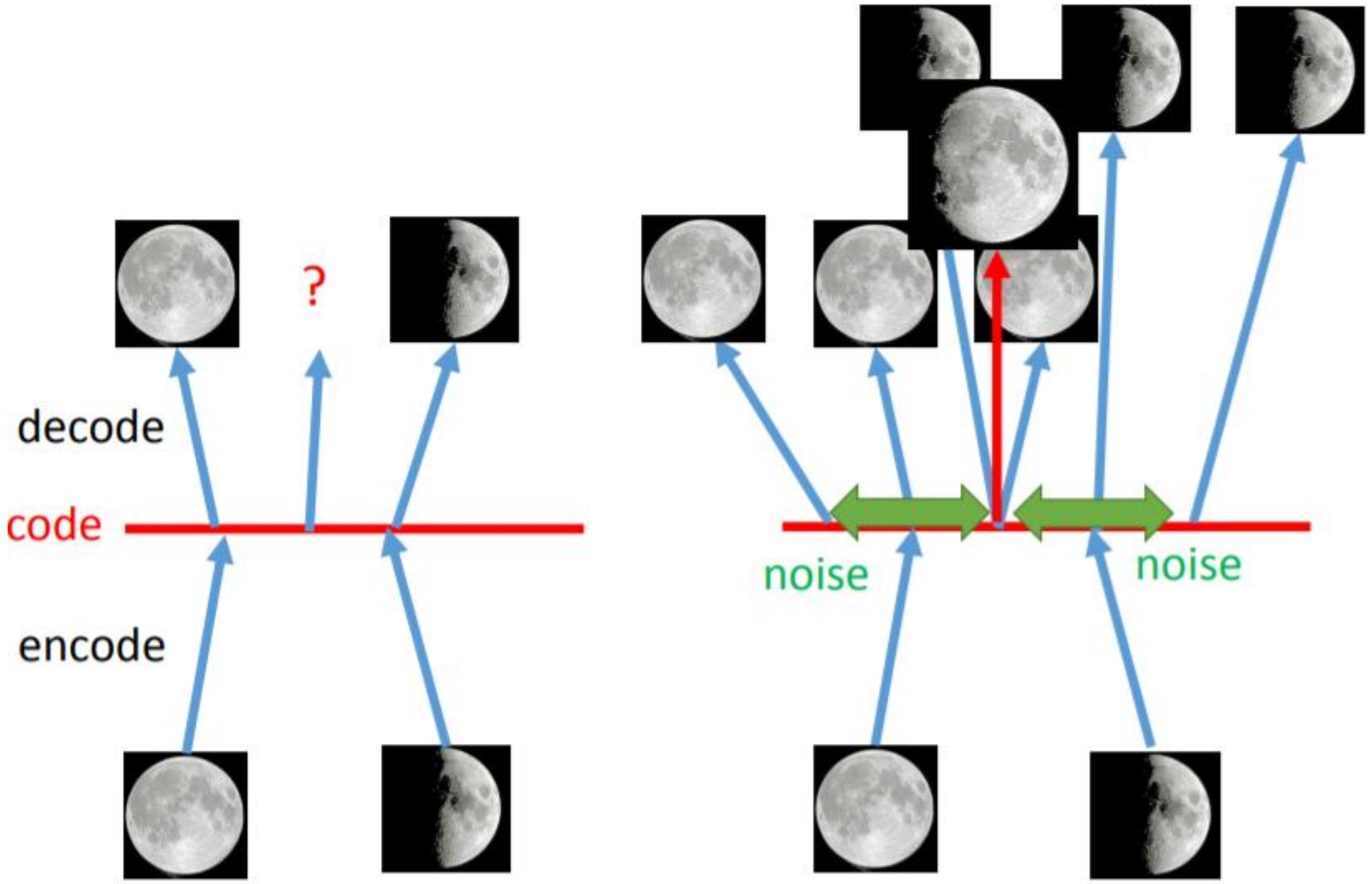
Variational Auto-Encoder (VAE)

Why VAE?

Assume 1-d code

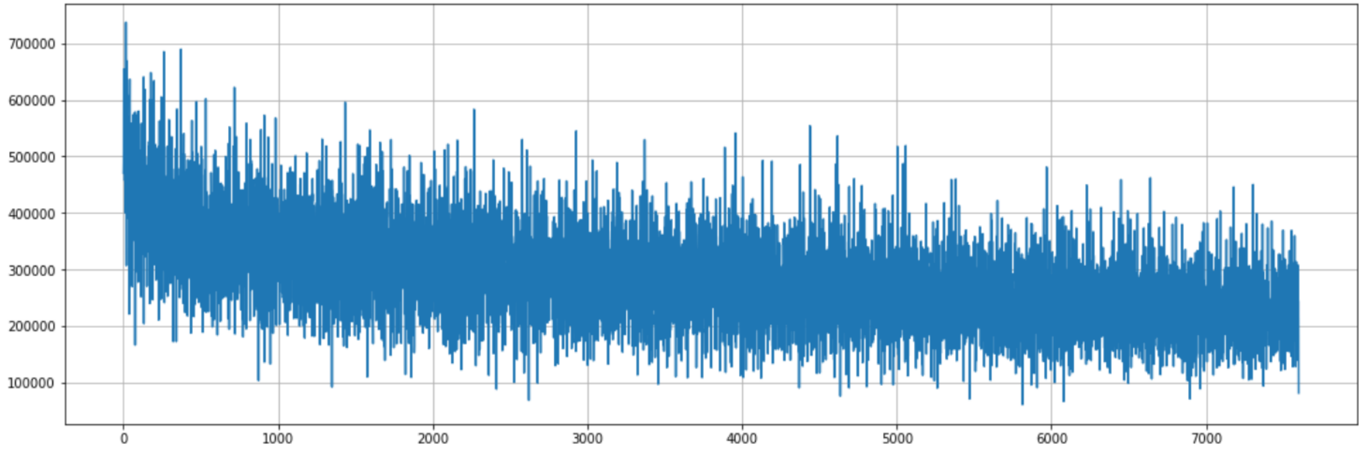


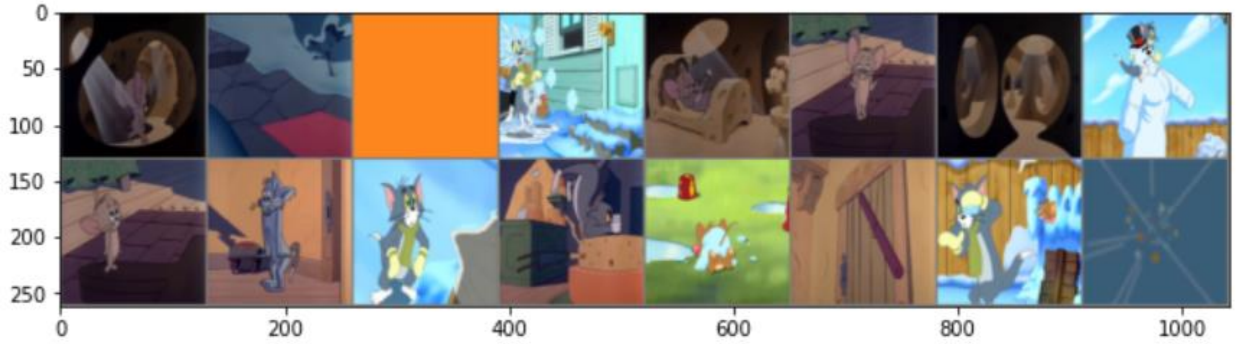
Reference: 李弘毅 ML Lecture 18 https://youtu.be/8zomhgKrsmQ

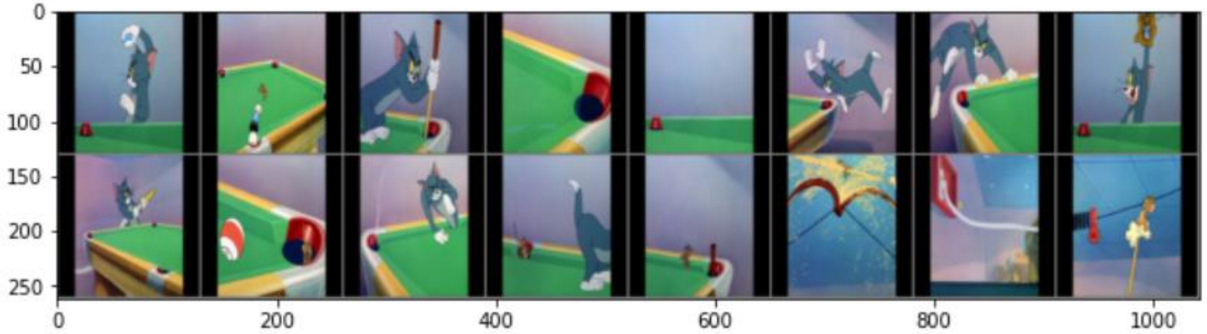
Practice

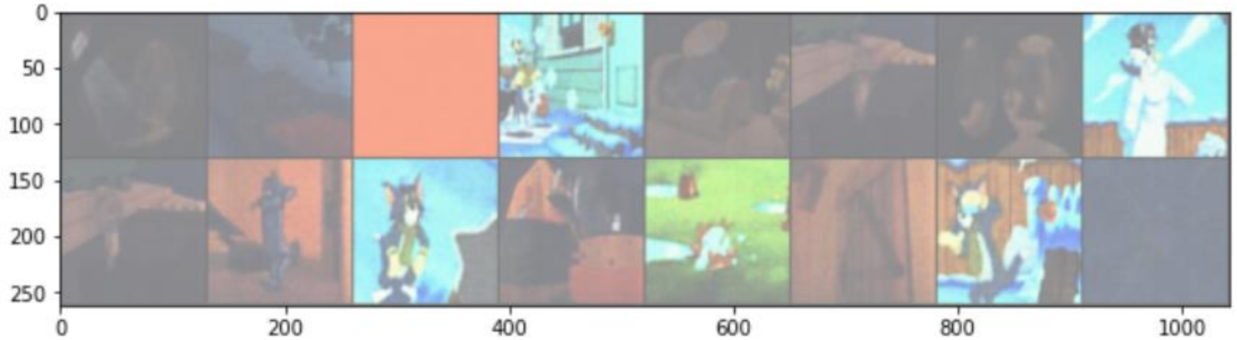
- Run “7.2.Conv_VAE.ipynb"

