Variational Autoencoder

Yinjie Wang

March 30, 2023

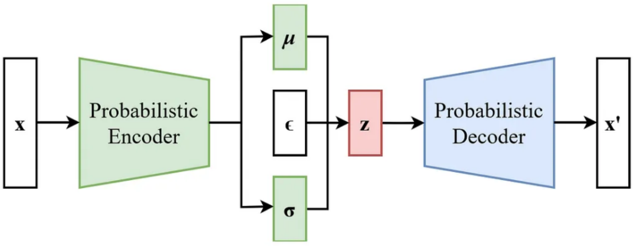
1 VAE

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(R^m) and can be generated by a latent variable z in a low-dimensional space(R^d). 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:

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

Our goal is to maximize $E[\log p(X)]$. 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 $(\mu_E = \mu_E(X, \phi))$ and variance $(\sigma_E^2 = \sigma_E^2(X, \phi))$ for $z \mid X; \phi$. With them, $q(z \mid X; \phi)$ is determined since it was set to be normal distribution $N(\mu_E, \sigma_E^2)$. We sample z based on this distribution to serve as the input for the decoder, which in turn maps it to $X' = \mu_D$. 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.

Evidence lower bound(ELBO) of $\log p(X)$ serves as our objective function here:

$$ELBO = E_{z\sim q(z|X;\phi)} [\log p(X \mid z; \theta)] - KL [q(z \mid X; \phi)||p(z)] ,$$

where $-E_{z\sim q(z|X;\phi)} [\log p(X \mid z; \theta)]$ is defined as reconstruction loss, and $KL [q(z \mid X; \phi)||p(z)]$ is defined as KL loss. The derivations are in the Appendix. We just focus on maximizing ELBO here.

In the training process, caculating $KL [q(z \mid X; \phi)||p(z)]$ is straightforward since $z|X \sim ; \phi$ $N(\mu_E,\sigma_E^2)$ and $z \sim N(0, I)$. But estimating $E_{z\sim q(z|X;\phi)} [\log p(X \mid z; \theta)]$ needs sampling z, which lacks a gradient for caculation. We use the reparameterization trick here to deal with this trouble. Specifically, we sample $\epsilon \sim N(0, I)$ for each X input, and make $z = \mu + \sigma * \epsilon$. Then the gradient can be derived from μ and σ.

2 Reimplementation of paper

We reimplemented all the results of the original VAE paper, Auto-Encoding Variational Bayes(https://arxiv.org/abs/1312.6114). Two datasets are used here, the MNIST dataset and the Frey Face dataset. The original paper used a Bernoulli MLP as decoder of MNIST, and used a Gaussian MLP as decoder of Frey Face (Details can be found in Appendix).

2.1 Training Processes with different latent dimensions

This paper only used one hidden linear layer to compose the encoder and decoder. In Figure 1 and 2, they employed 500 units in the latent layer. The optimizer chosen is Adagrad. Minibatches of size is 100. Active function is Tanh().

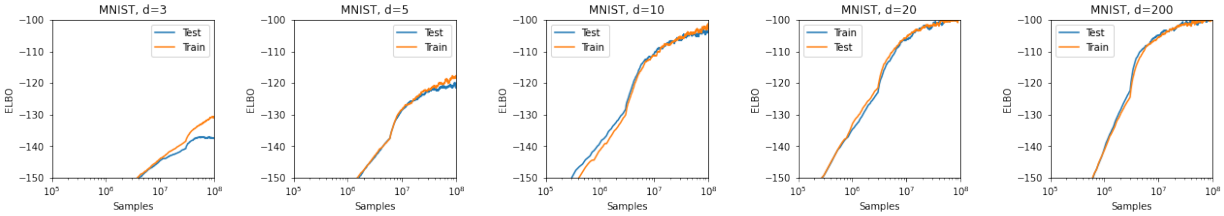


Figure 1: Training Process of MNIST data

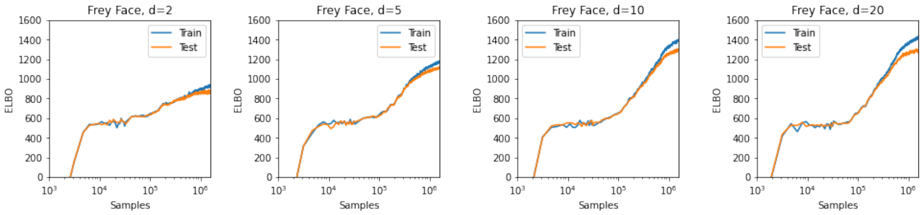
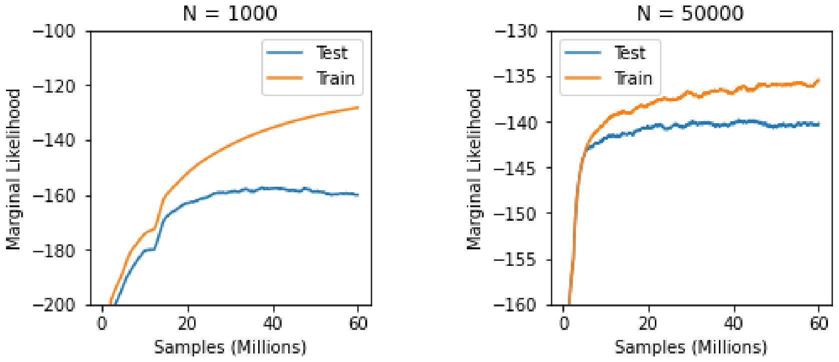


Figure 2: Training Process of Frey Face data

It is straightforward that the ELBO loss decreases as the dimensionality increases. Notably, employing more latent variables does not necessarily lead to overfitting in MNIST data. The author of the VAE suggests that this phenomenon is attributed to the regularizing effect of the ELBO.

