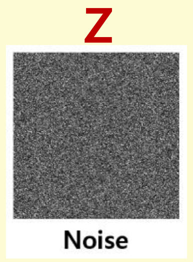
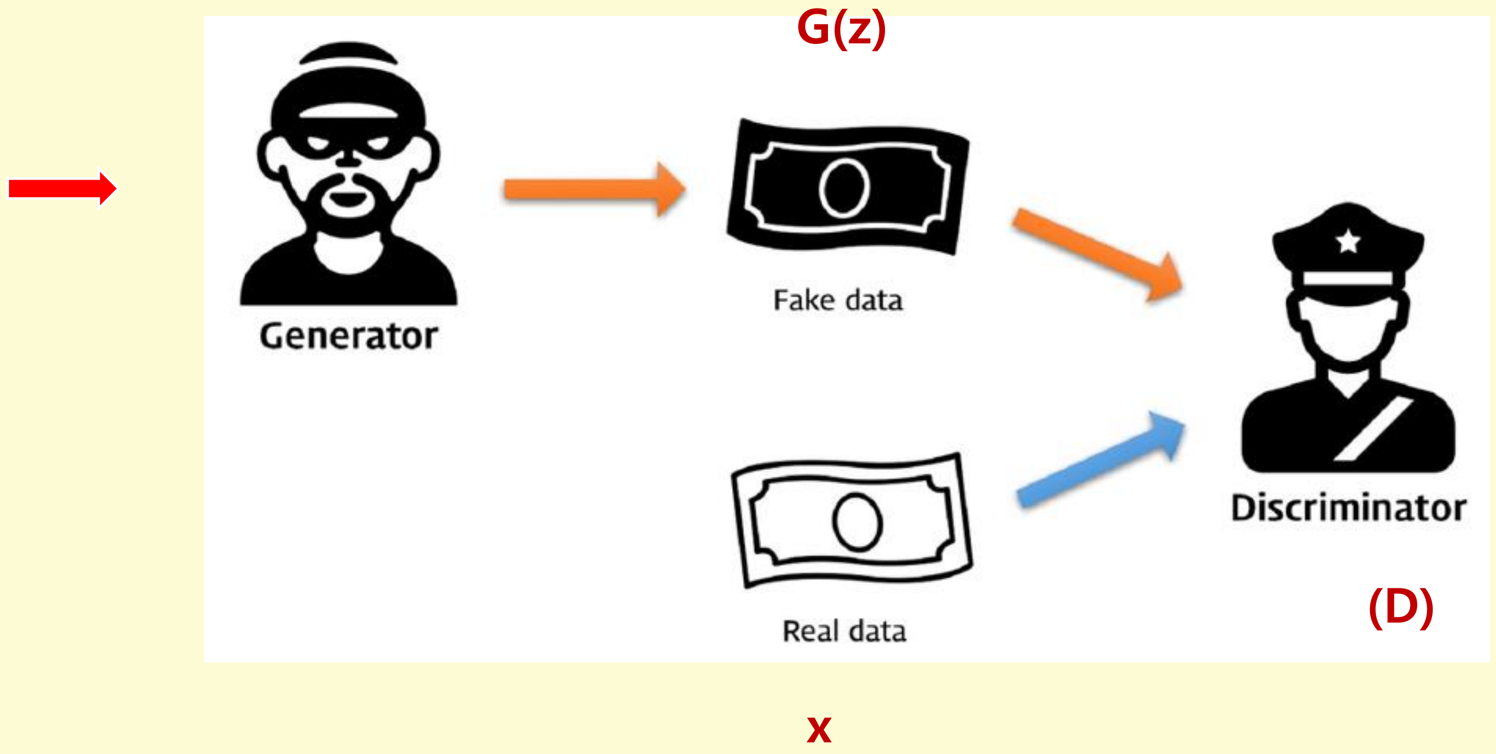
Analyzing and Improving the Image Quality of StyleGAN

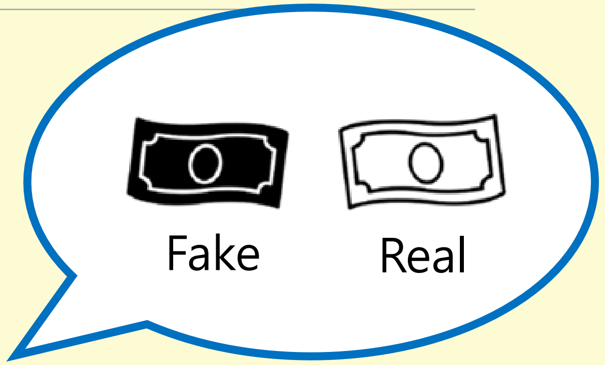
Team 연어유희

목차

GAN (Generative Adversarial Network)