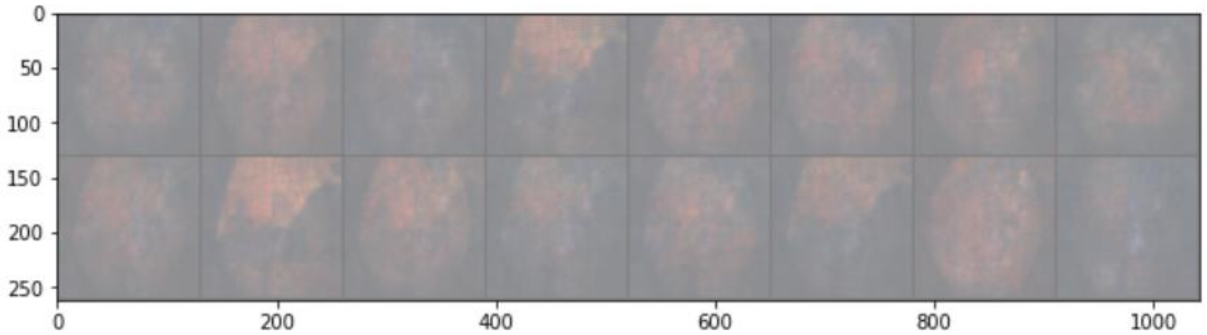
Train 400 epochs







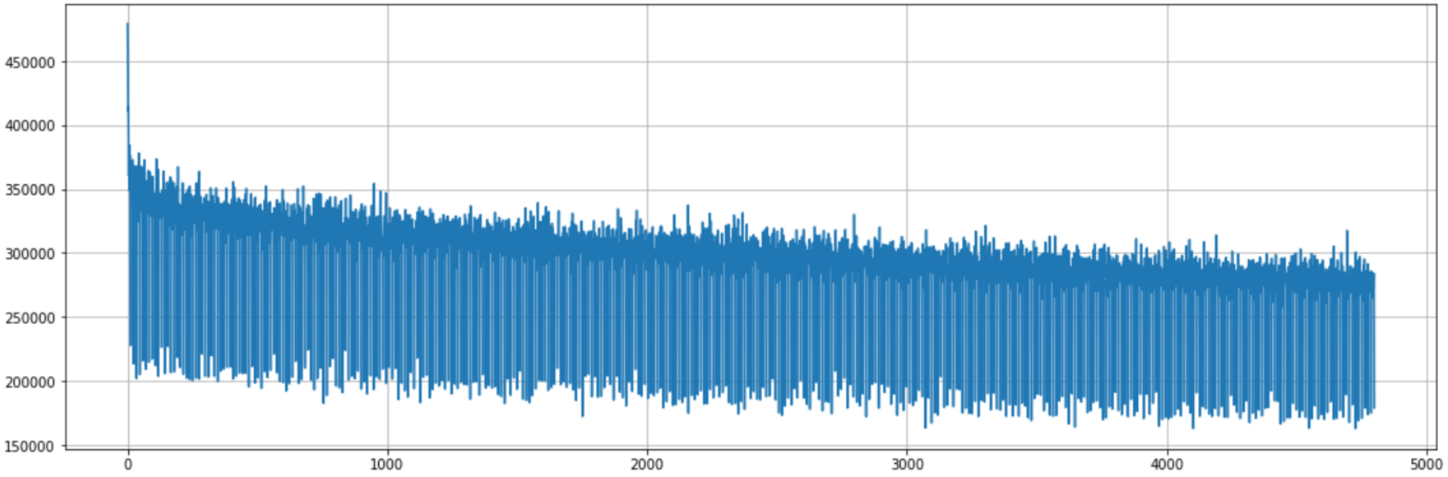


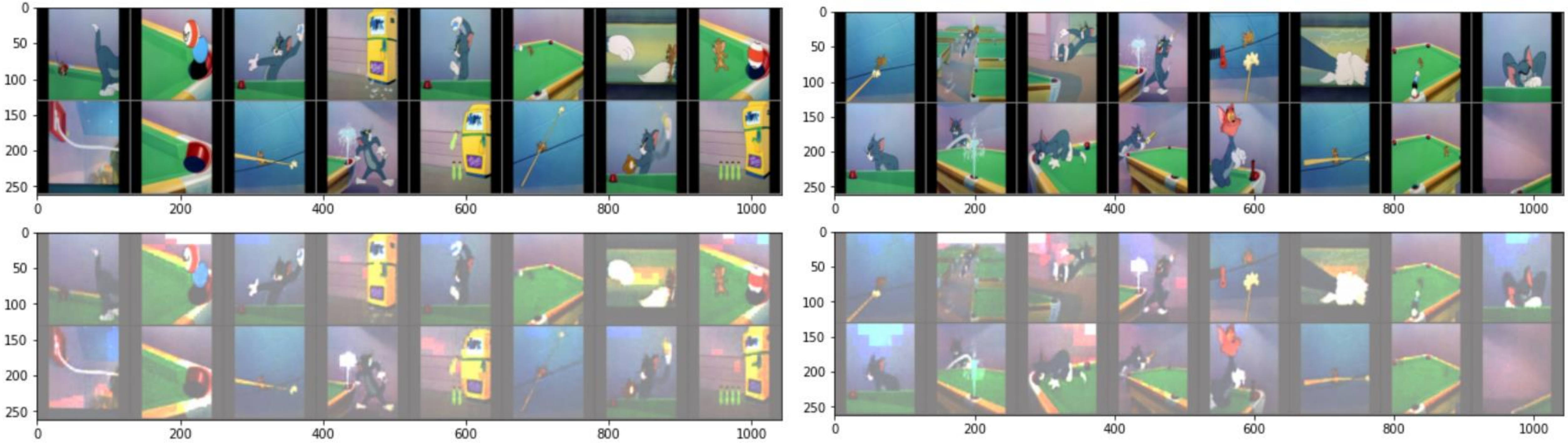


Able to recover the training images

Fails to reconstruct the test images

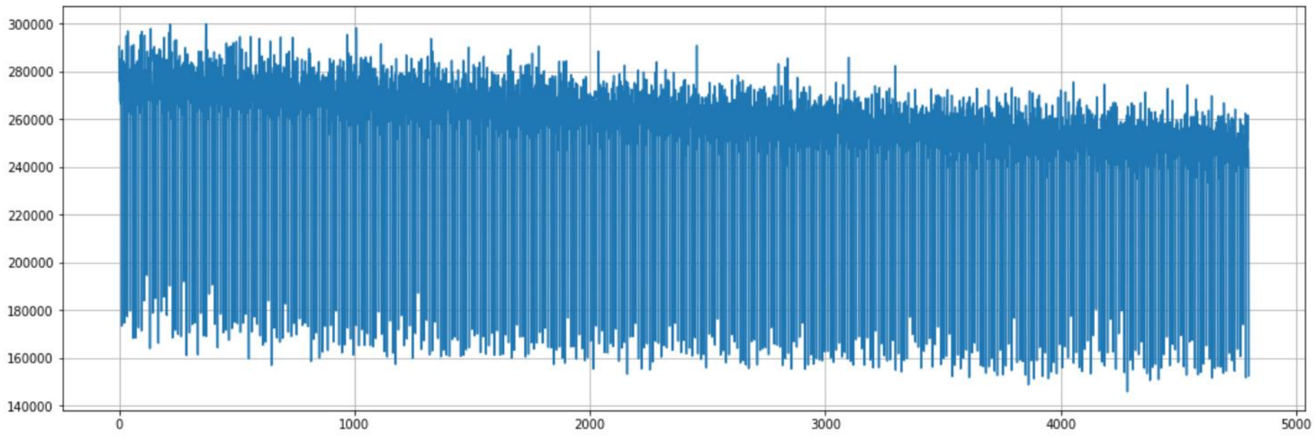
400+400 epochs (total = 800 epochs)

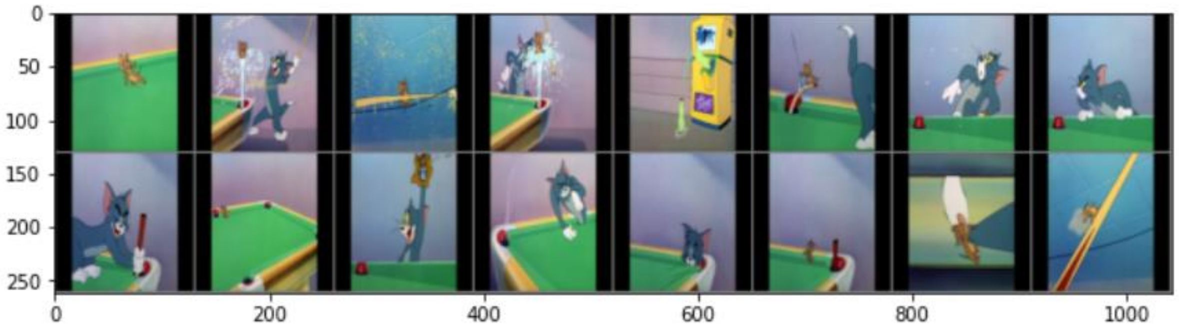


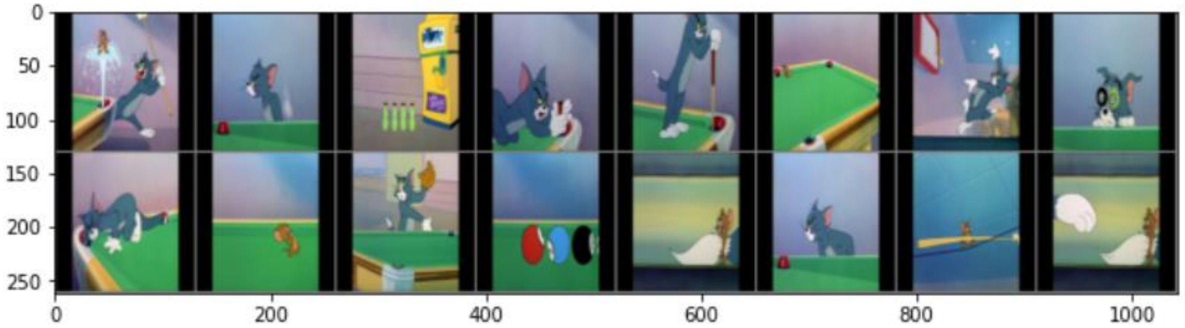


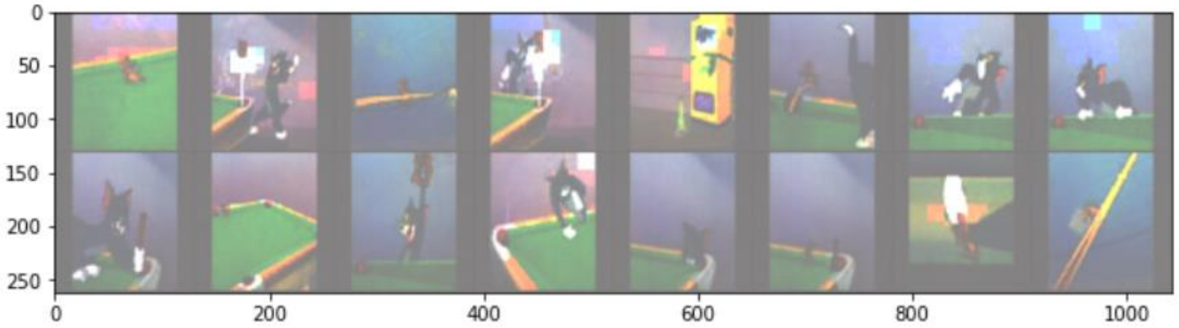
able to reconstruct the test images

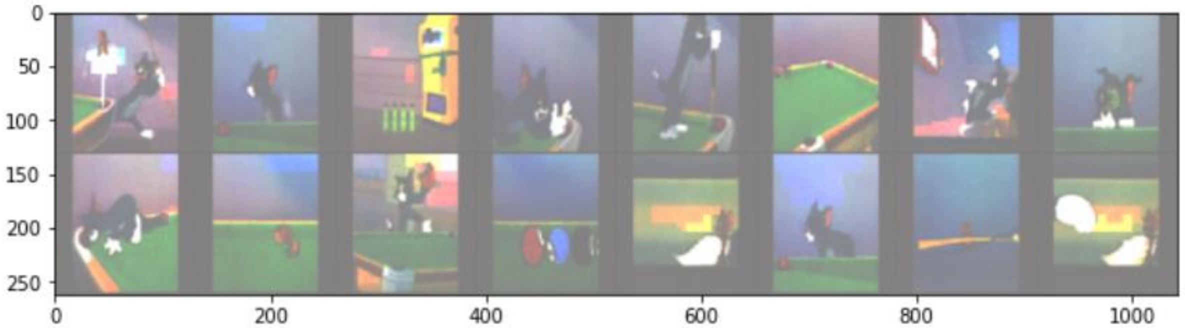
400+400+400 epochs (total = 1200 epochs)



