Autoencoder and GAN

Hands-On Machine Learning Part2

- Chapter 17 -

TAVE Research DL001

Heeji Won

Contents

01. Overview

02. Autoencoder

03. GAN

Contents

01. Overview

02. Autoencoder

03. GAN

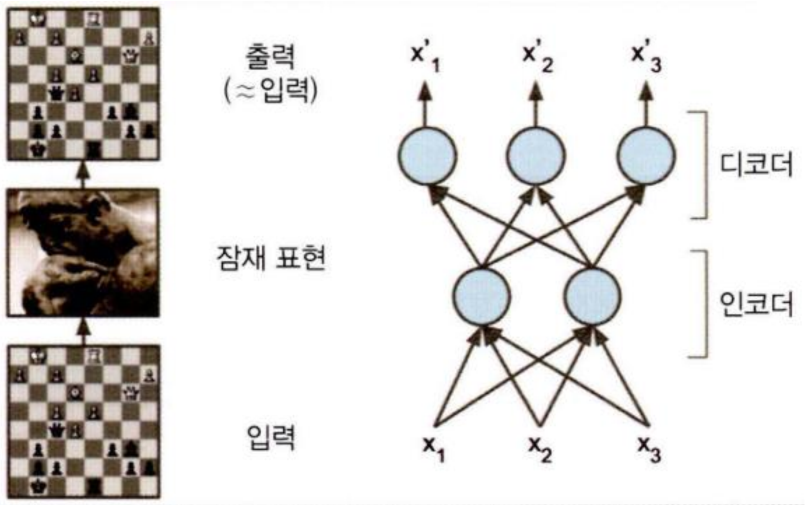
01. Overview

- **Autoencoder and GAN are unsupervised learning technique and learn useful representation**
- **But, they work differently**

▶ Autoencoder

- Learns to copy its input to its output
- are restricted in ways that force then to reconstruct the input approximately, preserving only the most relevant aspects of data
- A autoencoder consists of two parts, the encoder and the decoder

⇒ Get to know **how to represent data effectively**



▶ GAN

- Train by two networks (generative network, discriminative network) contesting with each other in a zero-sum game (**adversarial training**)

Contents

01. Overview

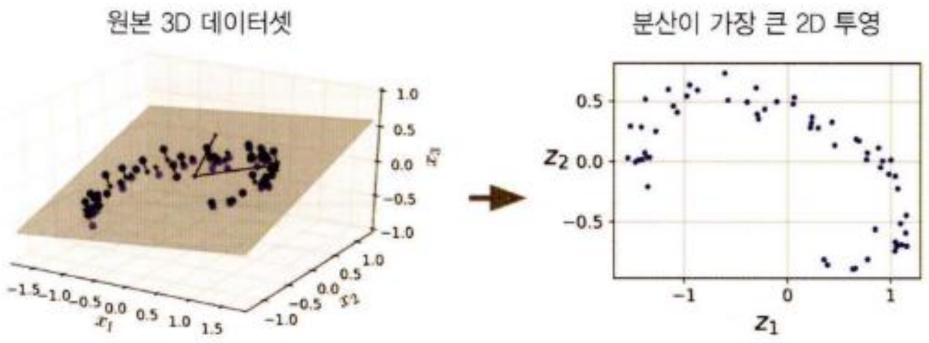
02. Autoencoder

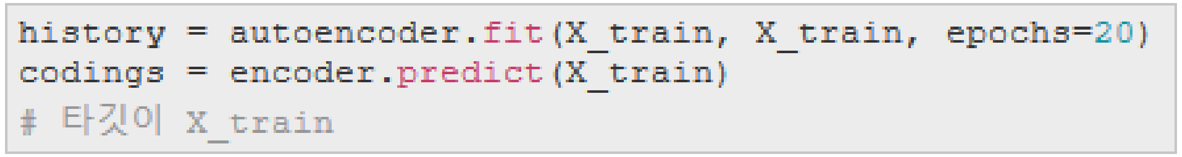
03. GAN

02. Autoencoder

➢ Undercomplete autoencoder

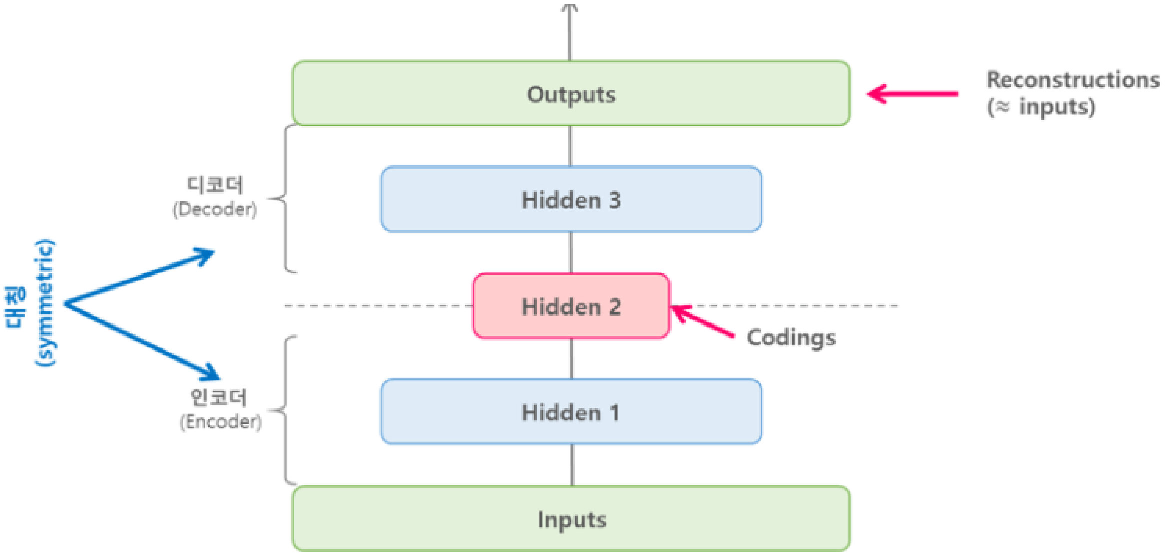
- Undercomplete means **the feature space have lower dimensionality than the input space**
- If autoencoder use only linear activation and MSE as cost function, it is the same as **PCA**





```
history = autoencoder.fit(X_train, X_train, epochs=20)
codings = encoder.predict(X_train)
# 타깃이 X_train
```

- **Stacked autoencoder**
  - A stacked autoencoder is a neural network consist several hidden layers
  - Can be used for pre-trained model