2.2 Marginal Likelihood

Our previous objective was $E[\log p(X)]$, representing the marginal likelihood. We followed them to utilize the marginal likelihood to evaluate the model with 100 hidden units and 3 latent variables ($d = 3$). The estimation for the marginal likelihood is provided in the Appendix.



Small sample

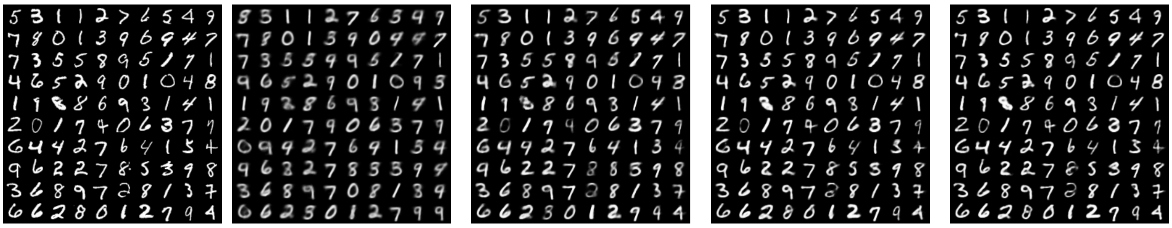
Large sample

Figure 3: Marginal Likelihood

Figure 3 illustrates the marginal likelihood curves when training on small and large proportions of the original training set. When trained with a large number of samples, the VAE effectively enhances the likelihood by optimizing the ELBO. However, with fewer training samples, it tends to exhibit significant overfitting. Notably, our reimplemented testing curve appears to decrease towards the end in the left figure, possibly due to the limited number of training samples utilized.

2.3 Reconstruction

For reimplementation, we used the trained model(only one latent layer with 500 units) to reconstruct some pictures.



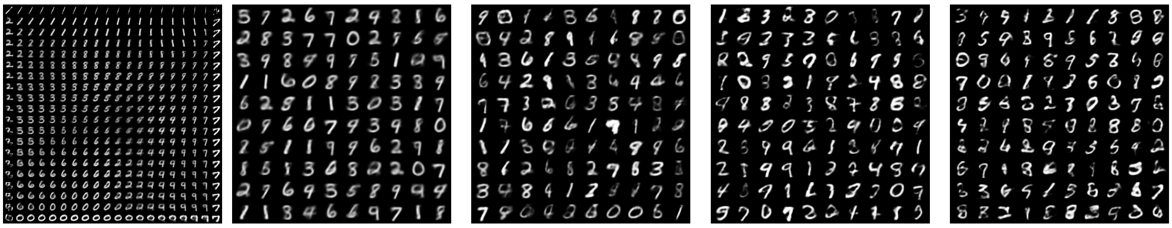
Original d=2 d=5 d=10 d=20

Figure 4: Reconstruction with different d

Figure 4 demonstrates that a larger latent dimension (d) can result in better reconstruction performance.

2.4 Generation

Figure 5 displays the manifold and generated pictures. Interestingly, with a larger latent dimension (d), the generated pictures are more blurry, consistent with the findings of the original VAE paper. While this paper does not provide an explanation, there are two possible reasons. Firstly, the relatively higher dimension of z makes it challenging to align $q(z \mid X; \phi)$ with $p(z)$. Another possible reason could be the antagonism between the KL loss and reconstruction loss. In other words, this VAE model may struggle to simultaneously optimize these two losses (https://stats.stackexchange.com/questions/341954/balancing-reconstruction-vs-kl-loss-variational-autoencoder). If the model focuses on optimizing the reconstruction loss, the KL loss may remain large or even increase as training progresses, and vice versa. In section 3, we will conduct experiments to investigate the relationship between these two types of loss.



Manifold d=2 d=5 d=10 d=20

Figure 5: Generations with different d

3 Other VAE models

3.1 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

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

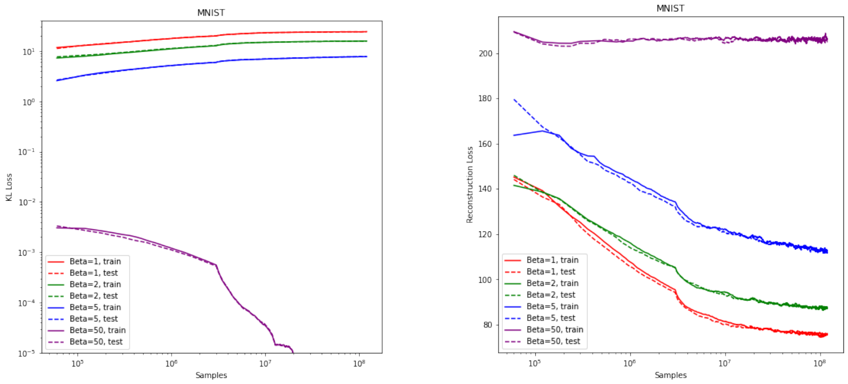
Condition $KL \left[q(z \mid X; \phi) || p(z) \right] \leq \epsilon$ requires the posterior probability $q(z \mid X; \phi)$ to align with $p(z)$. Once converted into Lagrangian form, the objective function becomes

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

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.

From another perspective, β is a hyperparameter that enables us to assign different weights to the two losses, KL loss and reconstruction loss. For instance, increasing β would prioritize optimizing the KL loss over the reconstruction loss during training. We will demonstrate this through the following experiments.