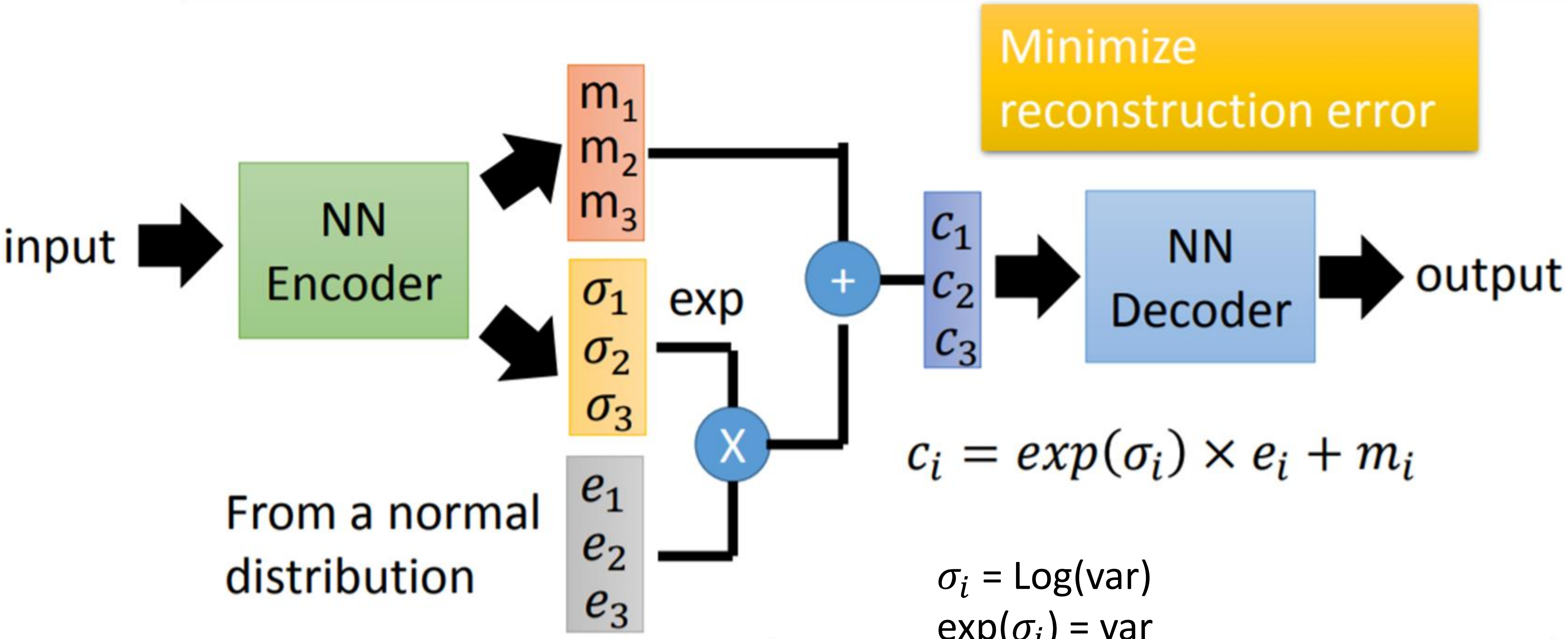








VAE



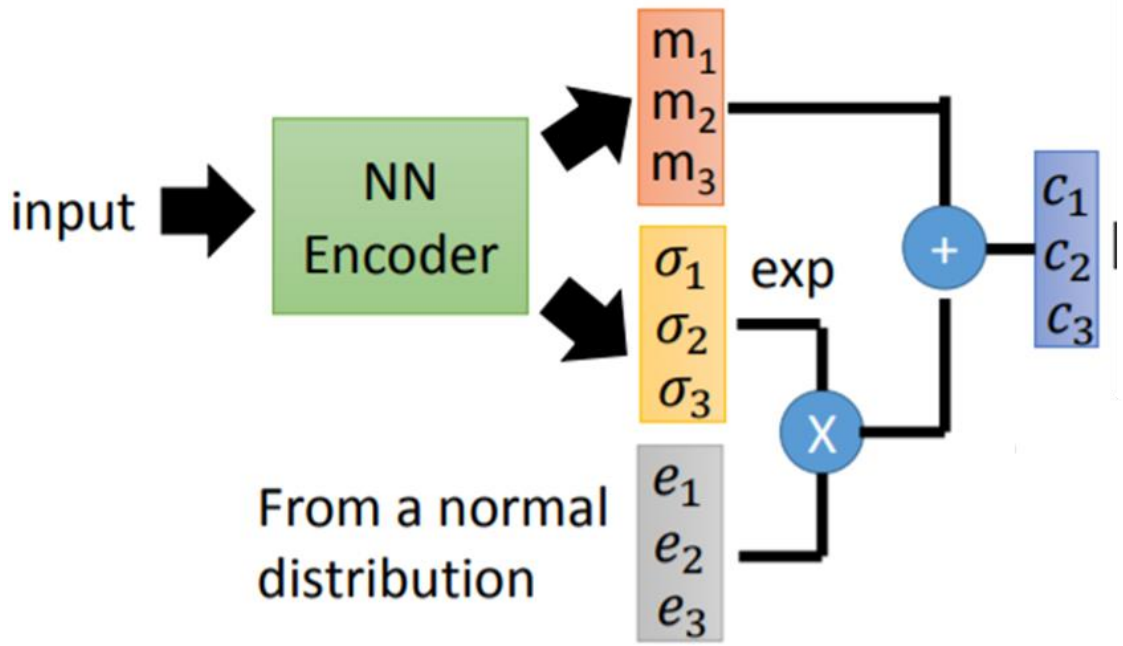
σ_i = Log(var)
exp(σ_i) = var

Encoder

```
[15]: for batchX, _ in loader:
          break;
      print(batchX.shape)
```

```
torch.Size([16, 3, 128, 128])
```

```
(fc1): Linear(in_features=1024, out_features=64,
(fc2): Linear(in_features=1024, out_features=64,
(fc3): Linear(in_features=64, out_features=1024,
```




```
[16]: h = model.encoder(batchX.to(device))
      print(h.shape)
```

```
torch.Size([16, 1024])
```

```
[17]: mu=model.fc1(h)
      print(mu.shape)
```

```
torch.Size([16, 64])
```

```
[18]: logvar=model.fc2(h)
      print(logvar.shape)
```

```
torch.Size([16, 64])
```

```
[19]: std = logvar.mul(0.5).exp_()
      print(std.shape)
```

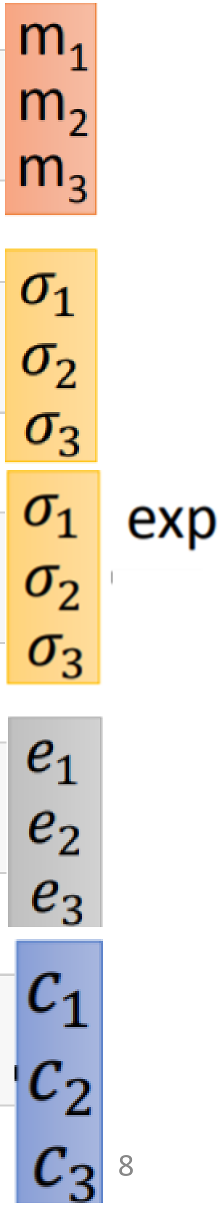
```
torch.Size([16, 64])
```

```
[20]: esp=torch.randn(*mu.size())
      print(esp.shape)
```

```
torch.Size([16, 64])
```

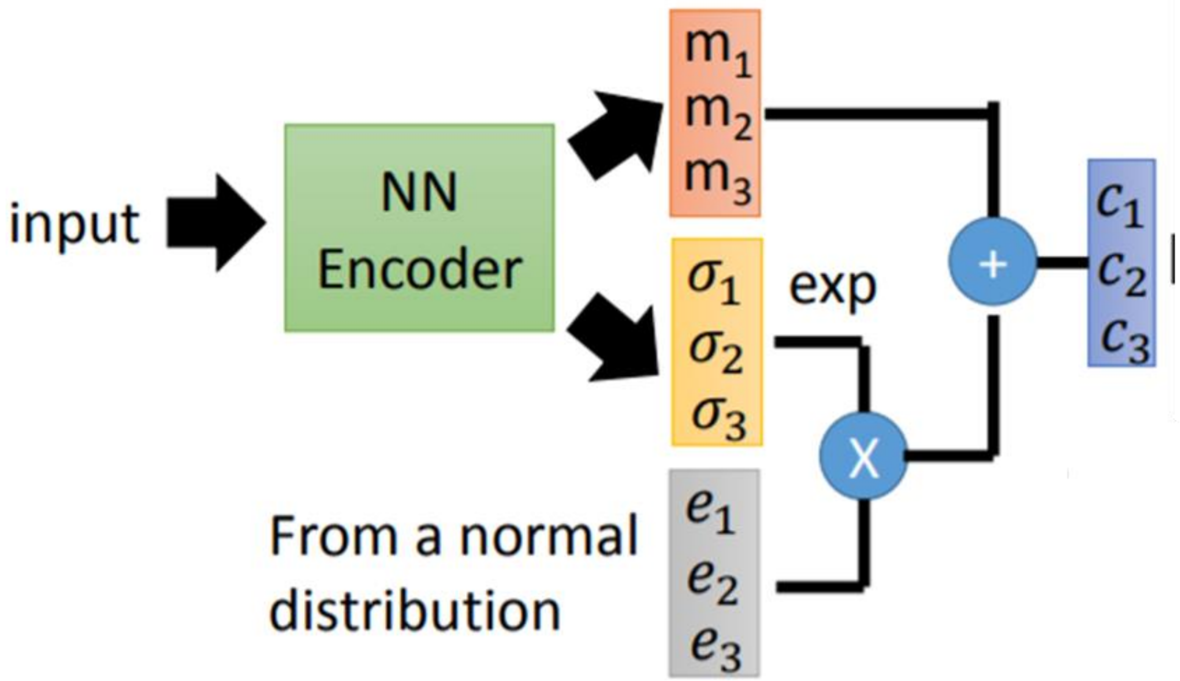
```
[21]: z=mu+std*esp.to(device)
      print(z.shape)
```

```
torch.Size([16, 64])
```