✓ Be careful not to be **Over-fitting**

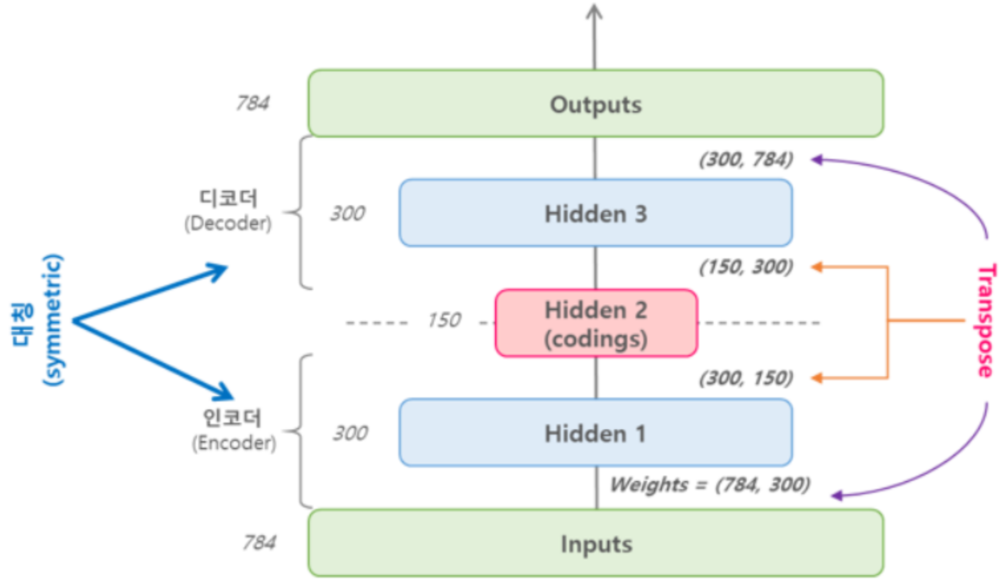
02. Autoencoder

➢ Undercomplete autoencoder

- Weight Tying

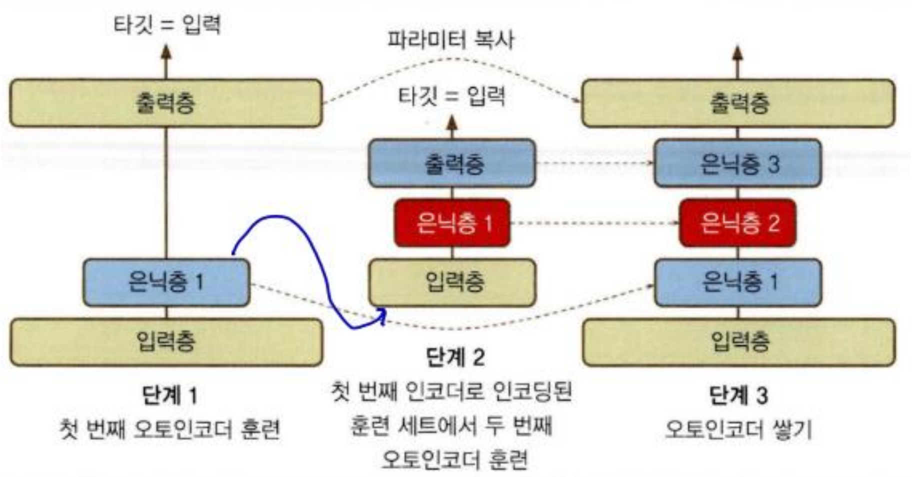
– When the weight matrix in the encoding layer is W, use W^T as the weight matrix in the decoding layer

$$\mathbf{W}_{N-L+1} = \mathbf{W}_L{}^T$$



- Greedy layer-wise training

(not used very well these days)



- Recurrent autoencoder

– Regard each row as a sequence

– Encoder is a Sequence-to-Vector RNN and decoder is a Vector-to-Sequence RNN

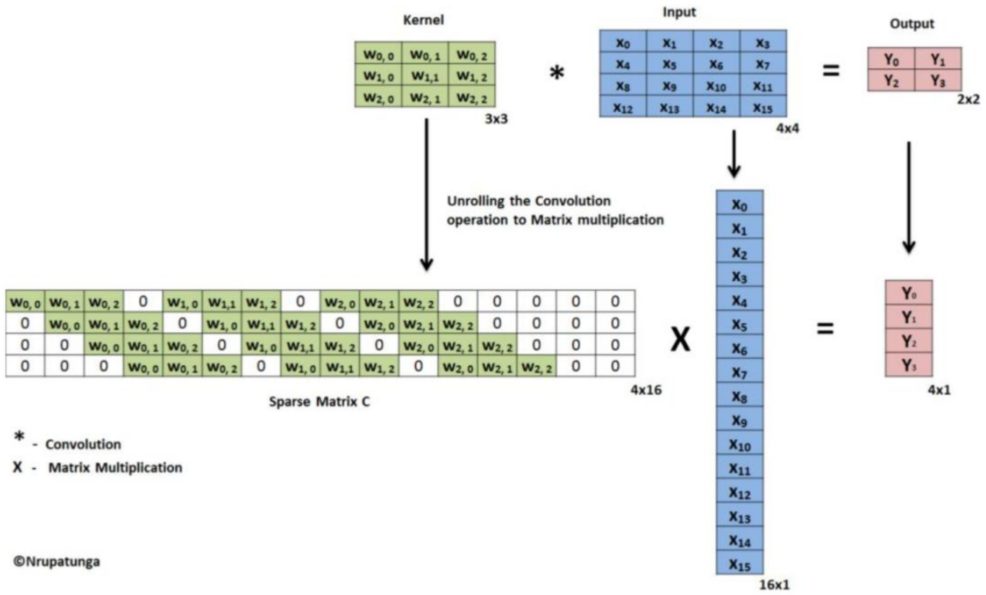
02. Autoencoder

➢ Undercomplete autoencoder

- Convolutional autoencoder

– Consists of convolution layers and pooling layers

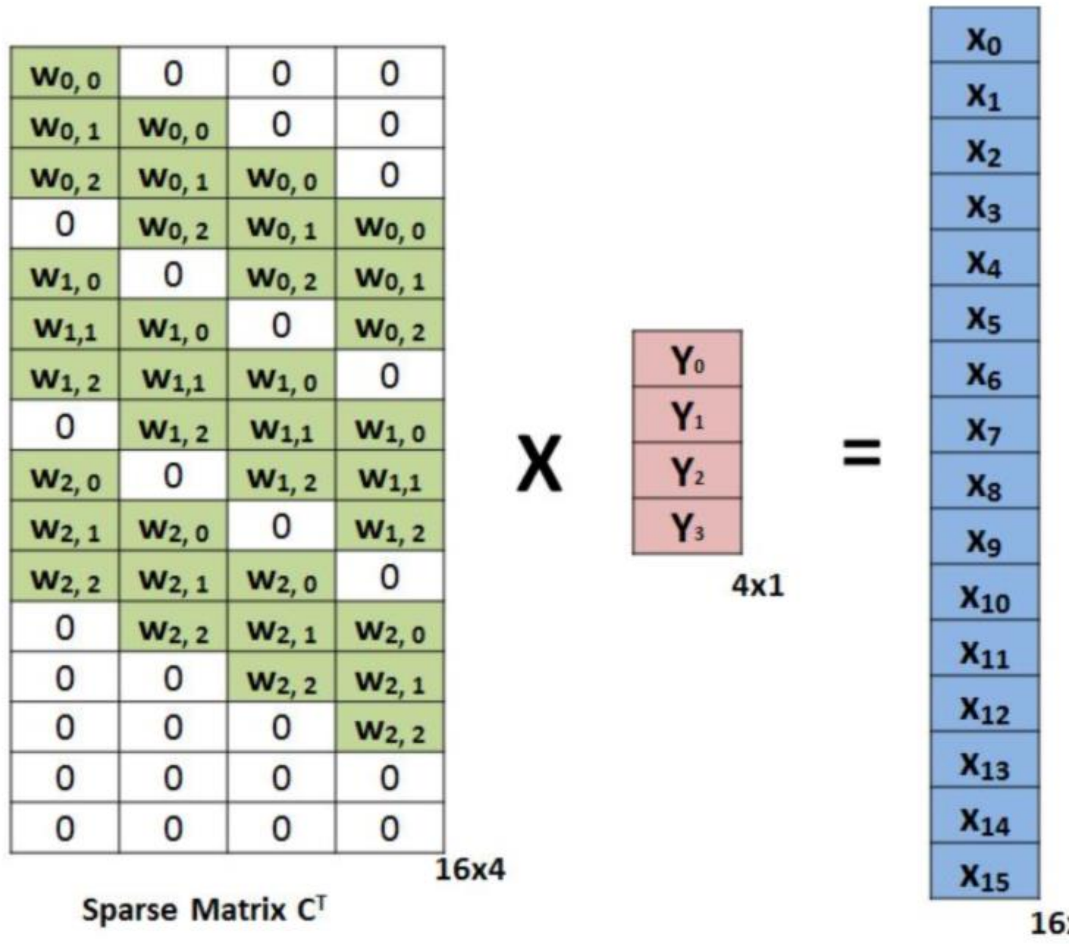
– Encoder reduce spatial-wise dimension (i.e. height and width) and increase depth (i.e. the number of feature maps)



Kernel
3x3
*
Input
4x4
=
Output
2x2
Unrolling the Convolution operation to Matrix multiplication
Sparse Matrix C
4x16
X
=
16x1
4x1
* - Convolution
X - Matrix Multiplication
©Nrupatunga

$$\begin{bmatrix} w_{0,0} & w_{0,1} & w_{0,2} \\ w_{1,0} & w_{1,1} & w_{1,2} \\ w_{2,0} & w_{2,1} & w_{2,2} \end{bmatrix} * \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} Y_0 & Y_1 \\ Y_2 & Y_3 \end{bmatrix}$$

$$\begin{bmatrix} w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 \end{bmatrix} \times \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{15} \end{bmatrix} = \begin{bmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \end{bmatrix}$$