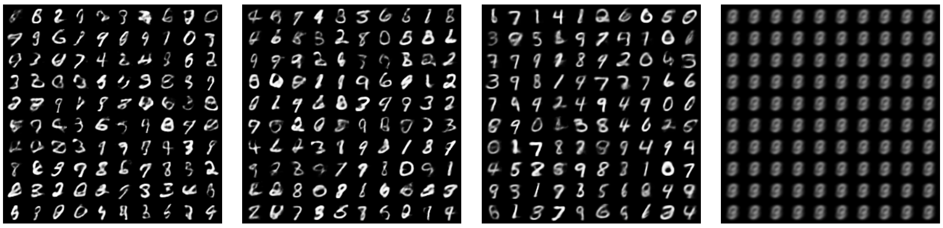
Specifically, we used one hidden layer with 500 units and set the latent dimension (d) to be 20. We chose β to be 1, 2, 5, and 50, and evaluated their performance based on the KL loss and reconstruction loss, respectively.



KL loss Reconstruction loss

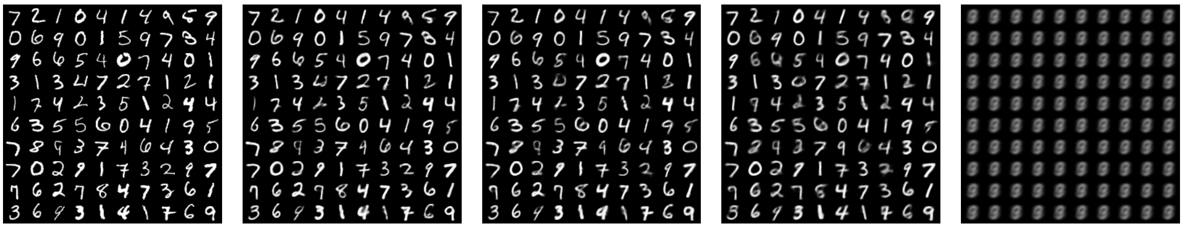
Figure 6: Training process of different β

Figure 6 illustrates the potential conflict between the KL loss and reconstruction loss. Notably, when $\beta = 1$, 2, and 5, the KL loss tends to increase(But their magnitudes are controlled). When β is extremely large, such as 50, both the KL loss and reconstruction loss are optimized, but the optimizing effect on the reconstruction loss is not as clear.



$\beta = 1$ $\beta = 2$ $\beta = 5$ $\beta = 50$

Figure 7: Generations with different β



Original $\beta = 1$ $\beta = 2$ $\beta = 5$ $\beta = 50$

Figure 8: Reconstructions with different β

From Figure 7 and 8, we observe that, in general, as β increases, the performance of generation improves while the performance of reconstruction deteriorates. However, when β becomes much larger ($\beta = 50$), both the performance of generation and reconstruction decline.

This is due to the loss of reconstruction power, which limits the generation ability, although the KL loss is small enough.

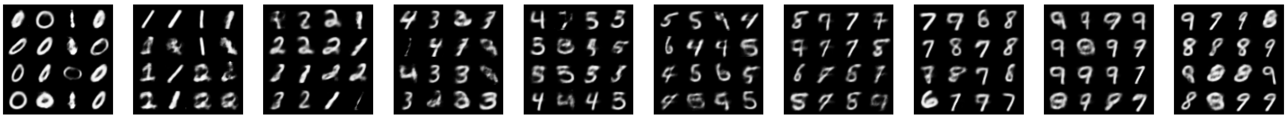
3.2 CVAE

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. The $ELBO$ of CVAE is

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

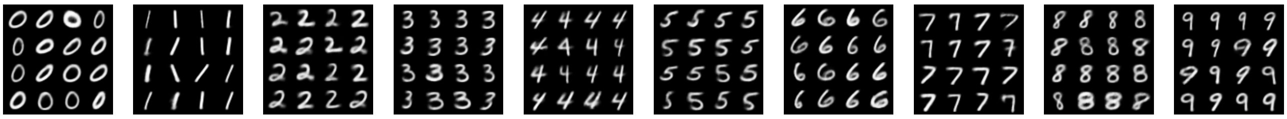
We often assume that the latent variables are independent with the conditions. Therefore $p(z \mid c)$ is just $p(z)$ here. What we need to change in the neural network is simply to include the conditions as part of the input for both the encoder and decoder.

For the implementation of MNIST data, the conditions (c) we used here are just the labels of the pictures. However, we found that the generated pictures are very blurry, and the generated pictures do not align with the given conditions (see Figure 9). Then we transformed the labels into one-hot vectors as inputs. The model with the same hidden layer structure performs much better (see Figure 10).



$c=0$ $c=1$ $c=2$ $c=3$ $c=4$ $c=5$ $c=6$ $c=7$ $c=8$ $c=9$

Figure 9: Conditional generation with raw input



$c=0$ $c=1$ $c=2$ $c=3$ $c=4$ $c=5$ $c=6$ $c=7$ $c=8$ $c=9$

Figure 10: Conditional generation with one-hot input

3.3 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 $encoder(X)$. Then $q(z \mid X;\phi)$ is nonzero only when $z = encoder(X)$(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:

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

where α and β are hyperparameters.