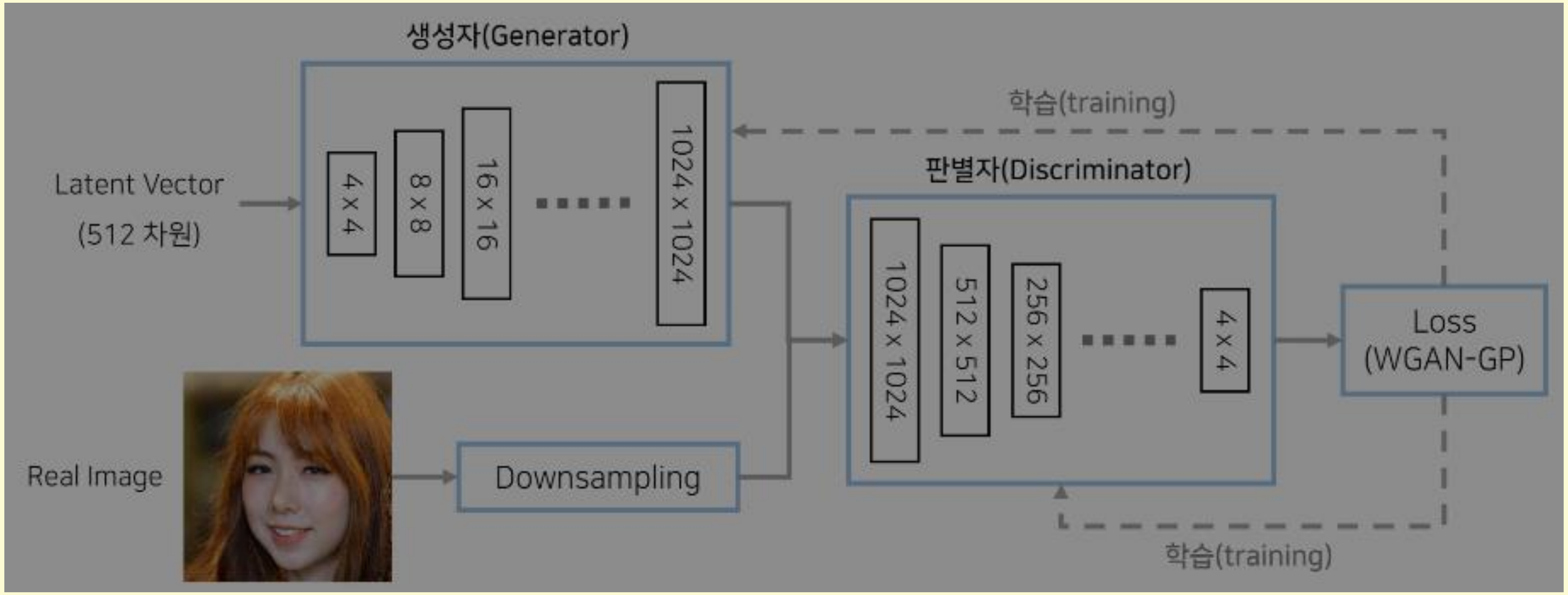


$D(G(z))$: 0
$D(x)$: 1

① 생성자(Generator), 판별자(Discriminator) 네트워크를 활용한 생성 모델

② 생성자는 판별자를 활용하여 이미지 분포를 학습하고 이미지를 생성함

PGGAN (Progressive Growing of GANs)