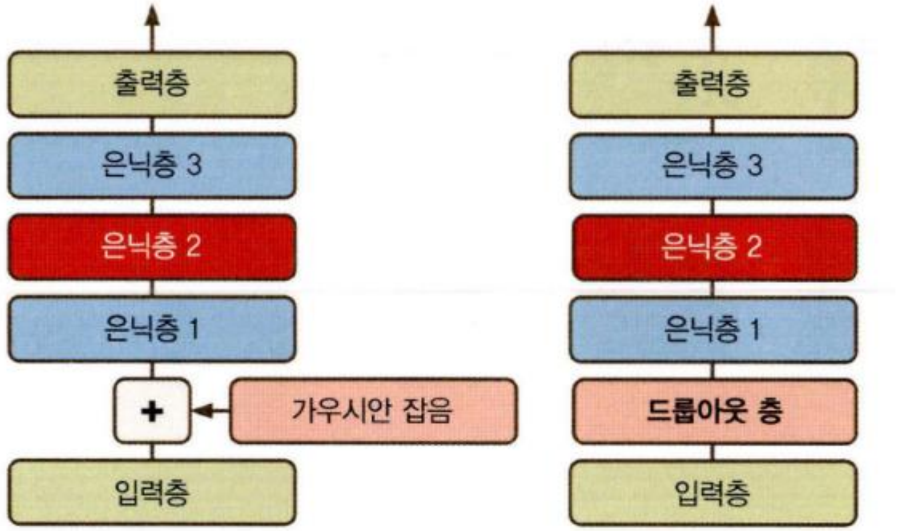


$$\begin{bmatrix} w_{0,0} & 0 & 0 & 0 \\ w_{0,1} & w_{0,0} & 0 & 0 \\ w_{0,2} & w_{0,1} & w_{0,0} & 0 \\ 0 & w_{0,2} & w_{0,1} & w_{0,0} \\ w_{1,0} & 0 & w_{0,2} & w_{0,1} \\ w_{1,1} & w_{1,0} & 0 & w_{0,2} \\ w_{1,2} & w_{1,1} & w_{1,0} & 0 \\ 0 & w_{1,2} & w_{1,1} & w_{1,0} \\ w_{2,0} & 0 & w_{1,2} & w_{1,1} \\ w_{2,1} & w_{2,0} & 0 & w_{1,2} \\ w_{2,2} & w_{2,1} & w_{2,0} & 0 \\ 0 & w_{2,2} & w_{2,1} & w_{2,0} \\ 0 & 0 & w_{2,2} & w_{2,1} \\ 0 & 0 & 0 & w_{2,2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \end{bmatrix} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{15} \end{bmatrix}$$

02. Autoencoder

➢ Overcomplete autoencoder

- Stacked denoising autoencoder



✓ For data visualization

✓ For pre-trained model

✓ For eliminating noise

- Sparsity Autoencoder

– Reduce the number of neurons which activates in coding layer (about 5% of neurons)

– Use sigmoid function as activation function in coding layer

- ***l*1** regularization $\mathcal{L}(x,\hat{x}) + \lambda \sum_i \left| a_i^{(h)} \right|$

- KL-Divergence $\mathcal{L}(x,\hat{x}) + \sum_j KL\left(\rho || \hat{\rho}_j\right)$

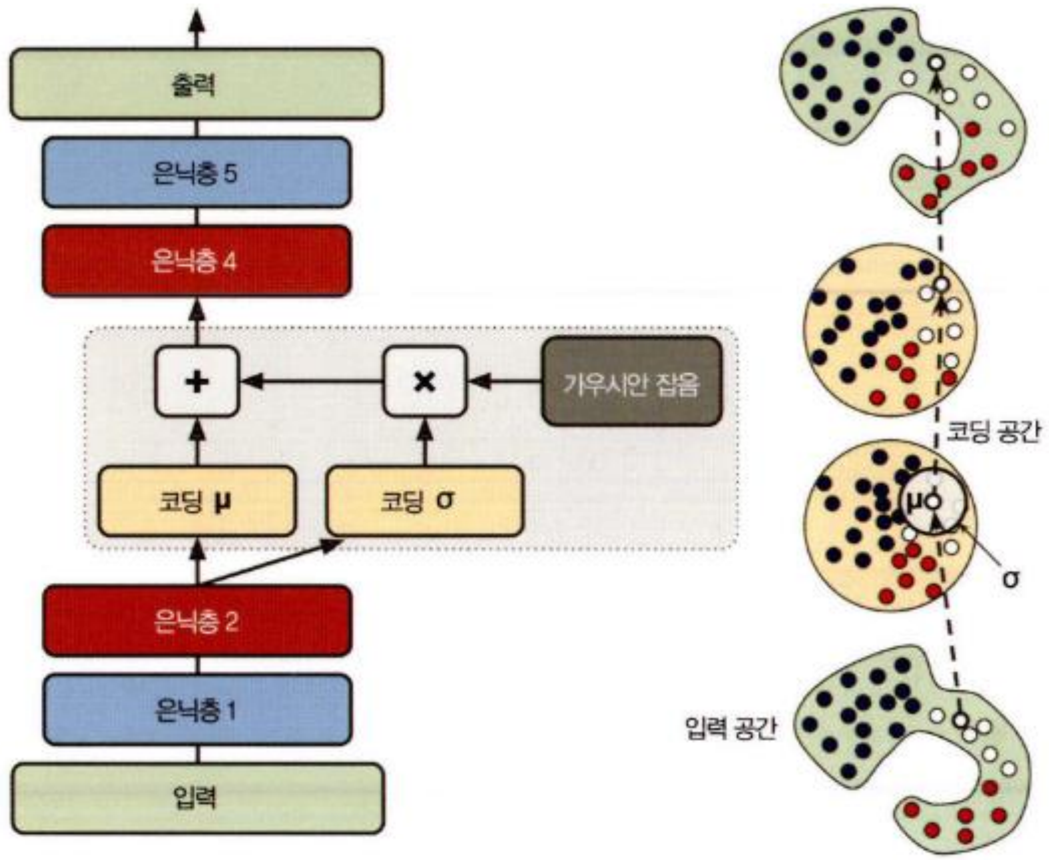
- ρ is a the average activation of a neuron over a collections of samples

$$D_{\mathrm{KL}}\left(p \,\|\, q\right) = p\log\frac{p}{q} + \left(1-p\right)\log\frac{1-p}{1-q}$$

02. Autoencoder

➢ Variational autoencoder (VAE)

- **Generative model** (different from AE)
- Encoder calculate mean coding μ and standard deviation coding σ
- And then, select random sample from gaussian distribution with mean μ and variance σ
- Cost function consists of reconstruction loss and **latent loss** (distance of distribution before sampling and distribution after sampling)

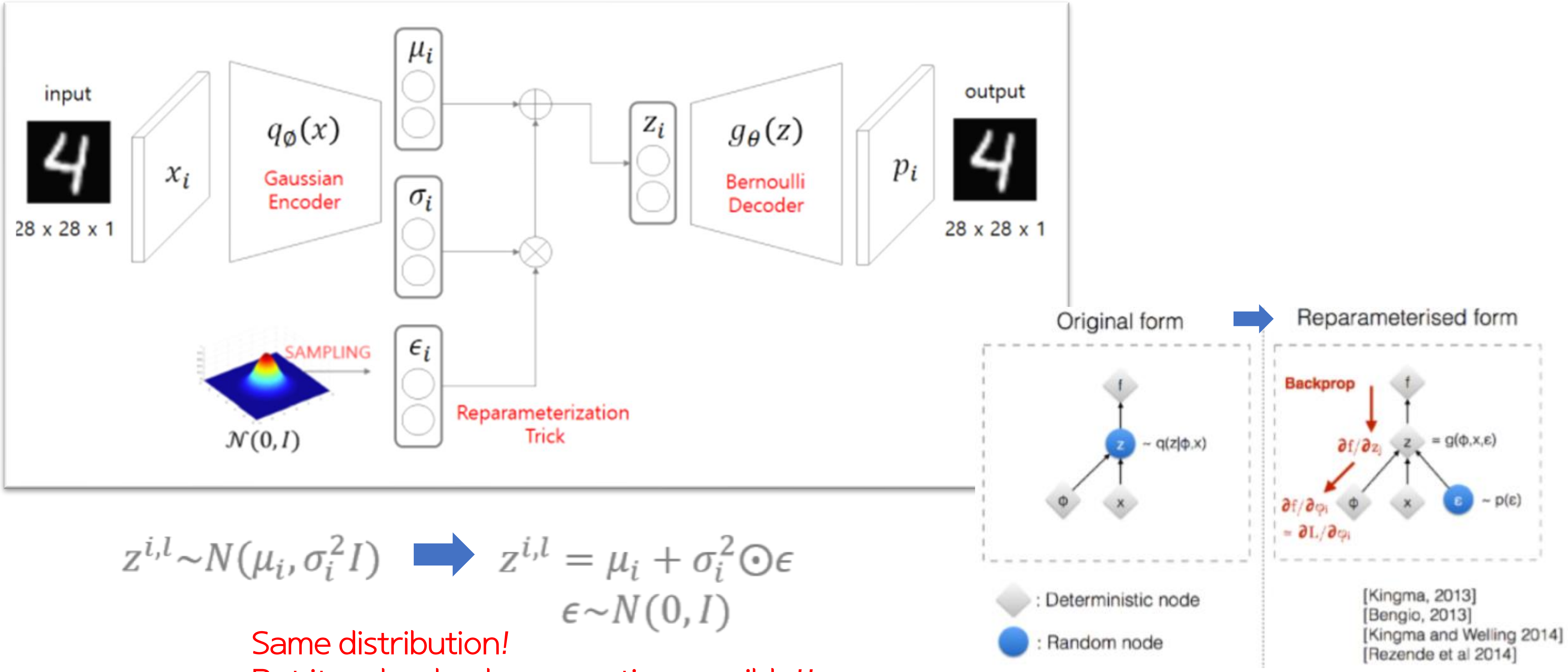


✓ Why don' t we use MLE?