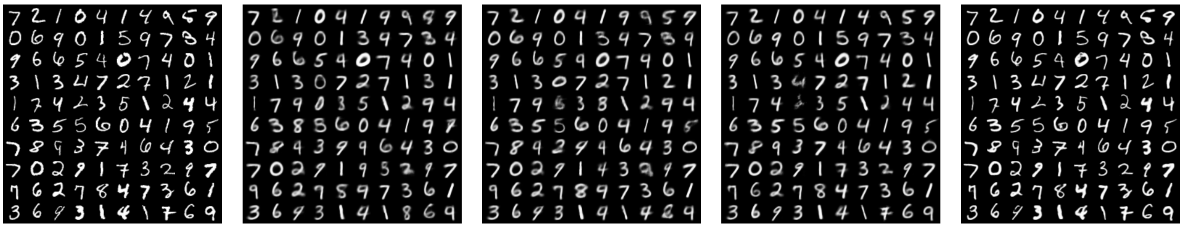
4 Benchmarks of different models

We need to choose the proper structure for the model to make the generated pictures look clear. Using Gaussian MLP as a decoder to estimate the variance will lead to an oversized loss function, given that the boundary of MNIST pictures is always 1 (black). Therefore, we use Bernoulli MLP as a decoder. We give up the convolutional layers for simplicity and only use MLP for the encoder and decoder. The following is the notation used to define the structure of VAE.

Definition 4.1. *We denote a VAE as* $\mathbf{h}$*=[*h_1*,* h_2*,. . . ,* h_n*], if its latent dimension is* h_n *and its encoder (decoder is just inverted) is composed of n-1 hidden layers with* h_1*,* h_2*,. . . ,and* h_{n-1} *units (*h_i *is the i-th hidden layer of encoder), respectively.*

After conducting additional experiments (in Appendix), the structure [500, 500, 3] of the VAE demonstrates satisfactory performance in both reconstruction and generation tests. We will employ this structure to construct our VAE, BVAE, and CVAE models for benchmarking they have been trained on 1.2×10^8 samples (2000 epochs). Specifically, we set $\beta = 5$ in BVAE and transform labels into one-hot vectors as conditional inputs in CVAE. It should be made clear that the loss function and intrinsic principles of VQVAE are quite different from those of the others (for VQVAE, we use the norm-2 reconstruction loss and employ convolutional layers to build the model). Therefore, we only compare its performance with the others in the reconstruction tast.

4.1 Reconstruction

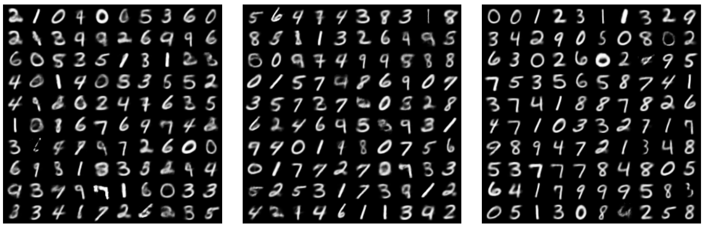


Original VAE BVAE CVAE VQVAE

Figure 11: Reconstruction

We find that the VQVAE performs the best, with the CVAE coming in second in the reconstruction task.

4.2 Generation



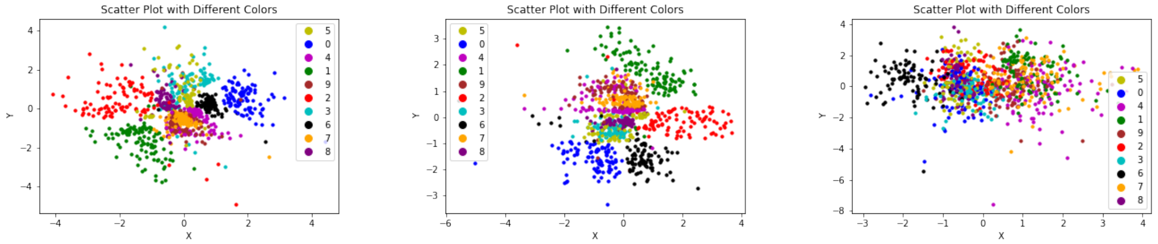
VAE BVAE CVAE

Figure 12: Generation

We find that CVAE perform best in generation tast. (Note that the generation is random, so we cannot ensure consistency.) VQVAE is not here since it alone can not be used to generate.

4.3 latent space

Let's make $d = 2$ and visualize the latent space of VAE, BVAE and CVAE. However, the embedding space of VQVAE consists of only 10 discrete points, as we've set the number of embedding vectors to be 10. We don't have a latent space for VQVAE that can be mapped to a 2-D space.

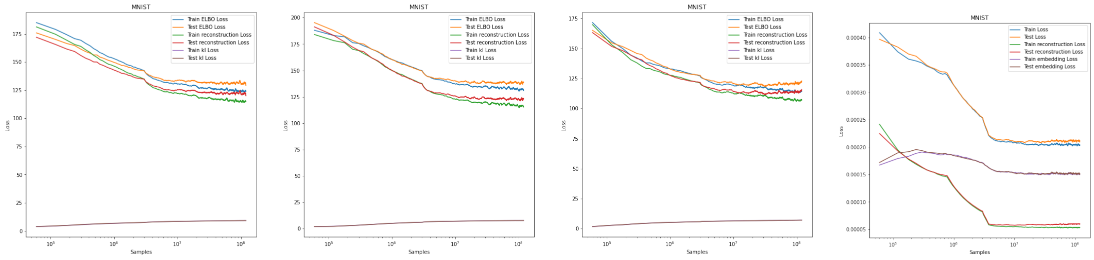


VAE BVAE CVAE

Figure 13: Latent space

Referring to Figure 13, each point represents a data point colored according to its label, ranging from 0 to 9. The coordinates of each point correspond to the latent variables z, which exist in a 2-D space ($d = 2$). It's observed that the latent spaces of VAE and BVAE exhibit relatively well-clustered points, while those of CVAE do not. This is because the CVAE model assumes that the latent variables are independent of the condition given (Recall that $p(z \mid c) = p(z)$ in its loss function).