Loss function

σ_i = Log(var)



We want σ_i close to 0
(variance close to 1)

Minimize

$$\sum_{i=1}^{3}(exp(\sigma_i) - (1 + \sigma_i) + (m_i)^2)$$

L2 regularization

Loss function

[9]:
```
def loss_fn(recon_x, x, mu, logvar):
    #BCE = F.binary_cross_entropy(recon_x, x, size_average=False).to(device)
    MSE = F.mse_loss(recon_x, x, reduction='sum')
    # see Appendix B from VAE paper:
    # Kingma and Welling. Auto-Encoding Variational Bayes. ICLR, 2014
    # 0.5 * sum(1 + log(sigma^2) - mu^2 - sigma^2)
    KLD = -0.5*torch.mean(1+logvar-mu.pow(2)-logvar.exp()).to(device)
    return MSE+KLD, MSE, KLD
```

Minimize

$$\sum_{i=1}^{3}(exp(\sigma_i) - (1+\sigma_i) + (m_i)^2)$$

L2 regularization

[23]:
```
tensorY,mu,logvar = model(batchX.to(device))
print(tensorY.shape)
```

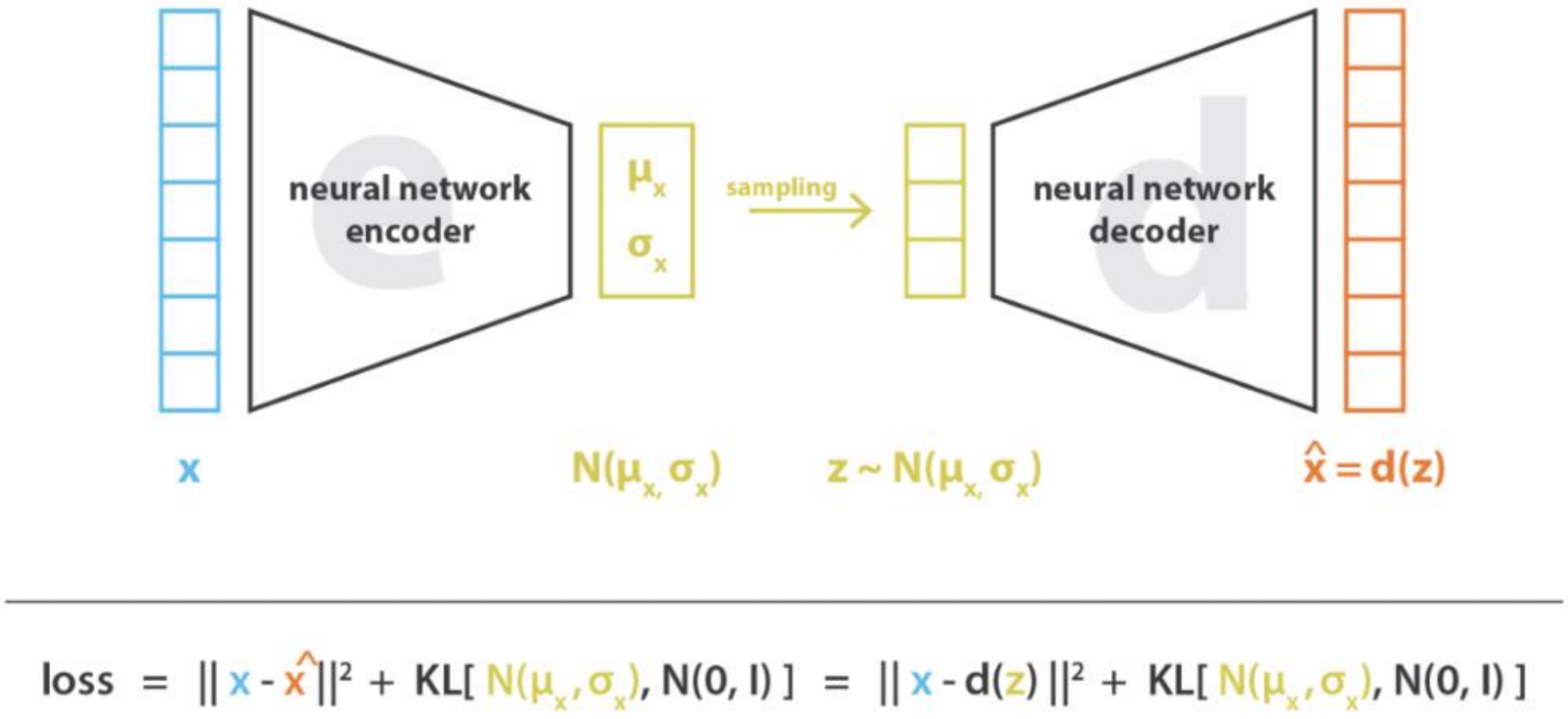
```
torch.Size([16, 3, 128, 128])
```

[24]:
```
loss, mse,kld = loss_fn(tensorY, batchX.to(device), mu, logvar)
print(loss)
```

```
tensor(627375.3750, device='cuda:0', grad_fn=<AddBackward0>)
```

Why loss = $MSE(x, \hat{x}) + KL(q(z|x)||P(z))$?

Source: https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

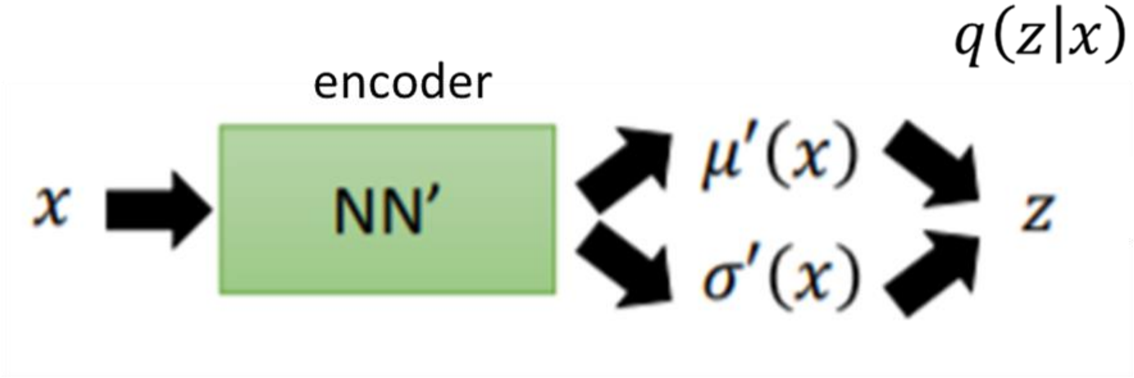


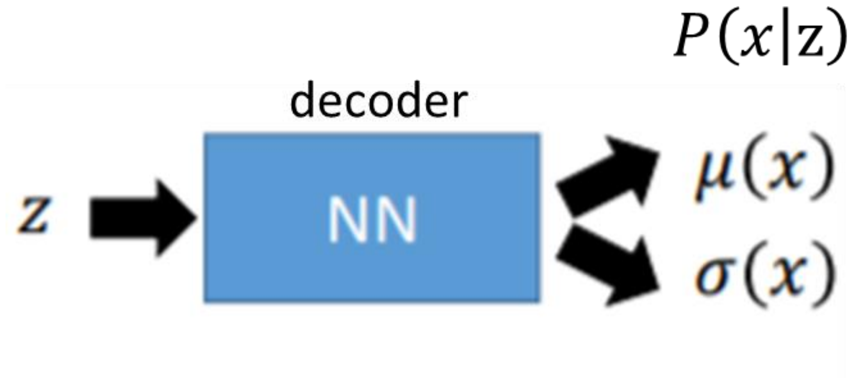
$$D_{KL}(p||q) = \sum_{i=1}^{N} p(x_i) log(\frac{p(x_i)}{q(x_i)})$$

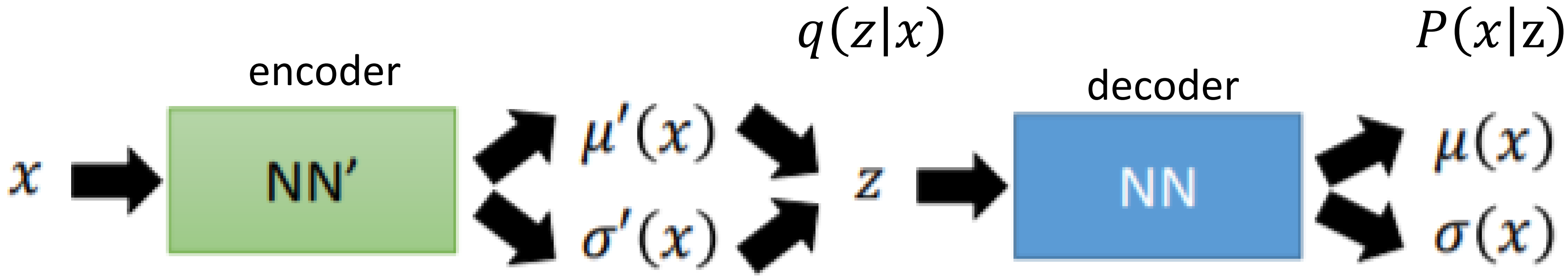
Minimize

$$\boxed{\sum_{i=1}^{3}(exp(\sigma_i) - (1 + \sigma_i) + (m_i)^2)}$$

The decoder and encoder of VAE model two conditional probability distributions







Gaussian mixture model

Gaussian Mixture Model

$$P(x) = \sum_m P(m)P(x|m)$$

How to sample?