If $p(x|g_\theta(z)\ ) = \mathcal{N}(x|g_\theta(z), \sigma^2 * I)$, the negative log probability of X is proportional squared Euclidean distance between $g_\theta(z)$ and x.

02. Autoencoder

➢ Variational autoencoder (VAE)



[Kingma, 2013]
[Bengio, 2013]
[Kingma and Welling 2014]
[Rezende et al 2014]

$$z^{i,l} \sim N(\mu_i, \sigma_i^2 I) \Rightarrow z^{i,l} = \mu_i + \sigma_i^2 \odot \epsilon$$
$$\epsilon \sim N(0, I)$$

Same distribution!
But it makes backpropagation possible!!

02. Autoencoder

➢ Variational autoencoder (VAE)

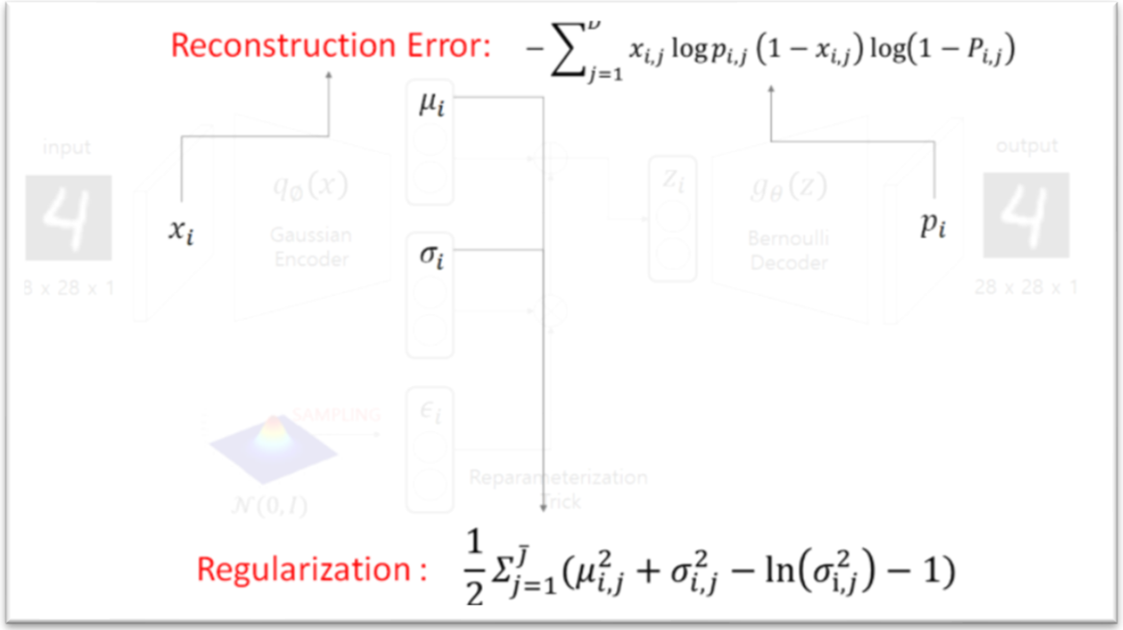
- ELBO $$\log p\left(x\right) \geq E_{z\sim q\left(z|x\right)}\left[\log p(x|z)\right] - D_{KL}\left(q\left(z|x\right)||p\left(z\right)\right) = ELBO$$

Reconstruction term

- 이상적인 샘플링 함수로부터 얼마나 잘 복원을 했는가

Regularlization term

- 이상적인 sampling함수가 최대한 prior과 같도록 만들어준다
- 여러 sample중에서 prior과 유사한 값을 samplin하도록 condition 부여



Reconstruction Error: $-\sum_{j=1}^{\nu} x_{i,j} \log p_{i,j} \left(1 - x_{i,j}\right) \log\left(1 - P_{i,j}\right)$

Regularization : $\frac{1}{2}\Sigma_{j=1}^{J}\left(\mu_{i,j}^{2} + \sigma_{i,j}^{2} - \ln\left(\sigma_{\mathrm{i},j}^{2}\right) - 1\right)$

$$D_{KL}\left(q\left(z|x\right)||p\left(z\right)\right) = D_{KL}\left[N\left(\left(\mu_1, \ldots, \mu_k\right)^T, \operatorname{diag}\left(\sigma_1^2, \ldots, \sigma_k^2\right)\right)||N\left(0, 1\right)\right]$$
$$= \frac{1}{2}\sum_{i=1}\left(\sigma_i^2 + \mu_i^2 - \ln\left(\sigma_i^2\right) - 1\right)$$

Contents

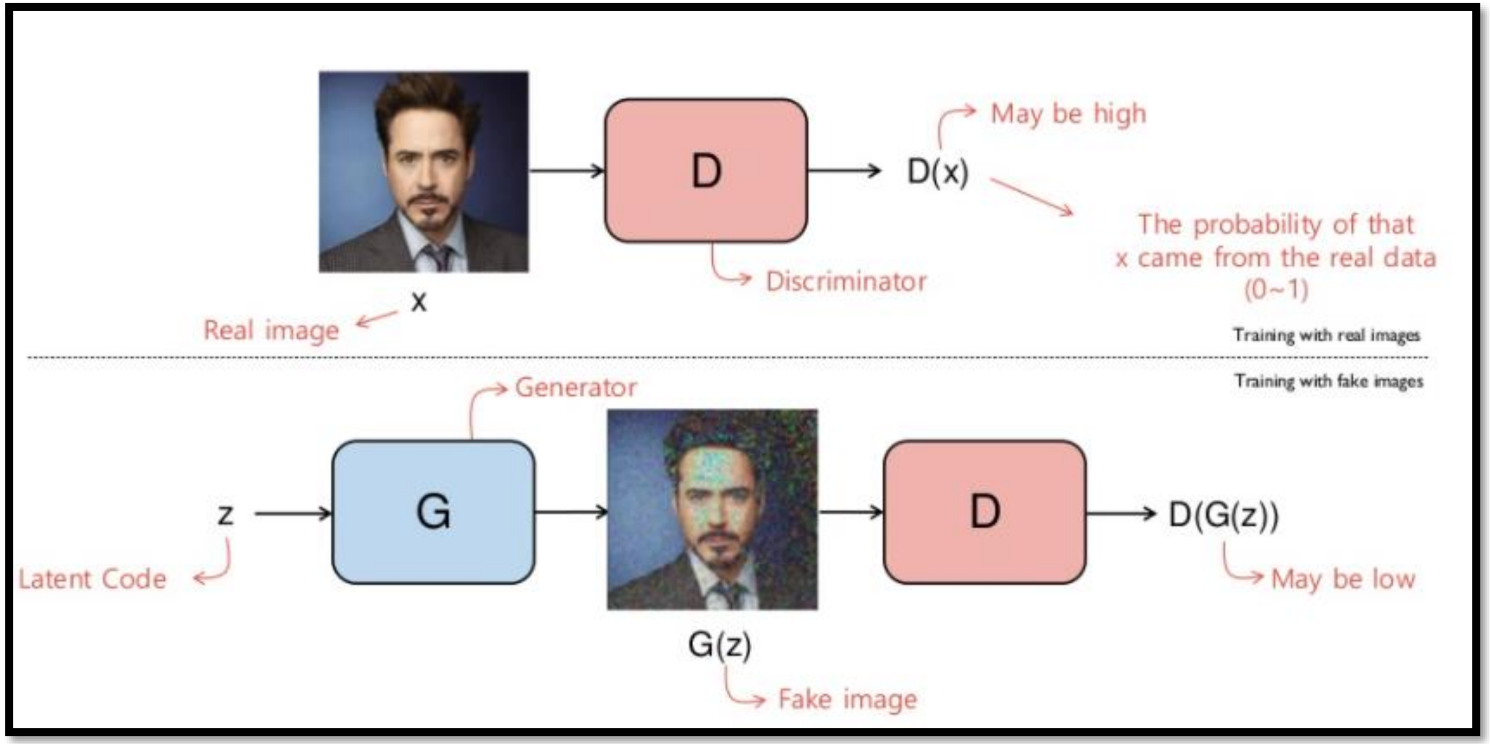
01. Overview

02. Autoencoder

03. GAN

03. GAN

- Training by two networks (generative network, discriminative network) contesting with each other in a zero-sum game (**adversarial training**)
- Process