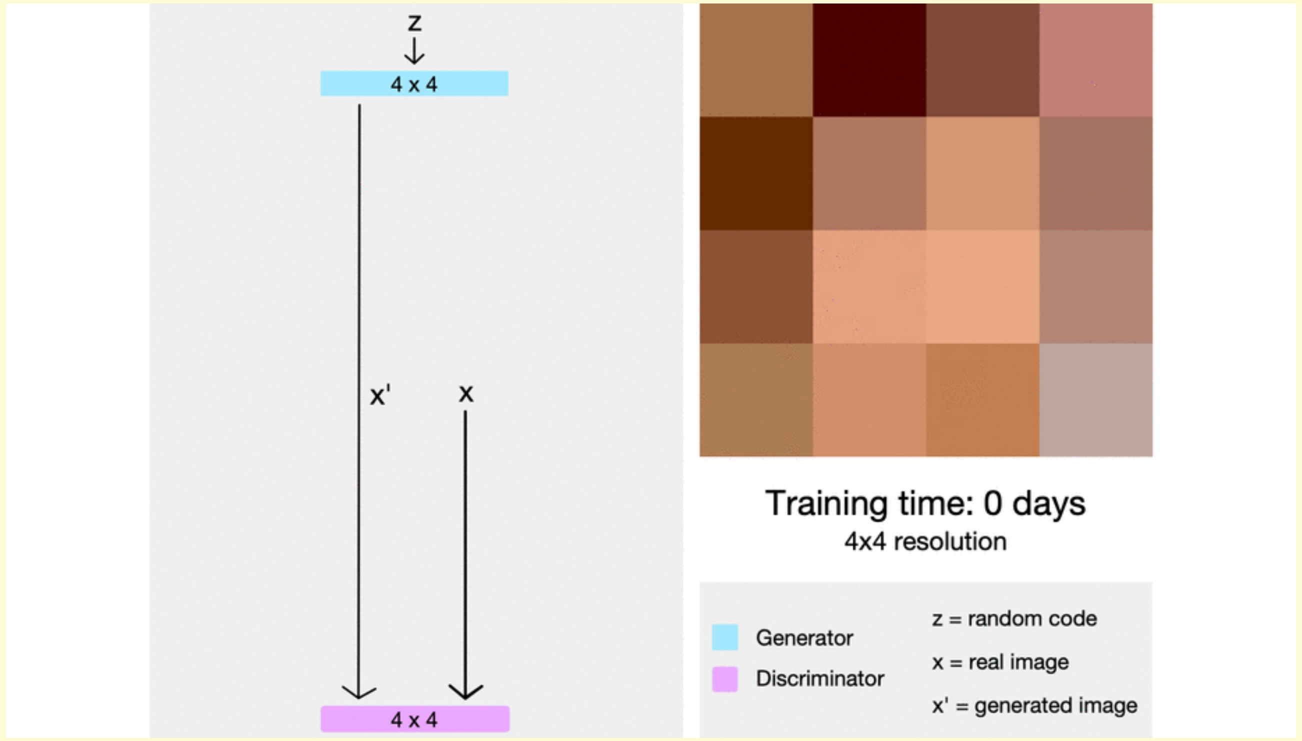


① 학습을 진행하는 과정에서 점진적(progressively)으로 네트워크의 레이어를 붙여 나감

② 판별자도 생성자와 마찬가지로 대칭적으로 여러 개의 layer를 붙여 나감

③ 한 번에 전체 크기의 이미지 feature들을 학습시키지 않음 (4 x 4의 저해상도로 start)

PGGAN (Progressive Growing of GANs)



Main Idea

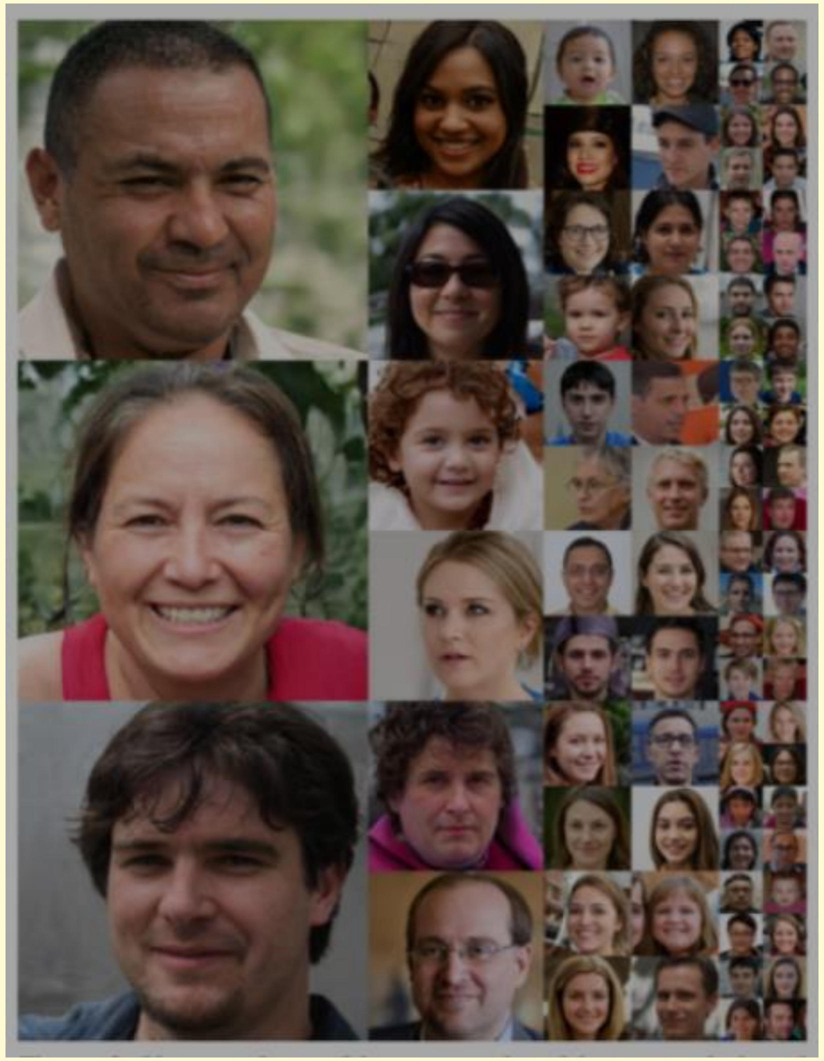
- 학습 과정에서 레이어를 추가
- 고해상도 이미지 학습 성공 (1024 x 1024)