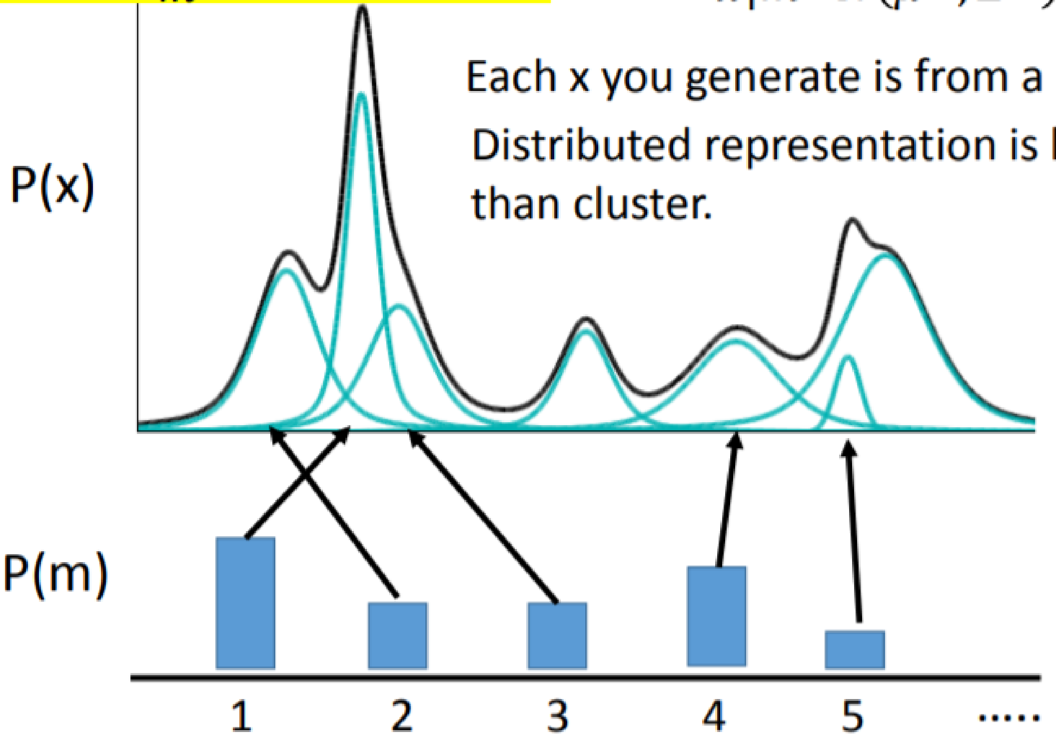
$m \sim P(m)$ (multinomial)

m is an integer

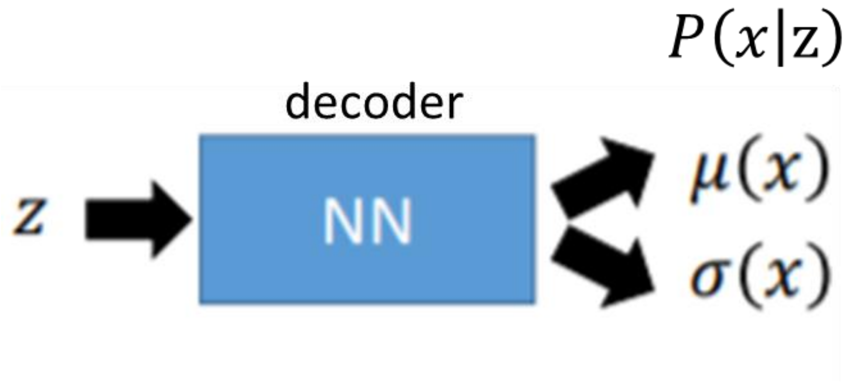
$x|m \sim N(\mu^m, \Sigma^m)$

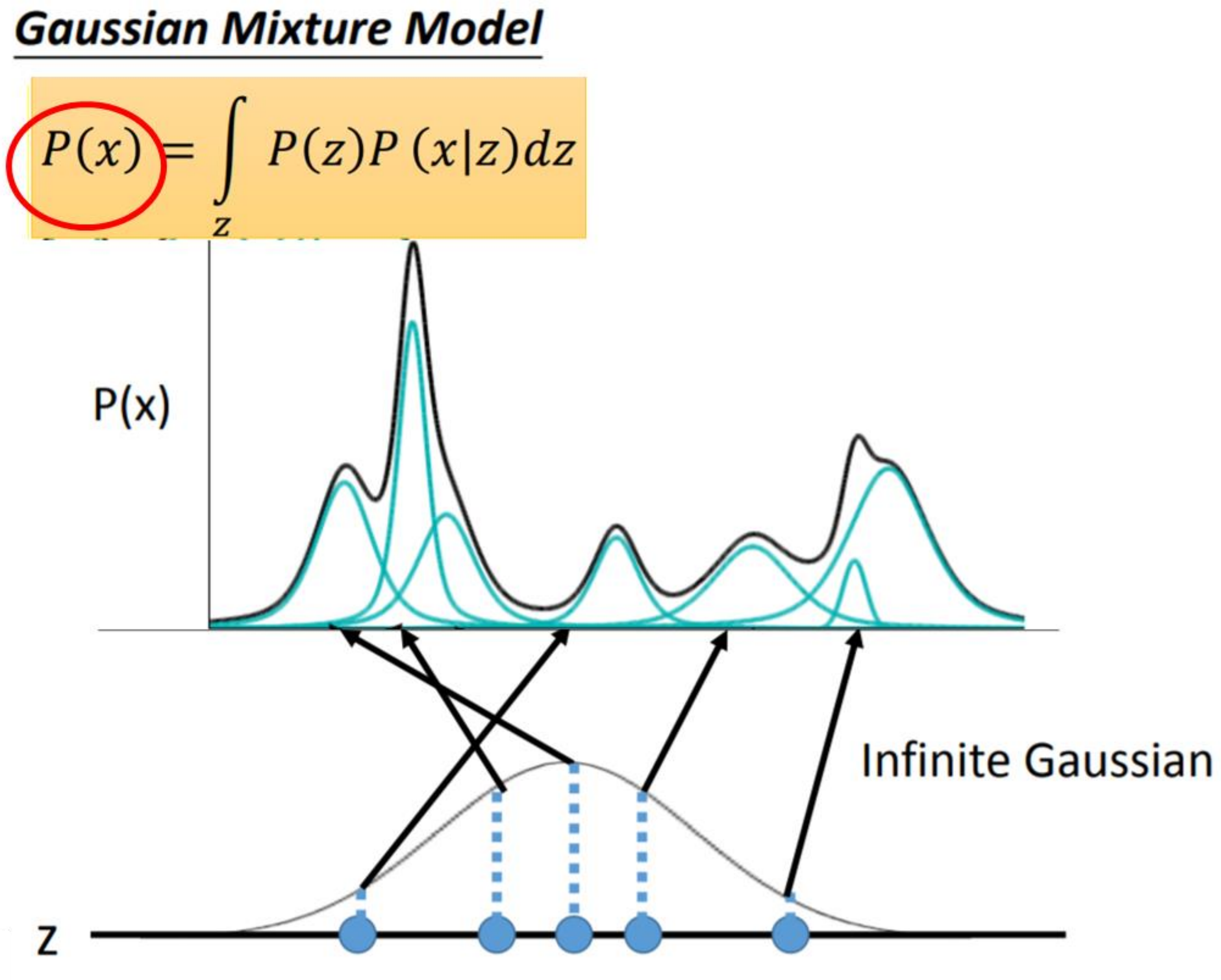
Each x you generate is from a mixture

Distributed representation is better than cluster.



The probability of sampling an output image *x* from latent vector space *z* can be modelled as a Gaussian mixture model





Reference: 李弘毅 ML Lecture 18 https://youtu.be/8zomhgKrsmQ

We want to train a decoder NN that can maximize the likelihood of observing the training images

Maximizing Likelihood

$$P(x) = \int_z P(z)P(x|z)dz$$

P(z) is normal distribution

$x|z \sim N(\mu(z), \sigma(z))$

$\mu(z), \sigma(z)$ is going to be estimated

$$L = \sum_x logP(x)$$

Maximizing the likelihood of the observed x

$$L = p(x^1) \times p(x^2) \times p(x^3) \times \cdots p(x^m) = \prod_{i=1,\dots,m} P(x^i)$$

Reference: 李弘毅 ML Lecture 18 https://youtu.be/8zomhgKrsmQ

Recap: maximize the likelihood of observing the classification of the training data

Training Data	x^1	x^2	x^3	……	x^N
	C_1	C_1	C_2		C_1

max $\quad L(w,b) = f_{w,b}(x^1) f_{w,b}(x^2) \left(1 - f_{w,b}(x^3)\right) \cdots f_{w,b}(x^N)$

min $\quad -lnL(w,b) = -lnf_{w,b}(x^1) - lnf_{w,b}(x^2) - ln\left(1 - f_{w,b}(x^3)\right) \cdots$

$\hat{y}^n$: 1 for class 1, 0 for class 2

$$= \sum_n -\underbrace{\left[\hat{y}^n lnf_{w,b}(x^n) + (1-\hat{y}^n) ln\left(1 - f_{w,b}(x^n)\right)\right]}_{\text{Cross entropy between two Bernoulli distribution}}$$

Reference: 李弘毅 ML Lecture 5 https://youtu.be/hSXFuypLukA

Rewrite the maximum likelihood item $\log P(x)$ as the summation of a lower bound L_b and KL divergence

$$\int_z q(z|x)dz = 1$$

$$logP(x) = \int_z q(z|x)logP(x)dz$$

q(z|x) can be any distribution

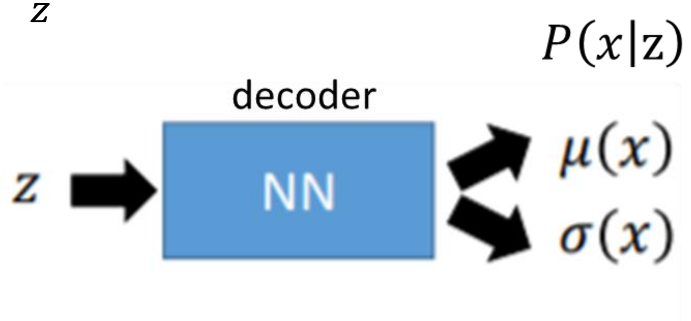
$$= \int_z q(z|x)log\left(\frac{P(z,x)}{P(z|x)}\right)dz = \int_z q(z|x)log\left(\frac{P(z,x)}{q(z|x)}\frac{q(z|x)}{P(z|x)}\right)dz$$

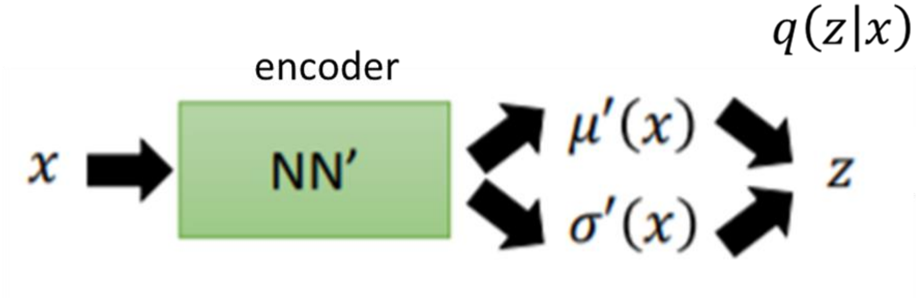
$$= \int_z q(z|x)log\left(\frac{P(z,x)}{q(z|x)}\right)dz + \underbrace{\int_z q(z|x)log\left(\frac{q(z|x)}{P(z|x)}\right)dz}_{KL(q(z|x)||P(z|x))}$$