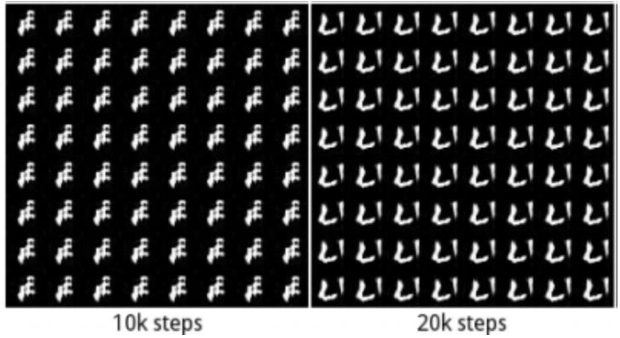


D
D(x)
May be high
The probability of that x came from the real data (0~1)
Discriminator
x
Real image
Training with real images
Training with fake images
Generator
z
Latent Code
G
D
D(G(z))
May be low
G(z)
Fake image

Toward Nash equilibrium...

Difficulty

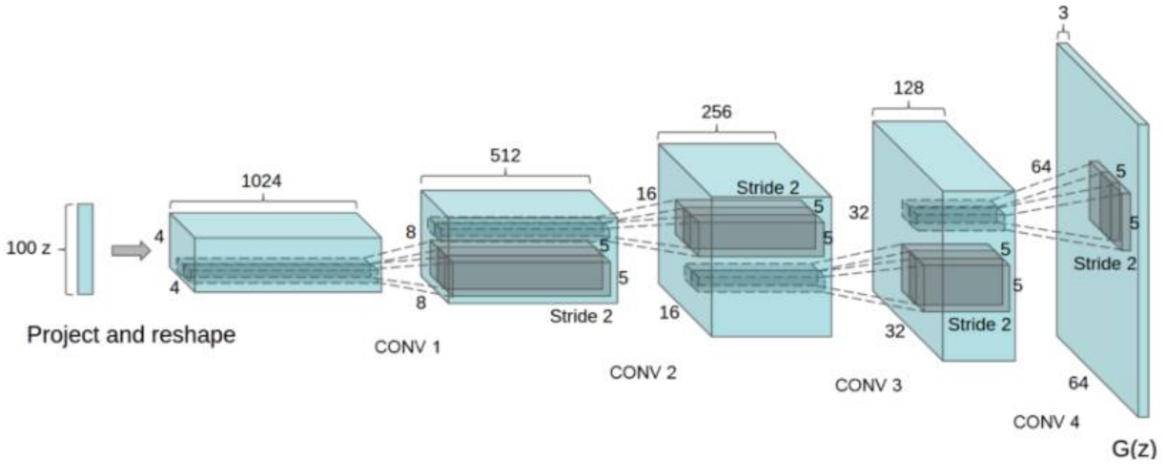
✓ Unstable parameters
✓ Mode collapse when reduced diversity of the generator' s output



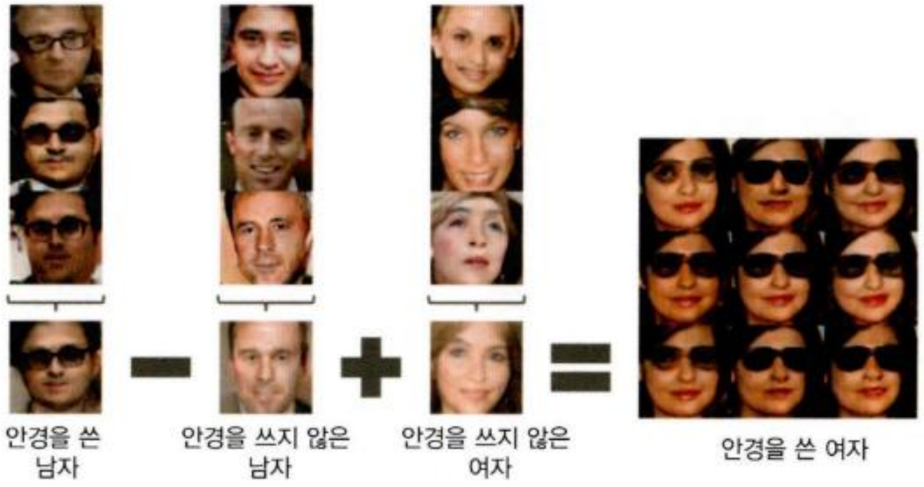
Mode collapse

03. GAN

➢ Deep convolutional GAN (DCGAN)



- Replace all max pooling with convolutional stride
- Use transposed convolution for upsampling.
- Eliminate fully connected layers.
- Use Batch normalization except the output layer for the generator and the input layer of the discriminator.
- Use ReLU in the generator except for the output which uses tanh.
- Use LeakyReLU in the discriminator.