4.4 Training Process and loss



VAE BVAE CVAE VQVAE

Figure 14: Training Process

There is no obvious overfitting observed for these four models. Notably, the embedding loss of VQVAE increases slightly at first, possibly because the optimizer focuses on the reconstruction loss initially. However, eventually, both the embedding loss and the reconstruction loss drop and converge.

Model	Reconstruction loss	KL loss	ELBO loss	Training time(min)
VAE	115.1	9.1	124.2	150
BVAE	116.1	7.8	131.7	148
CVAE	107.2	7.2	114.4	153

Table 1: Loss after convergence

Referring to Table 1, it's evident that CVAE achieves the best performance in both reconstruction and KL loss. The reason BVAE exhibits a larger reconstruction loss and smaller KL loss compared to VAE is due to our setting of $\beta = 5 > 1$. We didn't include VQVAE in this table because its loss function and intrinsic structure are completely different, making them incomparable.

In summary, CVAE stands out as our best model among these VAEs! (VQVAE is defined as an AE, not a VAE here.)

5 Minimal complexity architecture for MNIST

5.1 What's minimal complecity architecture?

The definition of minimal complexity architecture is very flexible. We need to address certain problems, such as what constitutes the minimal complex model to achieve a specific performance level and what threshold defines this performance. Given the infinite number of structures available, it is impossible to enumerate them all. Thus, what appropriate limitations can we impose on the network architecture we seek to identify? The runtime for training each model is approximately 2 hours. How can we efficiently identify the minimal structure? Based on the implementations before and some additional experiments (in Appendix), we give the answers of these questions:

- The threshold for "good model": From the previous findings and additional experiments in the appendix, we know that the ELBO loss alone cannot serve as the threshold to evaluate the performance. We define a "good model" as a model with a reconstruction loss of ≤ 120 and a KL loss of ≤ 10 after being trained with 6×10^7 samples (1000 epochs). We chose a training epoch cutoff as we believe a minimal complexity model doesn't need too much time to train.

- Appropriate limitations imposed on VAE structure: We only use MLP, skipping convolutional layers, for simplicity. We will use the notations **h** in Definition 4.1 to represent the structure hereafter. It is widely accepted that the structure of the encoder and decoder are inverses. Also, $h_n \leq h_{n-1} \leq \cdots \leq h_1$ in model **h**. To align with the minimal complexity demand, we set the upper bound of the latent dimension, d (which is h_n), to be smaller than 3, as it will be easier to visualize the latent space and can improve the generation ability (Smaller d can enhance the generation performance). Besides, given that the dimension of MNIST pictures is 784, we set $h_i \leq 500$.
- Finding the minimal complexity architecture: We consider the depth of the model (n) as the most determining aspect of complexity, which can help us identify n first. Assuming a monotonic relationship between reconstruction loss and the number of units in the latent layer, we can use a binary search algorithm to identify the number of units in each layer.

In summary, we articulate our definition of minimal complexity structure, assumptions, and algorithm rigorously as follows:

Definition 5.1. *The model $\boldsymbol{h}$, as defined in Definition 4.1, has a minimal complexity architecture if, first, it is a "good model"(reconstruction loss ≤ 120 and KL loss ≤ 10 after being trained with 6×10^7 samples, which is 1000 epochs) with smallest n; and second, for any $1 \leq i \leq n$, $[h_1, \ldots, h_{i-1}, t, h_{i+1}, \ldots, h_n]$ is not a "good model" for any $t < h_i$.*

Assumption 5.1. *Assumptions and conditions:*

1. *We assume that there is a monotonic relationship between the reconstruction loss, KL loss, and the number of units in hidden layers. For two models with the same number of hidden layers (n), denoted as $\boldsymbol{h}^{(1)}$ and $\boldsymbol{h}^{(2)}$, if $\boldsymbol{h}_i^{(1)} \leq \boldsymbol{h}_i^{(2)}$ for any $1 \leq i \leq n$, then the reconstruction loss of $\boldsymbol{h}^{(1)}$ is larger than that of $\boldsymbol{h}^{(2)}$, while the KL loss of $\boldsymbol{h}^{(1)}$ is smaller than that of $\boldsymbol{h}^{(2)}$. (This observation is counterintuitive but has been consistently observed in experiments.)*
2. *To align with the widely used structure and the demand for minimal complexity architecture, we set $h_{i+1} \leq h_i$, $h_n \leq 3$, and $h_i \leq 500$, for $1 \leq i \leq n-1$.*

Given these definitions and assumptions, we design an algorithm to find the minimal complexity architecture **h** for a fixed n. First, we use [500, ..., 500, 3] as the upper bound to find the maximal complexity architecture (ensure KL loss ≤ 10) for a "good model" $[m_1, \ldots, m_n]$, then use $[m_1, \ldots, m_n]$ as the upper bound to find the minimal complexity architecture (ensure reconstruction loss ≤ 120). The two steps are completely similar; therefore, we only write down the second step as follows:

Algorithm 1: Finding minimal complexity architecture for a fixed n

```
Function Is_good_model($\boldsymbol{h}$):
    if $Reconstruction\_loss(\mathbf{h}) \leq 120$ and $KL\_loss(\mathbf{h}) \leq 10$ then
        return True
    return False
Function Binary_search(left, right, $\mathbf{h}$, $i$):
    ans = 0
    while left $\leq$ right do
        mid=(left+right)/2
        $\mathbf{H} = [h_1, \ldots, h_{i-1}, \text{mid}, h_{i+1}, \ldots, h_n]$
        if Is_good_model($\boldsymbol{H}$) then
            right = mid-1
            ans =mid
        else
            left = mid +1
    return ans
$\mathbf{h} = [m_1, m_2, \ldots, m_{n-1}, m_n]$
$h_n$ = Binary_search(1, $m_n$, $\mathbf{h}$, $n$)
for $i \leftarrow n - 1$ to 1 do
    $h_i$ = Binary_search($h_{i+1}$, $m_i$, $\mathbf{h}$, $i$)
    if ! $h_i$ then
        Output: Not Found
        Exit
Output: h
```