≥ 0

$$\geq \int_z q(z|x)log\left(\frac{P(x|z)P(z)}{q(z|x)}\right)dz$$

lower bound L_b

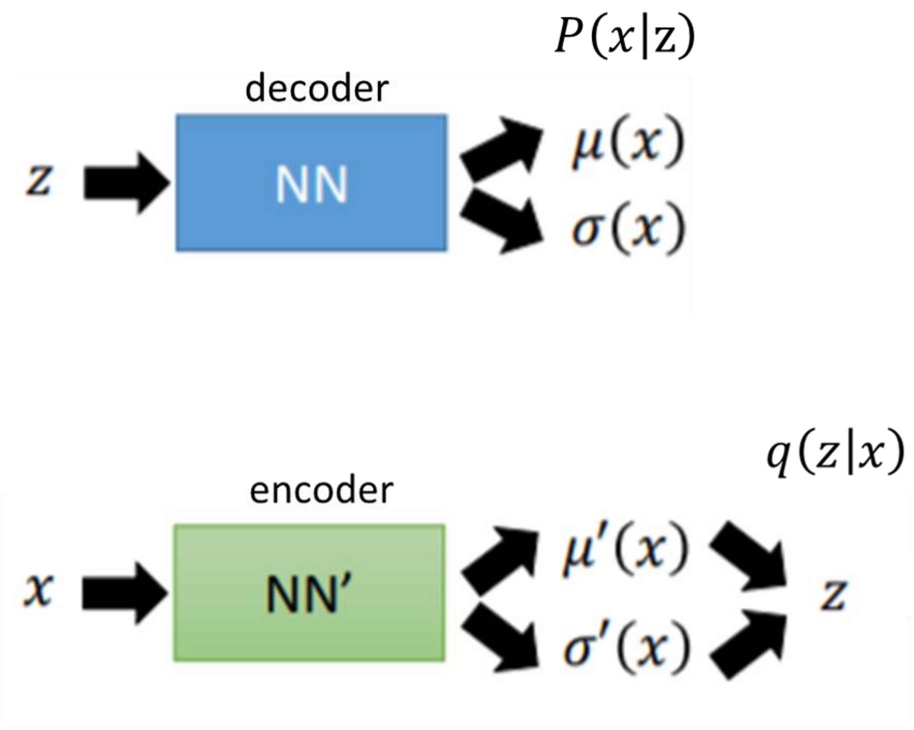




$$D_{KL}(q||p) = \sum_{i=1}^{N} q(x_i)\log\left(\frac{q(x_i)}{p(x_i)}\right)$$

Reference: 李弘毅 ML Lecture 18 https://youtu.be/8zomhgKrsmQ

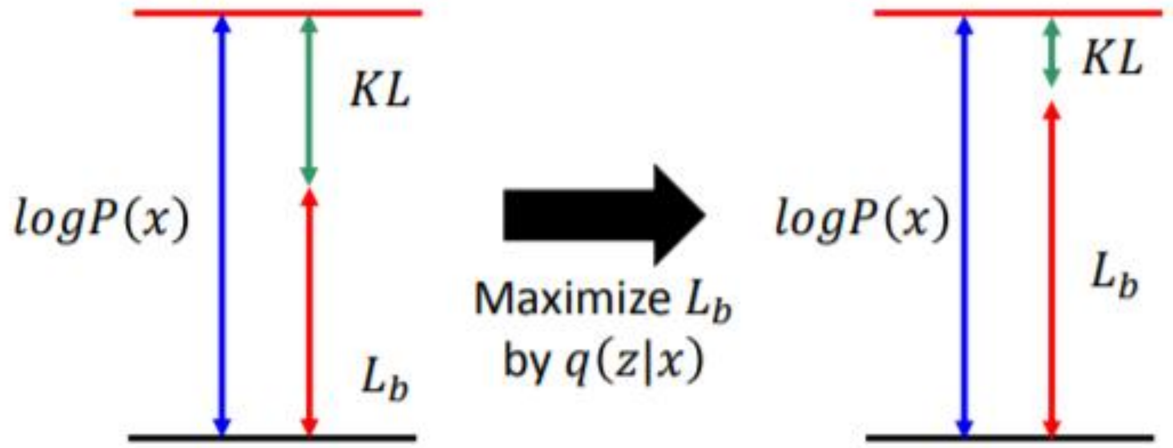
If we maximum L_b by adjusting $P(x|z)$ and $q(z|x)$ simutaneously, then we can maximum L_b and at the same time minimize the KL distance



$$logP(x) = L_b + KL(q(z|x)||P(z|x))$$

$$L_b = \int_z q(z|x)log\left(\frac{P(x|z)P(z)}{q(z|x)}\right)dz$$

Find $P(x|z)$ and $q(z|x)$ maximizing L_b



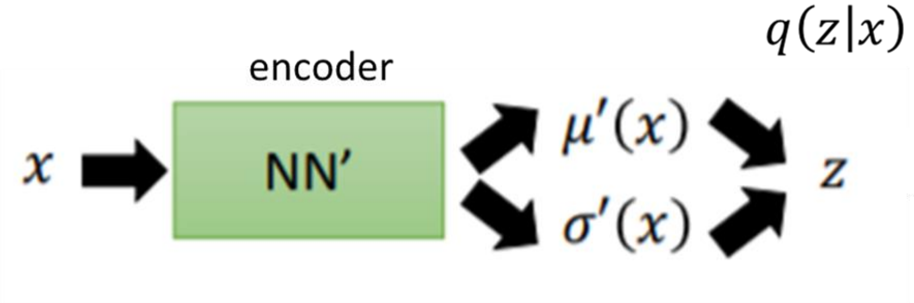
$q(z|x)$ will be an approximation of $p(z|x)$ in the end

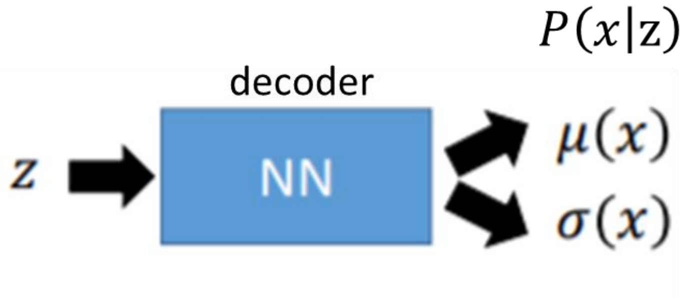
Reference: 李弘毅 ML Lecture 18 https://youtu.be/8zomhgKrsmQ

Rewrite the lower bound L_b as the summation of two terms: $-KL(q(z|x)||P(z))$, and $\int_z q(z|x)logP(x|z)dz$

max $$L_b = \int_z q(z|x)log\left(\frac{P(z,x)}{q(z|x)}\right)dz = \int_z q(z|x)log\left(\frac{P(x|z)P(z)}{q(z|x)}\right)dz$$

$$= \underbrace{\int_z \boxed{q(z|x)}log\left(\frac{P(z)}{\boxed{q(z|x)}}\right)dz}_{-KL(q(z|x)||P(z))} + \int_z q(z|x)logP(x|z)dz$$





Reference: 李弘毅 ML Lecture 18 https://youtu.be/8zomhgKrsmQ max. L_b can be done by min. $KL(q(z|x)||P(z))$ and max. $\int_z q(z|x)logP(x|z)dz$. That is why loss = KLD + MSE (x, $\hat{x}$)

Minimizing $KL(q(z|x)||P(z))$

Minimize

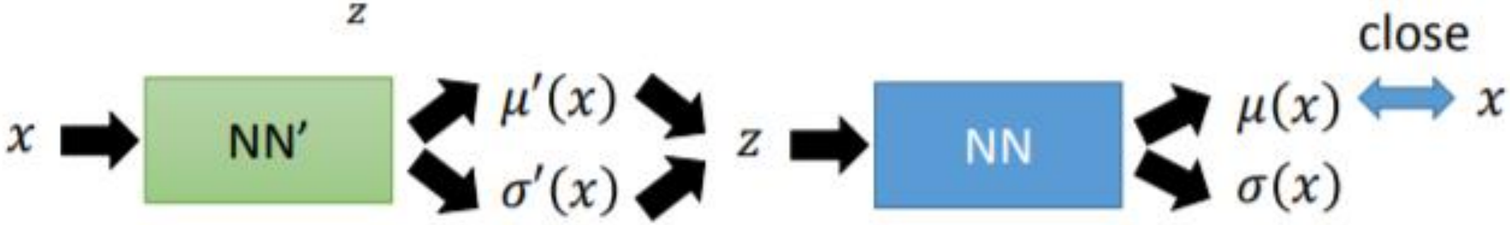
$$\sum_{i=1}^{3}(exp(\sigma_i) - (1+\sigma_i) + (m_i)^2)$$

(Refer to the Appendix B of the original VAE paper)