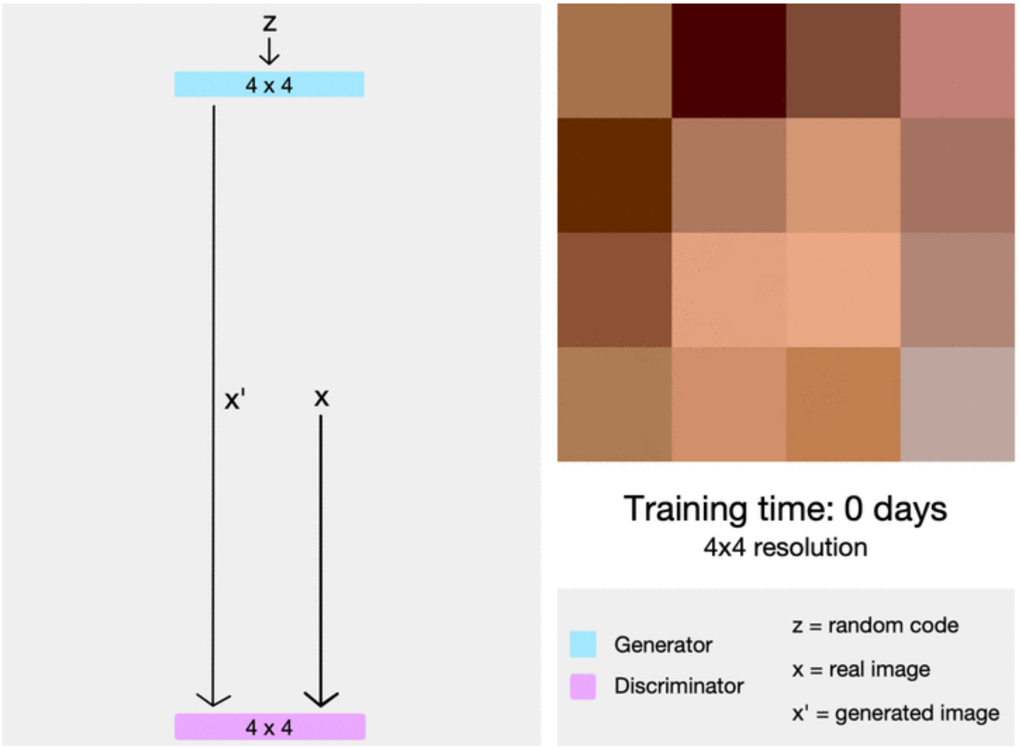


latent vector arithmetic

03. GAN

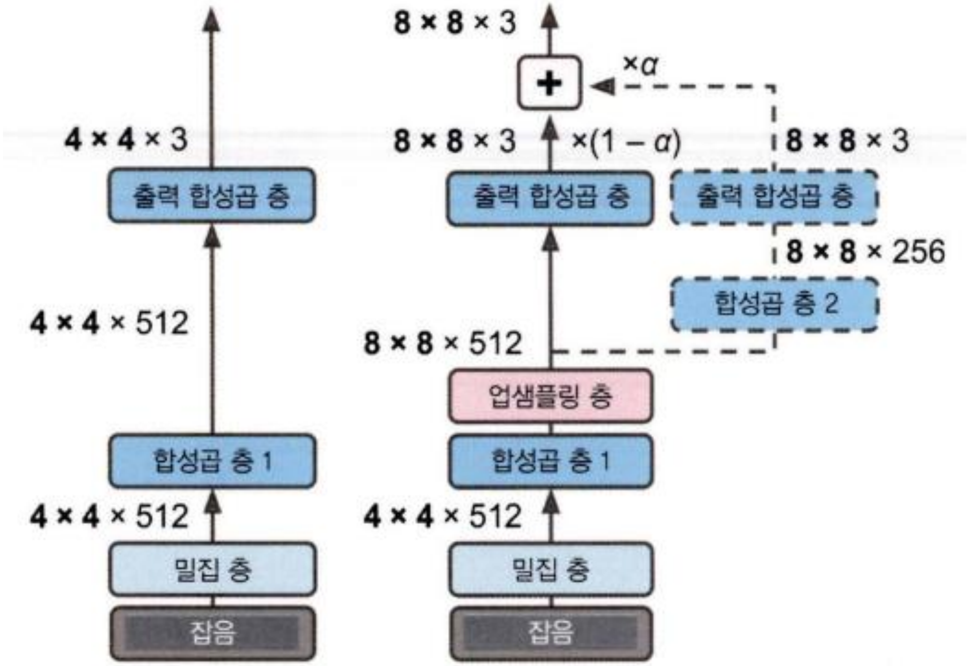
➢ Progressive growing GAN (ProGAN, PGGAN)

- Growing both the generator and discriminator progressively
- This both speeds the training up and greatly stabilizes it.



- Fade in

- fade in the new layers smoothly
- α is gradually increase from 0 to 1



03. GAN

➢ Progressive growing GAN (ProGAN, PGGAN)

- Increasing Variation using MiniBatch Standard Deviation

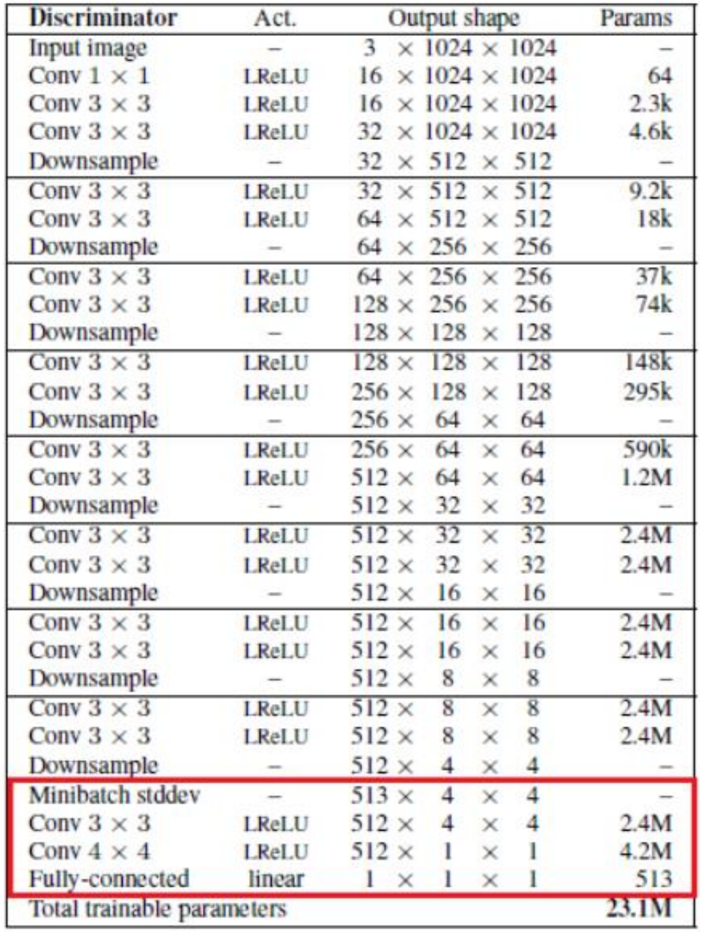
– Compute the standard deviation of each feature per spatial location ($\mathrm{N} \times C \times H \times W \rightarrow C \times H \times W$)

– And then, average these values to one value per spatial location ($C \times H \times W \rightarrow 1 \times H \times W$)

- Equalized learning rate $\hat{w}_i = \frac{w_i}{c}$

- PixelWise Feature Vector Normalization

$$b_{x,y} = \frac{a_{x,y}}{\sqrt{\frac{1}{N}\sum_{j=0}^{N-1}(a_{x,y}^{j})^2 + \epsilon}}$$

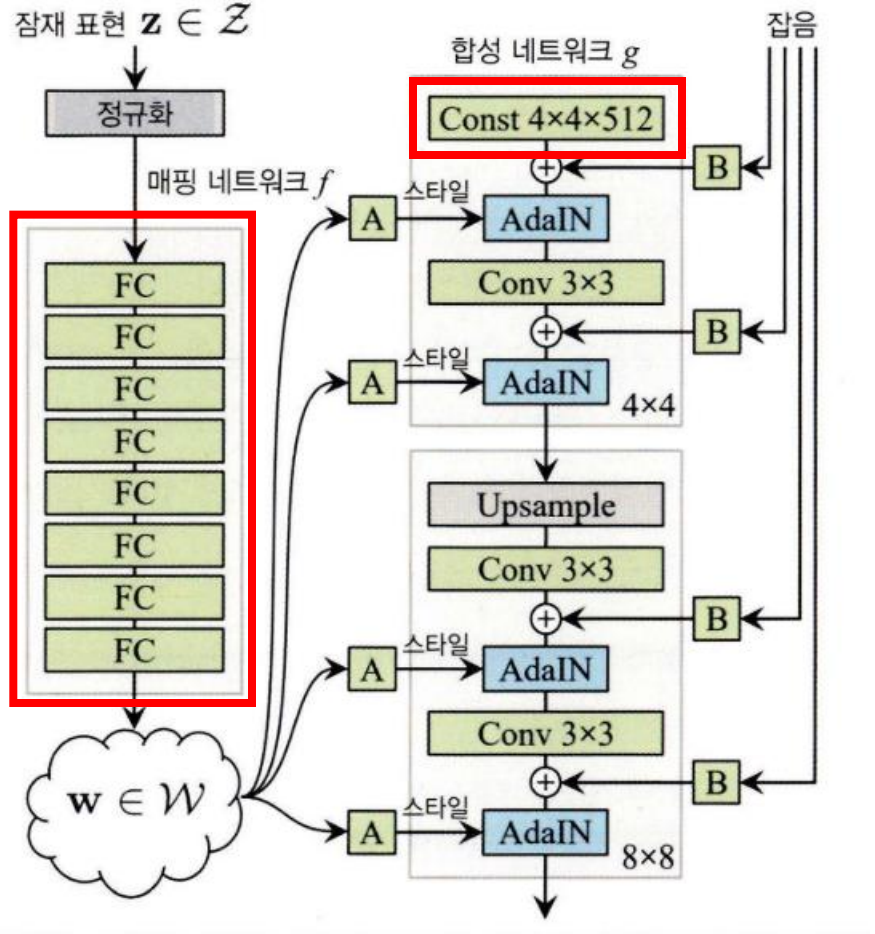


Discriminator	Act.	Output shape	Params
Input image	–	3 × 1024 × 1024	–
Conv 1 × 1	LReLU	16 × 1024 × 1024	64
Conv 3 × 3	LReLU	16 × 1024 × 1024	2.3k
Conv 3 × 3	LReLU	32 × 1024 × 1024	4.6k
Downsample	–	32 × 512 × 512	–
Conv 3 × 3	LReLU	32 × 512 × 512	9.2k
Conv 3 × 3	LReLU	64 × 512 × 512	18k
Downsample	–	64 × 256 × 256	–
Conv 3 × 3	LReLU	64 × 256 × 256	37k
Conv 3 × 3	LReLU	128 × 256 × 256	74k
Downsample	–	128 × 128 × 128	–
Conv 3 × 3	LReLU	128 × 128 × 128	148k
Conv 3 × 3	LReLU	256 × 128 × 128	295k
Downsample	–	256 × 64 × 64	–
Conv 3 × 3	LReLU	256 × 64 × 64	590k
Conv 3 × 3	LReLU	512 × 64 × 64	1.2M
Downsample	–	512 × 32 × 32	–
Conv 3 × 3	LReLU	512 × 32 × 32	2.4M
Conv 3 × 3	LReLU	512 × 32 × 32	2.4M
Downsample	–	512 × 16 × 16	–
Conv 3 × 3	LReLU	512 × 16 × 16	2.4M
Conv 3 × 3	LReLU	512 × 16 × 16	2.4M
Downsample	–	512 × 8 × 8	–
Conv 3 × 3	LReLU	512 × 8 × 8	2.4M
Conv 3 × 3	LReLU	512 × 8 × 8	2.4M
Downsample	–	512 × 4 × 4	–
Minibatch stddev	–	513 × 4 × 4	–
Conv 3 × 3	LReLU	512 × 4 × 4	2.4M
Conv 4 × 4	LReLU	512 × 1 × 1	4.2M
Fully-connected	linear	1 × 1 × 1	513
Total trainable parameters			23.1M

