence of the given condition is not sufficiently strong. With the similar number of parameters and convolutional layers, the classical CNN+softmax would perform better in the labeling task.

Figure 11: Generated samples of CVAE

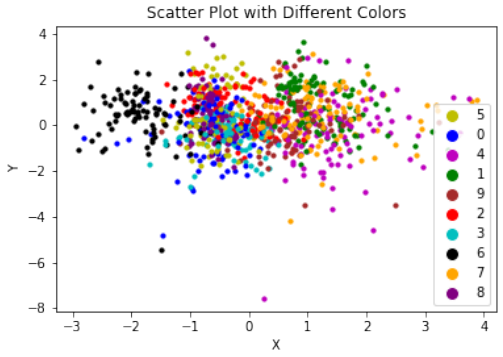


Figure 12: Latent space of CVAE

Here we also evaluate CVAE performance of randomly sampled pictures(Figure 11). Notice that in Figure 12, we can not see any clustering this time, this it because the latent space z and condition c are independent (We set $p(z \mid c)$ to be $p(z)$).

4 VQVAE

The authors of Vector Quantized Variational Autoencoder(VQVAE) believe that the reason for the low quality of generated pictures in VAE is because the images are encoded into continuous vectors. In reality, encoding pictures into discrete vectors would be more natural.

VQVAE is not a VAE, but rather a AE. It compresses pictures into discrete vectors. For example, after passing through the convolutional layers of the encoder, our input picture $(z_e(X))$ becomes a tensor with size [C, H, W], where C represents the number of channels, H represents the height, and W represents the width. We replace each R^C vector (a total of H*W vectors) by selecting the nearest R^C vectors from a finite discrete vector set (embedding space). After this replacement, the tensor $(z_q(X))$ will serve as the input of decoder. So the pictures that VQVAE can generate is finite, which is K^{H*W} totally, where K is the size of embedding space.

From the perspective of a VAE, VQ-VAE does not have a KL loss. Let's assume that the discrete encoding of the input X is represented by $decoder(X)$. Then $q(z \mid X; \phi)$ is nonzero only when $z = decoder(X)$(As it is discrete). Thus, the KL divergence is a constant and is not included in the loss function here.

However, $||X - decoder(z_q(X))||^2$ can not be chosen as loss function to optimize. Since the step from $z_e(X)$ to $z_q(X)$ is non-differentiable. VQVAE utilizes a technique called the 'straight-through estimator' to accomplish gradient passing. Therefore, we design the reconstruction loss function:

$$||X - decoder(z_e(X) + sg(z_q(X) - z_e(X)))||^2,$$

where $sg(x)$ takes x in forward propogation, and takes 0 in backward propogation.

To optimize the embedding space, we add two terms and obtain the loss function of VQVAE:

$$L = ||X - decoder(z_e(X) + sg(z_q(X) - z_e(X)))||^2 + \alpha||sg(z_e(X)) - z_q(X)||^2 + \beta||sg(z_q(X)) - z_e(X)||^2$$

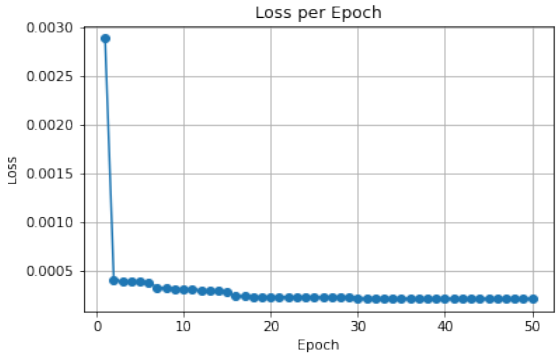


Figure 13: Training process

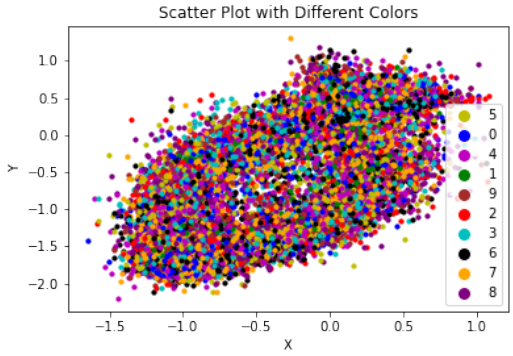


Figure 14: Generated samples by condition 8

The latent space $(z_e(X))$ cannot be used for clustering because each "pixel" of it is assigned to one of K different groups.(Figure 14).

Note that VQVAE alone is not a picture generative model, since we can not sample from the discret embedding space. However, PixelCNN can generate discrete codes. Then, we can use VQVAE to transform the codes into images, helping us generate the pictures. The specific implementation will be updated later.

5 Some other benchmarks for models

Here we evaluate the reconstruction performance, running time and number of parameters across differnt models.

(a) Reconstructions of different models

Model	Time	N
VAE	9:22	681367
BAE	9:50	681367
CAE	10:00	682411
VQVAE	7:45	307225

Table 1: Time and number of parameters

Figure 16: Plot and Table: N stands for number of parameters.

From Figure 16, it is evident that VQVAE achieves the best performance(focus on number 2,3,4,5, VQVAE achieves the clearest reconstructions), even with the smallest amount of time and parameters, across all 50 epochs.