Maximizing

$$\int_z q(z|x)logP(x|z)dz = E_{q(z|x)}[logP(x|z)]$$

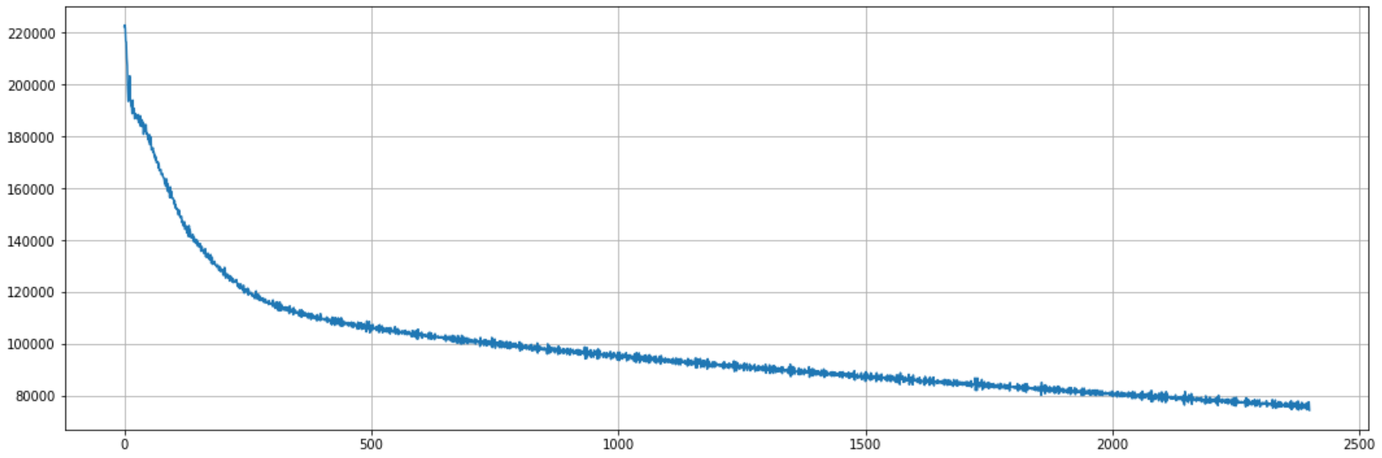
by min. $MSE(x, \hat{x})$



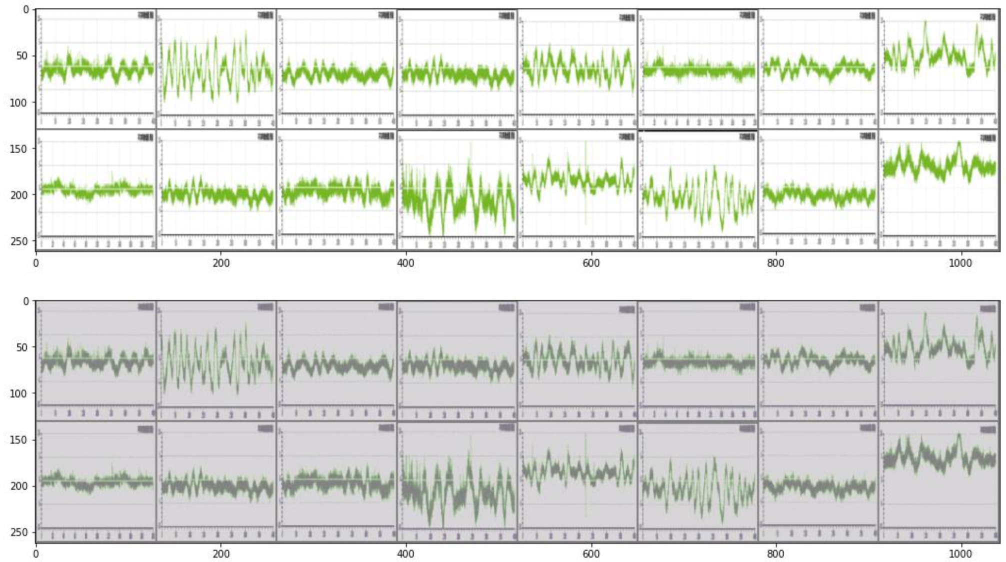
HW6 (2)

- Train an VAE to learn a compact representation (try latent vector of size 20, 30, 50) of your facial expression. Test with 10 happy and 10 angry faces.
- Show the recovered image.
- Send the latent vectors to *t*-SNE to see whether they form clusters.

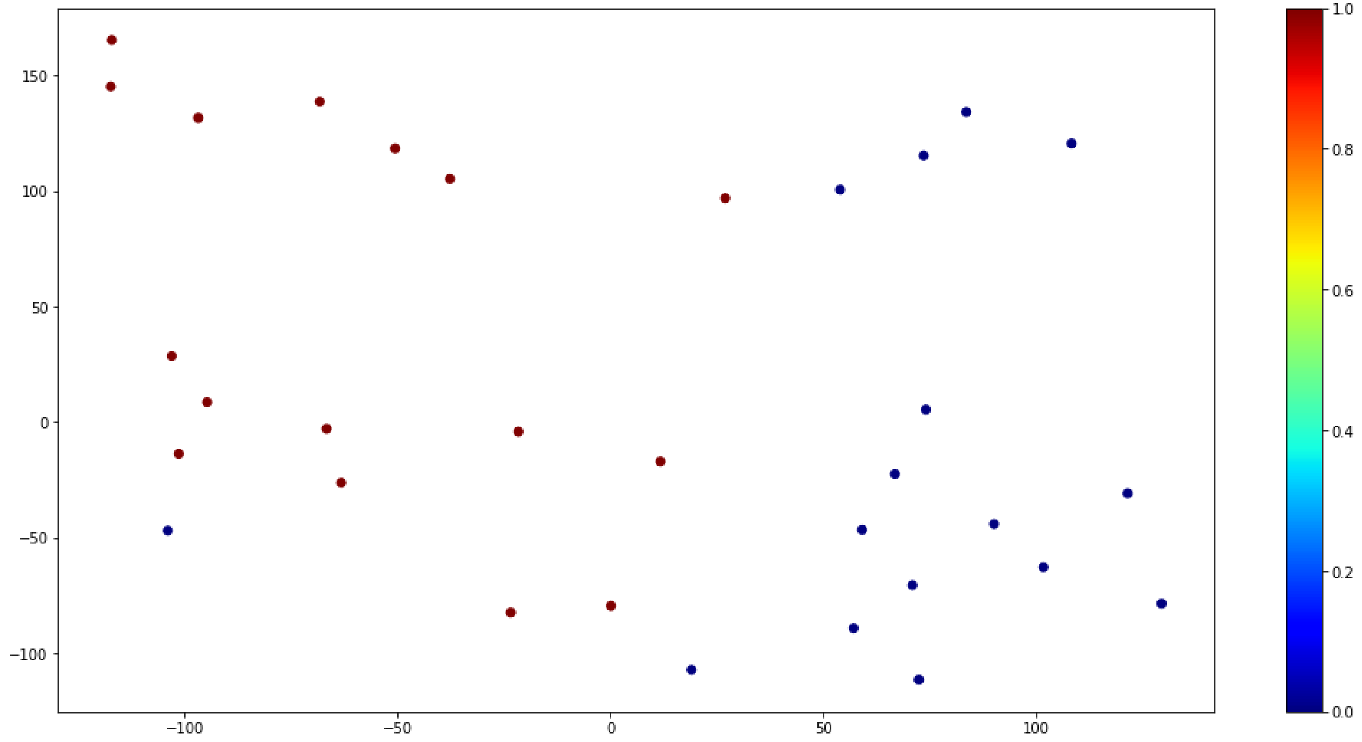
Normal = 16, Abnormal = 16, Latent vector size = 32, 1200 epochs