Let's prove the correctness of this algorithm: After finding the minimal complexity architecture **h**, if there exists i such that $[h_1, \ldots, h_{i-1}$, t, $h_{i+1}, \ldots, h_n]$ is a "good model" for a $t < h_i$, then we have $[m_1, \ldots, m_{i-1}$, t, $h_{i+1}, \ldots, h_n]$ is a "good model" too because of the monotonicity assumption. However, h_i should be the smallest t which makes $[m_1, \ldots, m_{i-1}$, t, $h_{i+1}, \ldots, h_n]$ a "good model". Then, we have the conflict, which finishes this proof.

5.2 Finding minimal complecity architecture

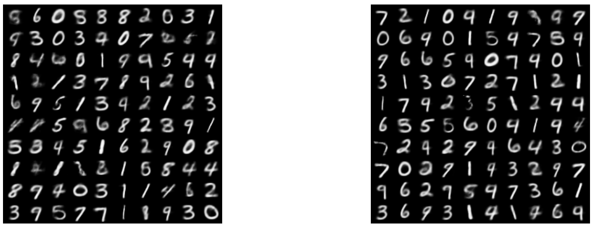
We start from $n = 2$. From experiments (in Appendix), we observe that the reconstruction loss of [500, 3] exceeds 120. Given the monotonicity assumption in condition 5.1, there exists no "good model" for for n=2.

Then we focus on $n = 3$ and employ the proposed algorithm 1 to find the minimal complexity architecture. Firstly, [500, 500, 3] is a "good model". According to the constraint outlined in condition 5.1 regarding the limitation on the number of units, [500, 500, 3] is the maximal complexity architecture for $n = 3$. Therefore we set $m_1 = 500$, $m_2 = 500$ and $m_3 = 3$ to initiate the binary search. The detailed process is outlined as follows:

Epoch	Model **h**	Reconstruction loss	KL loss
1	[500, 500, 3]	115.5	9.1
2	[500, 251, 3]	127.2	8.5
3	[500, 376, 3]	116.1	9.1
4	[500, 313, 3]	116.9	8.9
5	[500, 282, 3]	141.4	7.1
6	[500, 297, 3]	117.2	8.9
7	[500, 289, 3]	117.1	9.0
8	[500, 285, 3]	117.6	8.9
9	[500, 283, 3]	124.7	8.8
10	[500, 284, 3]	118.2	8.8
11	[392, 284, 3]	118.4	8.9
12	[337, 284, 3]	119.9	8.8
13	[310, 284, 3]	120.4	8.7
14	[323, 284, 3]	120.5	8.7
15	[330, 284, 3]	119.3	8.8
16	[326, 284, 3]	119.9	8.8
17	[324, 284, 3]	119.9	8.7

Table 2: Binary search process

Ultimately, we obtain our minimal complexity architecture: [324, 284, 3]! We have already proved the minimality. Now, we demonstrate its performance in both generation and reconstruction as follows:



Generation

Reconstruction

Figure 15: Performance of minimal complexity architecture

At last, there are still some potential issues with our approach. For instance, the monotonicity assumption may not always hold precisely due to random fluctuations. Additionally, questions remain regarding the definition of the minimal complexity architecture. This includes considerations how to define it when convolutional layers are present and why prioritize depth over the number of non-zero parameters. These concerns highlight areas for further investigation and refinement in our methodology.

6 Appendix

6.1 Math Derivations

6.1.1 ELBO loss

In this section, we discuss the derivations of the loss function. Our goal is to maximize $E[\log p(X)]$. As $E[\log p(X)]$ can be estimated by $\frac{1}{N}\sum_{i=1}^{N}\log p(x_i)$, where x_i is randomly sampled from the dataset, we just need to focus on $\log p(X)$. Note that $\log p(X)$ can be written as

$$\begin{aligned}
&\log p(X)\\
=&\int \log p(X) q(z \mid X;\phi)dz\\
=&\int \log\left(\frac{q(z \mid X;\phi)}{p(z \mid X;\phi)}\frac{p(z,X)}{q(z \mid X;\phi)}\right) q(z \mid X;\phi)dz\\
=&\int \log\left(\frac{q(z \mid X;\phi)}{p(z \mid X;\phi)}\right) q(z \mid X;\phi)dz + \int \log\left(\frac{p(z,X)}{q(z \mid X;\phi)}\right) q(z \mid X;\phi)dz\\
=&KL\left[q(z \mid X;\phi)||p(z \mid X;\phi)\right] + E_{z\sim q(z|X;\phi)}\left[\log\left(\frac{p(z,X)}{q(z \mid X;\phi)}\right)\right]\\
=&KL\left[q(z \mid X;\phi)||p(z \mid X;\phi)\right] + ELBO,
\end{aligned}$$

where $ELBO := E_{z\sim q(z|X;\phi)}\left[\log\left(\frac{p(z,X)}{q(z|X;\phi)}\right)\right]$ is the evidence lower bound ($\log(p(X)) \geq ELBO$). So $ELBO$ is what we aim to maximize. 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:

$$\begin{aligned}
ELBO &= E_{z\sim q(z|X;\phi)}\left[\log\left(\frac{p(z,X)}{q(z \mid X;\phi)}\right)\right]\\
&= \int \log\left(\frac{p(z,X)}{q(z \mid X;\phi)}\right) q(z \mid X;\phi)dz\\
&= \int \log\left(\frac{p(X \mid z;\theta)p(z)}{q(z \mid X;\phi)}\right) q(z \mid X;\phi)dz\\
&= \int \log p(X \mid z;\theta) q(z \mid X,\phi)dz - \int \log\left(\frac{q(z \mid X,\phi)}{p(z)}\right) q(z \mid X,\phi)dz\\
&= E_{z\sim q(z|X;\phi)}\left[\log p(X \mid z;\theta)\right] - KL\left[q(z \mid X;\phi)||p(z)\right]\\
&= -Loss_{reconstruction} - Loss_{KL}
\end{aligned}$$