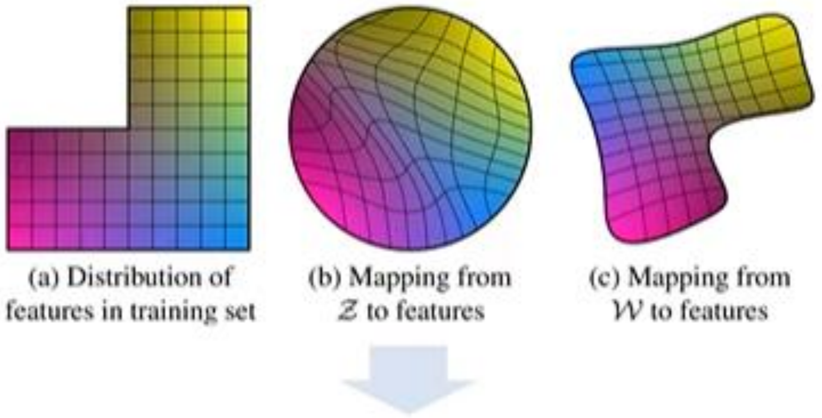
03. GAN

➢ StyleGAN

: A novel GAN using **style transfer method**



- Mapping network



(a) Distribution of features in training set

(b) Mapping from 𝒵 to features

(c) Mapping from 𝒲 to features

ı W space, the factors of variation become more linear.

Z: Fixed distribution

Learned mapping $f: z \rightarrow w$

- Constant input

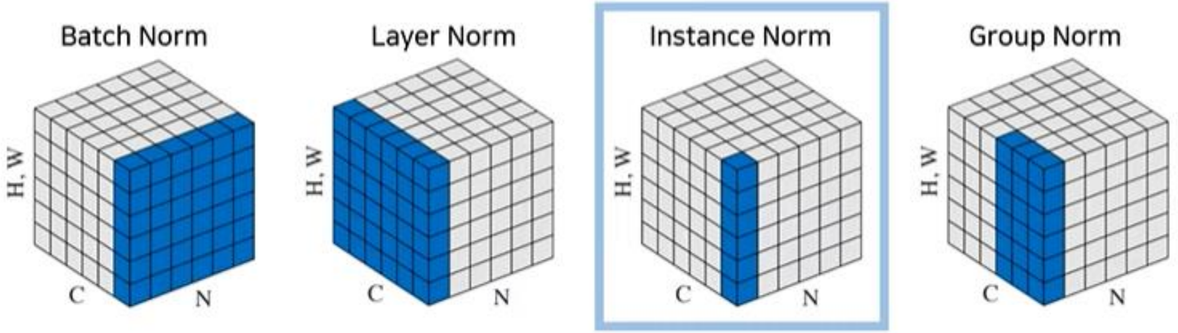
– Use constant as input of synthesis Network