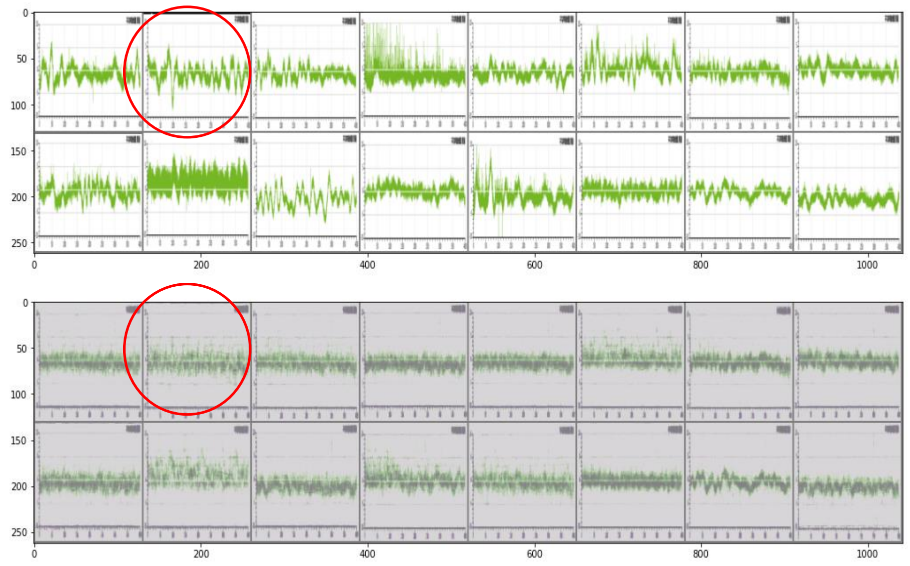
Recovered training images



t-SNE (perplexity=?) results of training images

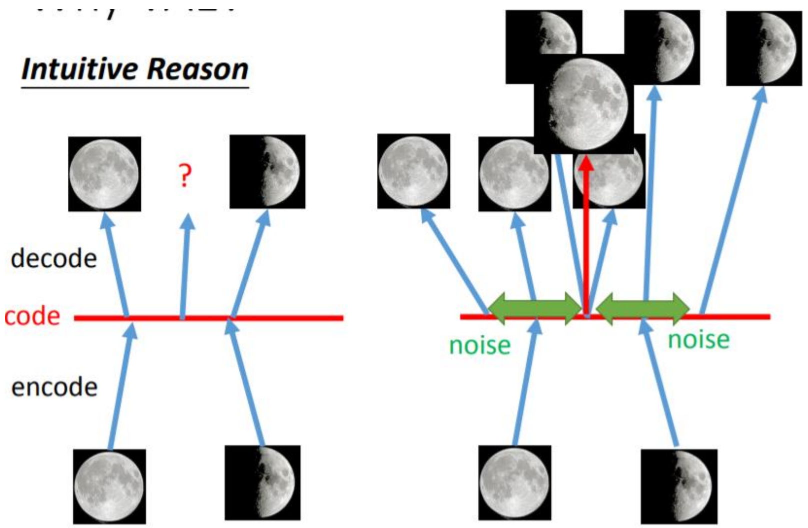


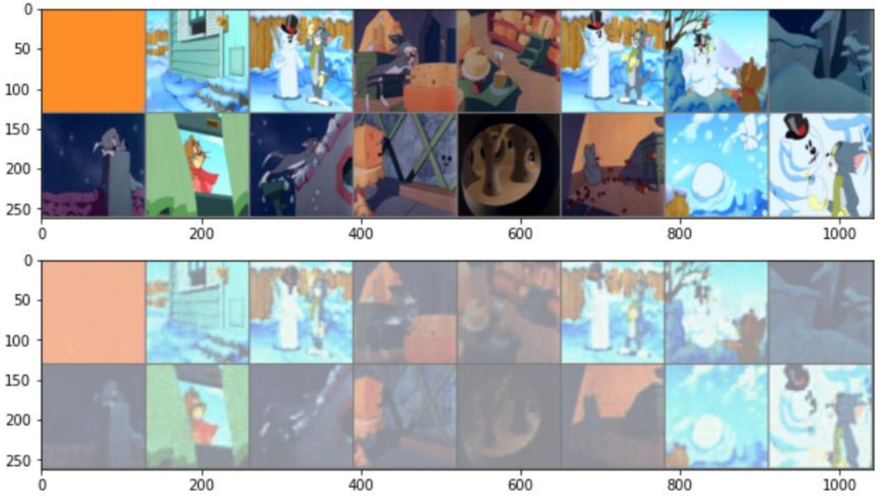
Recovered un-seen test images

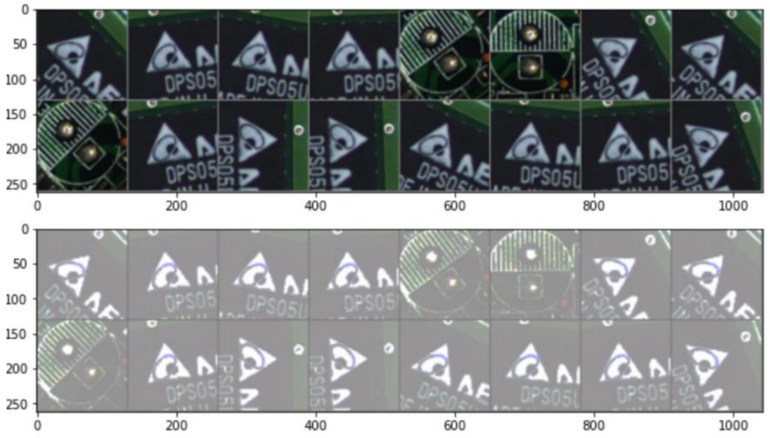


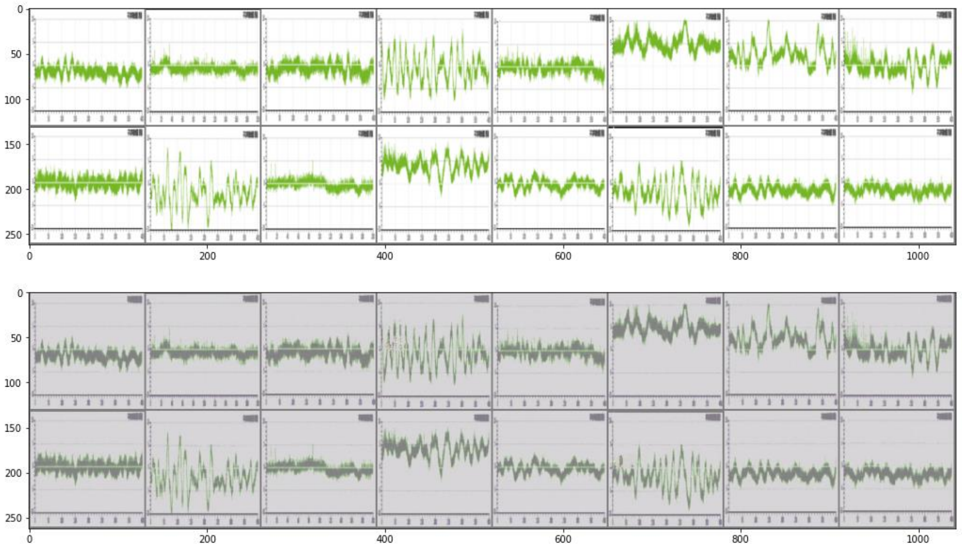
Problem of using decoder to generate images

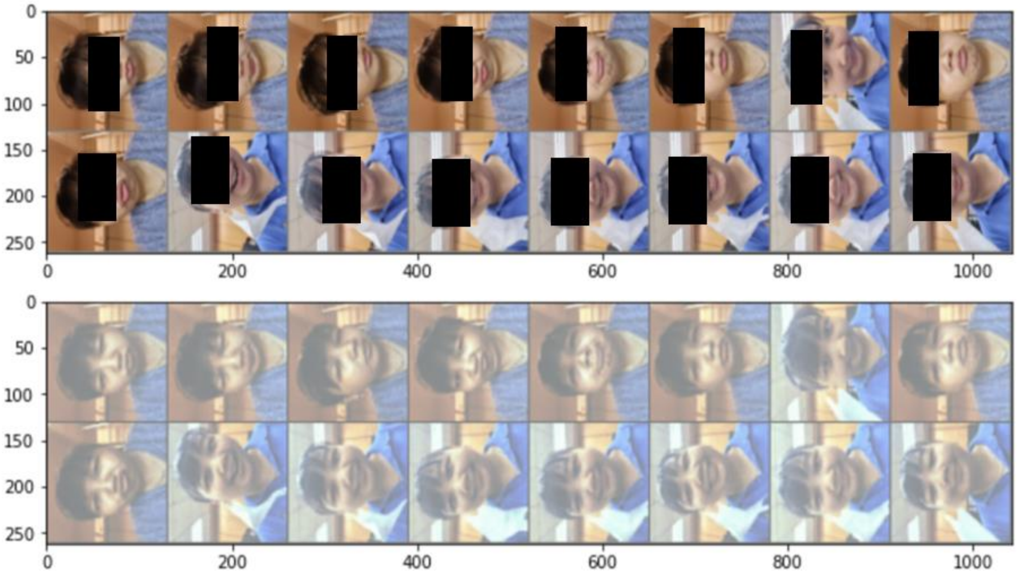
The images generated by AE/VAE are blur because they are the average of the input images



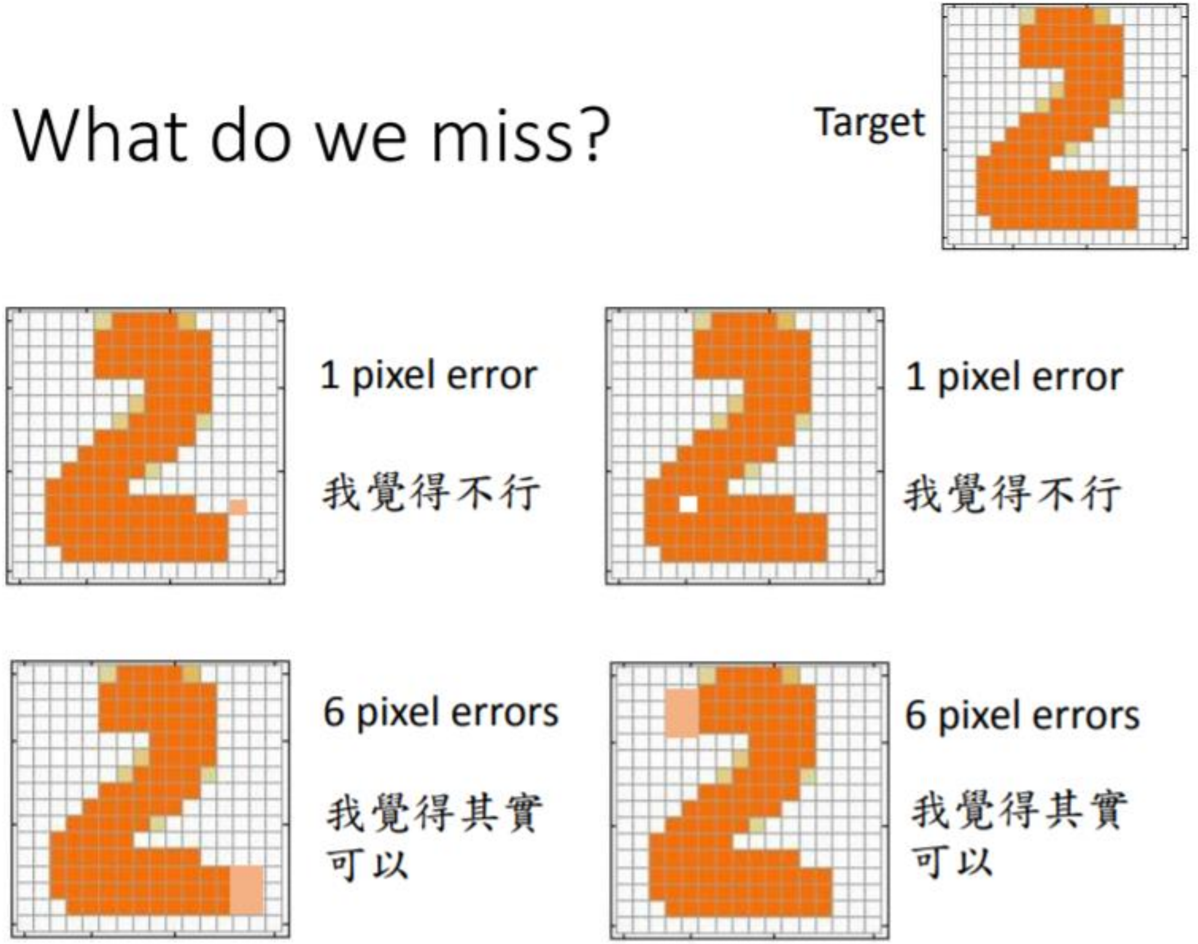






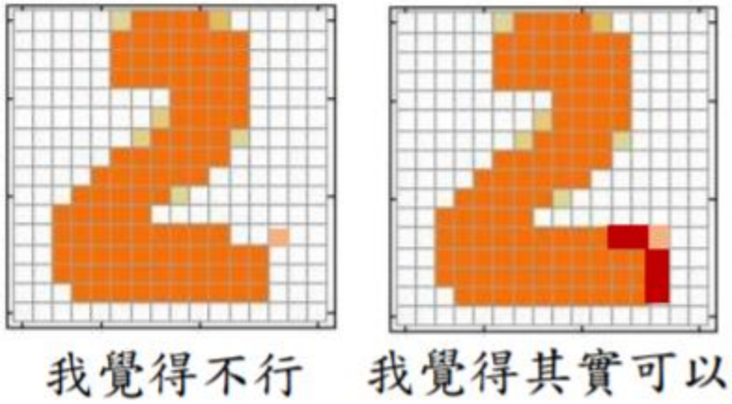


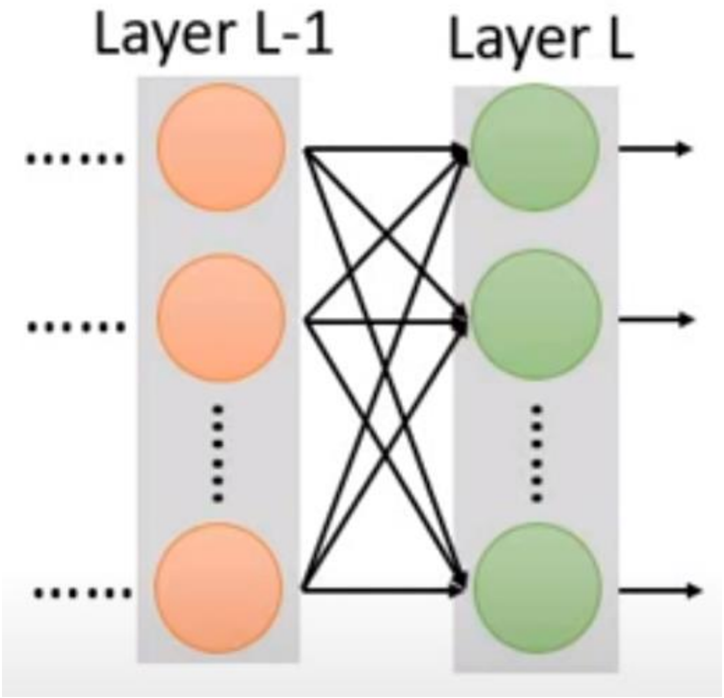
Decoder in AE or VAE generates images that minimize MSE + KL. It does not consider the context information



Reference: 李弘毅 GAN Lecture 1 (2018): https://youtu.be/DQNNMiAP5lw

The decoder does not consider the structured relationships between pixels





The relation between the components are critical.

Although highly correlated, they cannot influence each other.

Need deep structure to catch the relation between components.

Reference: 李弘毅 GAN Lecture 1 (2018): https://youtu.be/DQNNMiAP5lw

Advantages of GAN

- Still generate the object component-by-component
- But it is learned from the discriminator with global view.

Reference: 李弘毅 GAN Lecture 1 (2018): https://youtu.be/DQNNMiAP5lw