We first focus on $Loss_{KL}$, which is $KL\left[q(z \mid X;\phi)||p(z)\right]$. Given $z \mid X,\phi \sim N(\mu_E, \sigma_E^2 I)$ and $z \sim N(0, I)$, we have

$$\int \log q(z \mid X;\phi) q(z \mid X;\phi)dz = -\frac{d}{2}\log 2\pi - \frac{1}{2}\sum_{i=1}^{d}(1+\log\sigma_{E,i}^2)$$

$$\text{and} \quad \int \log p(z)q(z \mid X;\phi)dz = -\frac{d}{2}\log 2\pi - \frac{1}{2}\sum_{i=1}^{d}(\mu_{E,i}^2 + \sigma_{E,i}^2),$$

where $\mu_{E,i}$ and $\sigma_{E,i}^2$ represent the i-th element of μ_E and σ_E^2, respectively. Thus,

$$\begin{aligned} KL\left[q(z \mid X;\phi)||p(z)\right] &= \int \log q(z \mid X;\phi)q(z \mid X;\phi)dz - \int \log p(z)q(z \mid X;\phi)dz \\ &= -\frac{1}{2}\sum_{i=1}^{d}(1 + \log\sigma_{E,i}^2 - \mu_{E,i}^2 - \sigma_{E,i}^2). \end{aligned}$$

Then we need to estimate $Loss_{reconstruction} = E_{z\sim q(z|X;\phi)}\left[\log p(X \mid z;\theta)\right]$. Using Monte Carlo method, it can be estimated by $\frac{1}{L}\sum_{i=1}^{L}\log p(X \mid z_l;\theta)$, where z_l are randomly sampled from the distribution $N(\mu_E, \sigma_E^2)$. We set $L = 1$ in the implementation. Therefore, we focus on $\log p(X \mid z;\theta)$ here.

We have used the Gaussian output as the encoder, while there are different choices for the decoder. We discuss the choices of the decoder in the following:

- $X \mid z;\theta \sim N(\mu_D, \sigma_D^2 I)$, where $\mu_D \in R^m$ and $\sigma_D^2 \in R^m$(or just a single number) are both the outputs of the decoder. This distribution is often used when the data is continuous. $\log p(X \mid z;\theta)$ can be calculated straightforwardly as:

$$-\frac{1}{2}(d\log 2\pi + \sum_{i=1}^{m}\log\sigma_{D,i}^2 + \frac{||X-\mu_D||^2}{\prod_{i=1}^{m}\sigma_{D,i}^2}),$$

  where $\sigma_{D,i}^2$ is the i-th element of σ_D^2. However, if the picture has some "constant points", such as the boundary of MNIST picture always being 1(black), the estimation of some elements in σ_D^2 will tend to be 0. This, in turn, can lead to an oversize of loss function.

- $X \mid z;\theta \sim N(\mu_D, \lambda I)$, where λ is a hyperparameter. Then $\log p(X \mid z;\theta)$ is given by:

$$-\frac{1}{2}(d\log 2\pi + \log\lambda + \frac{||X-\mu_D||^2}{\lambda}).$$

  This approach can reduce the complexity of the network and avoid the issue of an oversized loss function. However, it takes effort to choose an appropriate λ.

- $X \mid z;\theta \sim Bernoulli(\mu_D)$. Then $\log p(X \mid z;\theta)$ is

$$\sum_{i=1}^{m}X^{(i)}\log\mu_{D,i} + (1 - X^{(i)})\log(1-\mu_{D,i}),$$

  where $X^{(i)}$ is the i-th element of X. We used this approach to design our reconstruction function in the MNIST data experiment.

6.1.2 Marginal likelihood estimator

Our previous objective is $E[\log p(X)]$, the marginal likelihood, which can be estimated by $\frac{1}{N}\sum_{i=1}^{N}\log p(x_i)$, where x_i are randomly sampled from the whole dataset. For each i, we estimate $p(x_i)$ through:

$$
\begin{aligned}
\frac{1}{p(x_i)} &= \int \frac{q(z)}{p(x_i)} dz \\
&= \int \frac{q(z)q(z \mid x_i; \phi)}{p(x_i, z)} dz \\
&= \int \frac{q(z)}{p(z)p(x_i \mid z; \theta)} q(z \mid x_i; \phi) dz \\
&\approx \frac{1}{L} \sum_{l=1}^{L} \frac{q(z_l)}{p(z_l)p(x_i \mid z_l; \theta)}, \text{ where } z_l \sim q(z \mid x_i; \phi).
\end{aligned}
$$

Accroding to Bayesian methodology, we align $q(z)$ with the posterior probability. Thus we set $q(z)$ as $q(z \mid X; \phi)$. Then $p(x_i)$ can be estimated by:

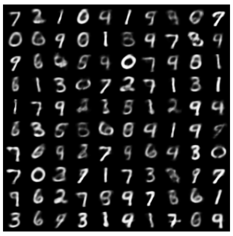
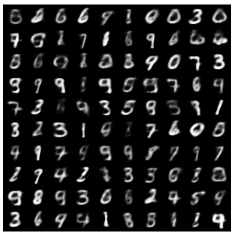
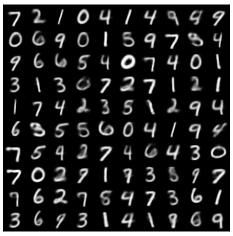
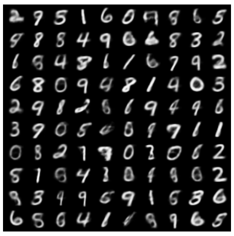
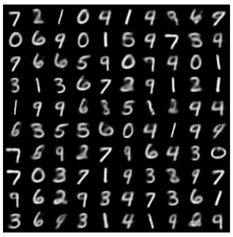
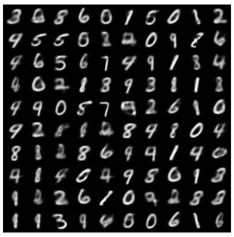
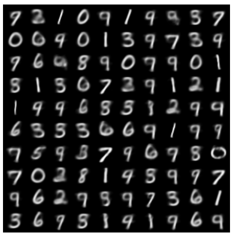
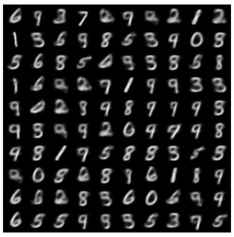
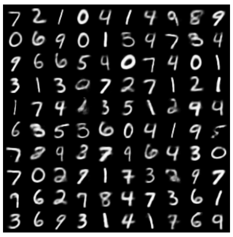
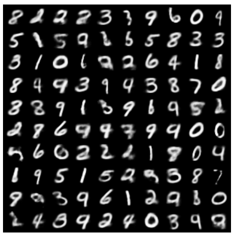
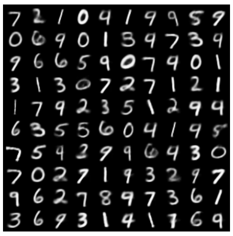
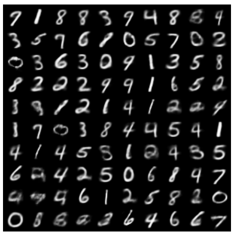
$$
\left(\frac{1}{L} \sum_{l=1}^{L} \frac{exp\left\{ -\frac{1}{2} \sum_{j=1}^{d} \frac{(\mu_{E,j} - z_l^{(j)})^2}{2\sigma_{E,j}^2} + \frac{1}{2} ||z_l||^2 - \sum_{j=1}^{m} \left[x_i^{(j)} \log \mu_{D,j} + (1 - x_i^{(j)}) \log(1 - \mu_{D,j}) \right] \right\}}{\prod_{j=1}^{d} \sigma_{E,j}} \right)^{-1},
$$

where $z_l^{(j)}$ is the j-th element of z_l, which is sampled from $N(\mu_E, \sigma_E^2 I)$. In the MNIST implementation, we set $N = 1000$ and $L = 50$.

6.2 Additional experiments

We conducted some preliminary experimentation to understand the influence of the VAE network structure on ELBO loss, reconstruction loss, KL loss, and the quality of generated pictures (see Table 3). Models are all trained with 6×10^7 samples. We summarize our findings in the following:

- The reconstruction ability is influenced by the reconstruction loss, while both the reconstruction and generation losses affect the generation ability. To evaluate a model, relying solely on ELBO loss is insufficient; We should set thresholds for both reconstruction and KL losses to choose apropriate model.
- The increase in the number of units in each network layer typically improves the reconstruction power. However, this improvement may come at the cost of deteriorating the generation ability due to potential conflicts between these two types of loss. Specifically, increasing the latent dimension (d) can degrade the generation ability, possibly because of the challenge in aligning two distributions when d is relatively large. However, this difficulty can be mitigated by increasing the depth of the network.
- The network [500, 500, 3] demonstrates satisfactory performance in both reconstruction and generation tests. We will employ this structure to construct our VAE, BVAE, and CVAE models for benchmarking purposes.
- Based on these results, we define a ”good model” as a model with a reconstruction loss of ≤ 120 and a KL loss of ≤ 10 after being trained with 6×10^7 samples.

Structure	Loss	Reconstruction	Generation
[50, 4]	ELBO: 144.5 Reconst: 135.4 KL: 9.1		
[500, 4]	ELBO: 124.2 Reconst: 114.0 KL: 10.2		
[500, 3]	ELBO: 137.9 Reconst: 130.7 KL: 7.2		
[500, 2]	ELBO: 144.9 Reconst: 138.5 KL: 6.4		
[500, 500, 5]	ELBO: 113.6 Reconst: 100.9 KL: 12.7		
[500, 500, 4]	ELBO: 121.5 Reconst: 111.1 KL: 10.4		

[500, 500, 3]	ELBO: 124.6 Reconst: 115.5 KL: 9.1		
[500, 500, 2]	ELBO: 137.7 Reconst: 130.4 KL: 7.3		
[500, 100, 5]	ELBO: 125.5 Reconst: 115.4 KL: 10.1		
[500, 100, 50, 10, 5]	ELBO: 154.8 Reconst: 150.5 KL: 4.3		
[500, 100, 50, 10, 2]	ELBO: 155.1 Reconst: 150.9 KL: 4.2		

Table 3: Performance of different network

6.3 Resources

- Frey Face data: https://cs.nyu.edu/~roweis/data/
- All codes: https://github.com/yinjjiew/Variational-Autoencoder