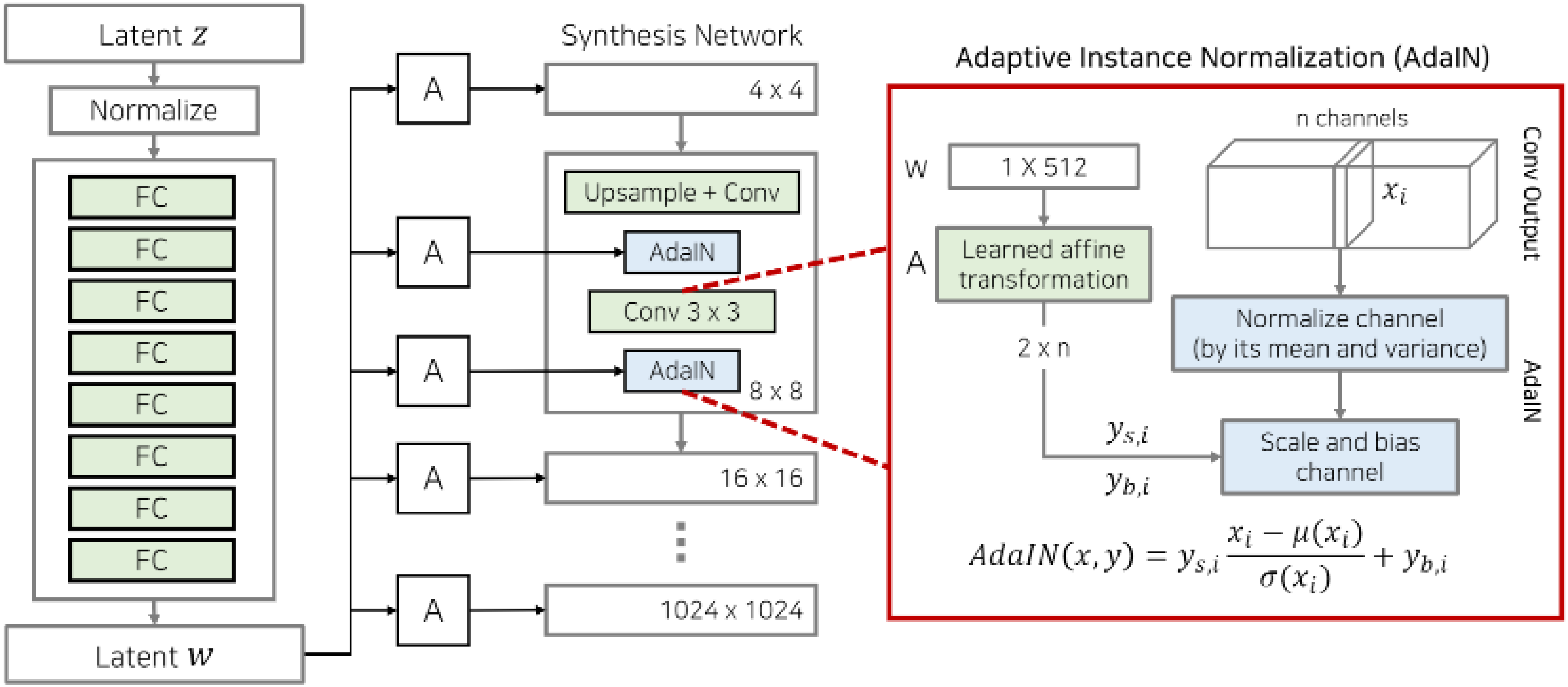
– Increase performance empirically

03. GAN

➢ StyleGAN

- AdaIN – Normalize each feature map





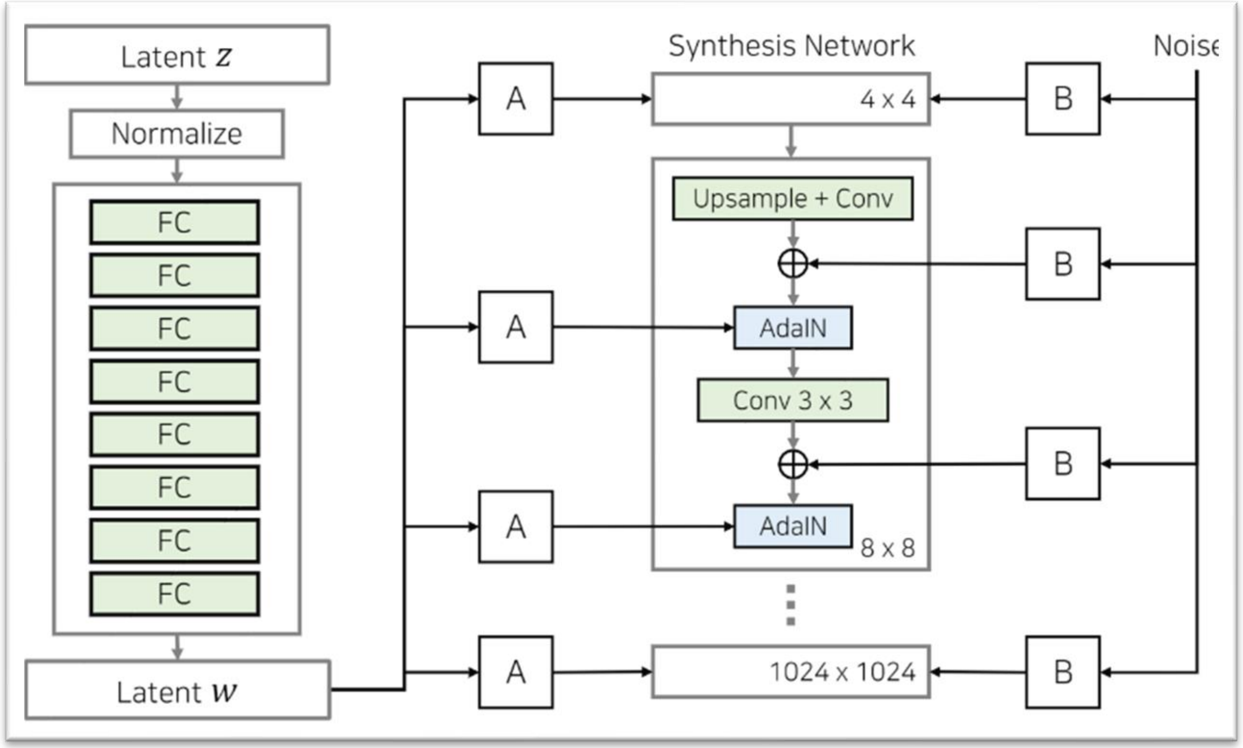
$$AdaIN(x, y) = y_{s,i}\frac{x_i - \mu(x_i)}{\sigma(x_i)} + y_{b,i}$$

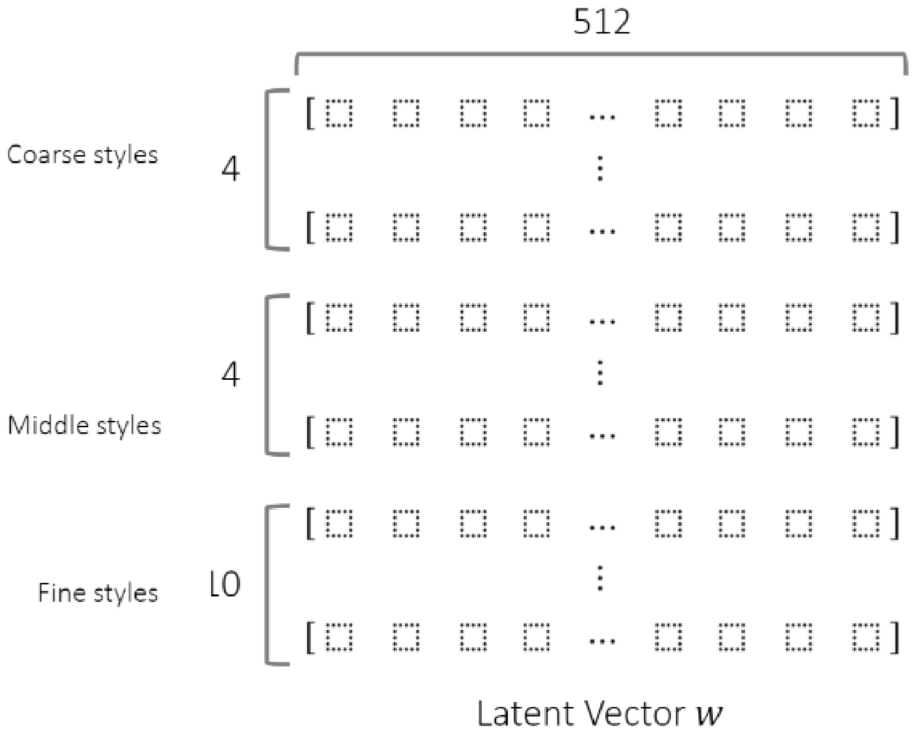
03. GAN

➢ StyleGAN

- Stochastic Variation

– Control stochastic variation like freckles and hair arrangement





03. GAN

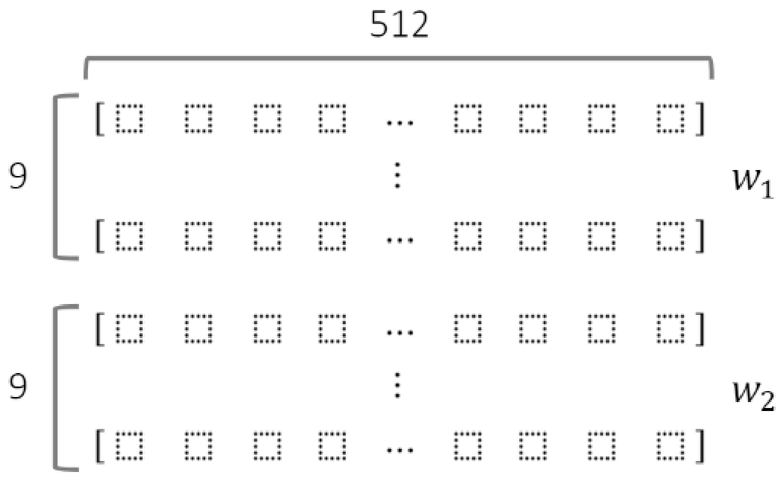
➢ StyleGAN

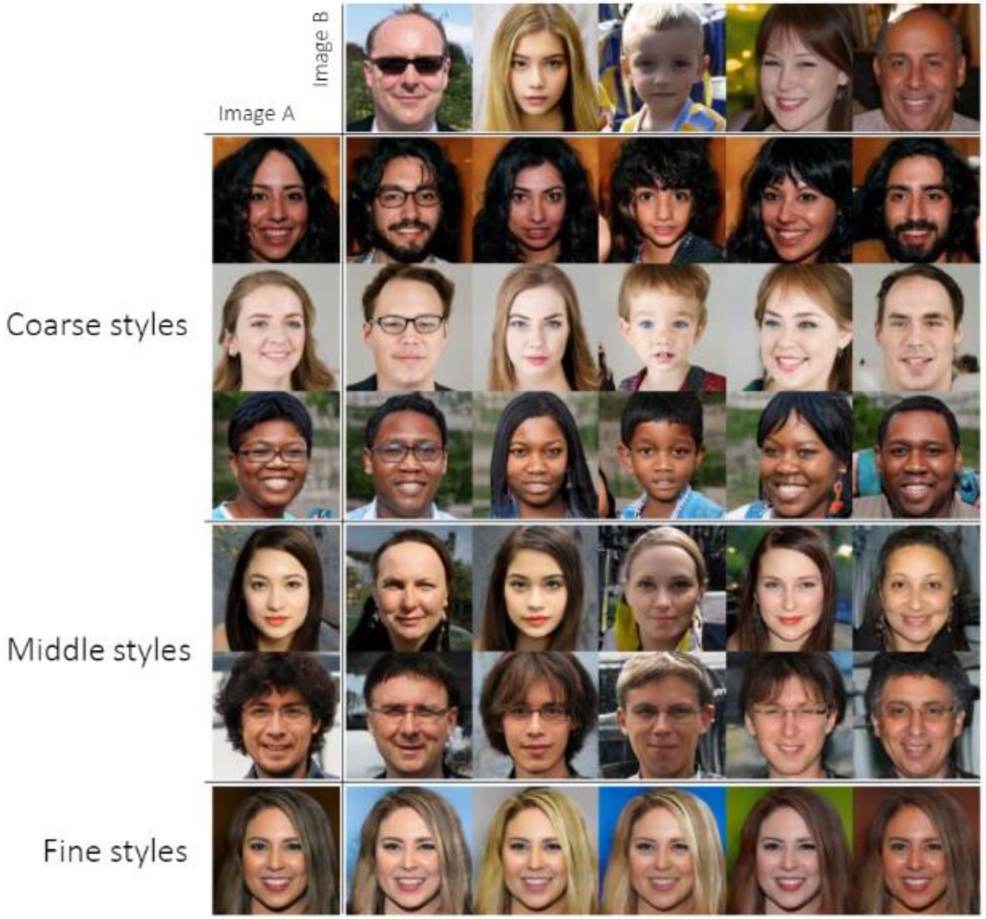
- Style Mixing (Mixing Regularization)

– To reduce the correlation between adjacent layers

– With two input vectors

– Train some of the levels with the first and switches (in a random point) to the other to train the rest of the levels





Thank you