Limitation

- 이미지의 특징 제어 어려움

StyleGAN 에서 개선

StyleGAN

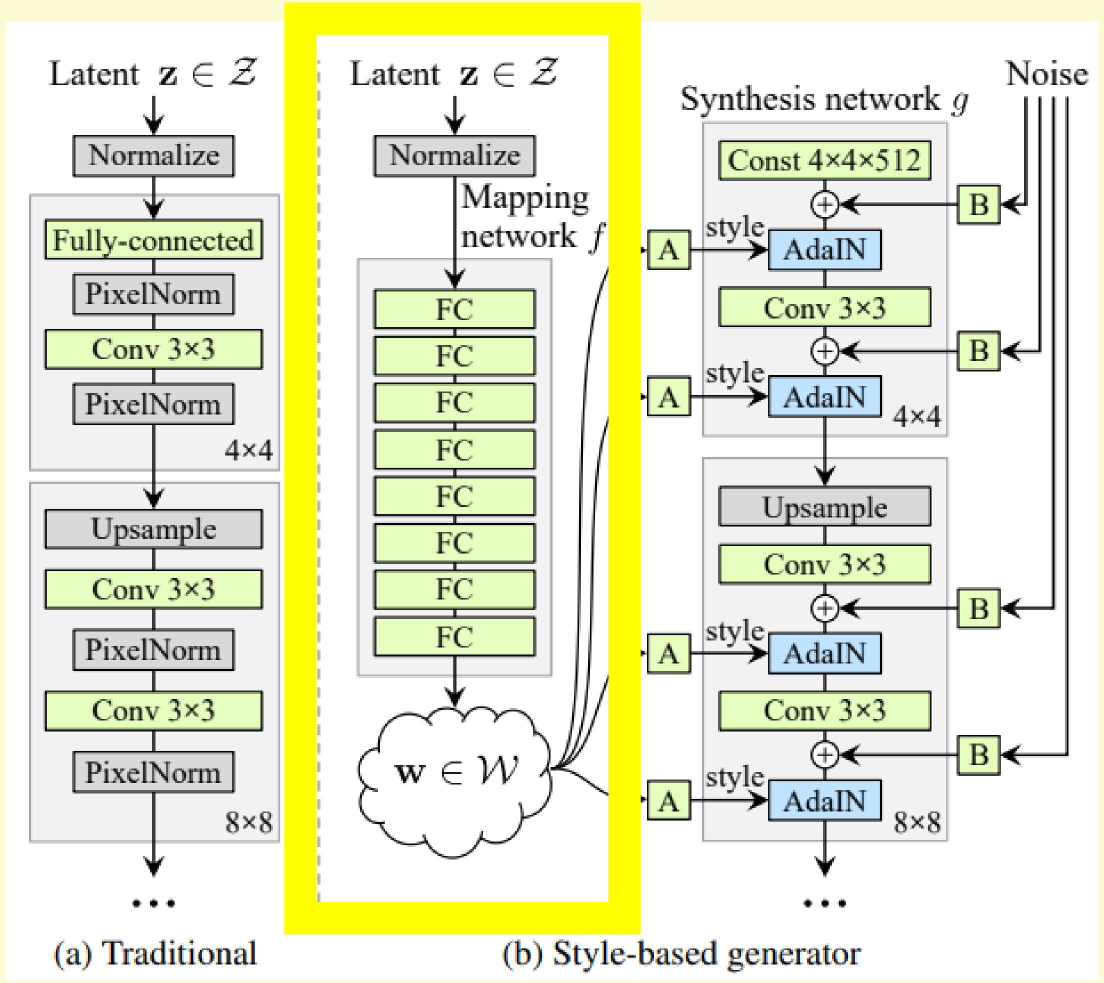


① PGGAN 베이스라인 아키텍처의 성능 향상

② Disentanglement 특성 향상

③ 고해상도 얼굴 데이터셋(FFHQ) 발표

StyleGAN Architecture



(a) Traditional (b) Style-based generator

PGGAN

Mapping Network

- 특정 space에서의 벡터를 다른 space에서의 벡터로 매핑을 시켜주는 네트워크

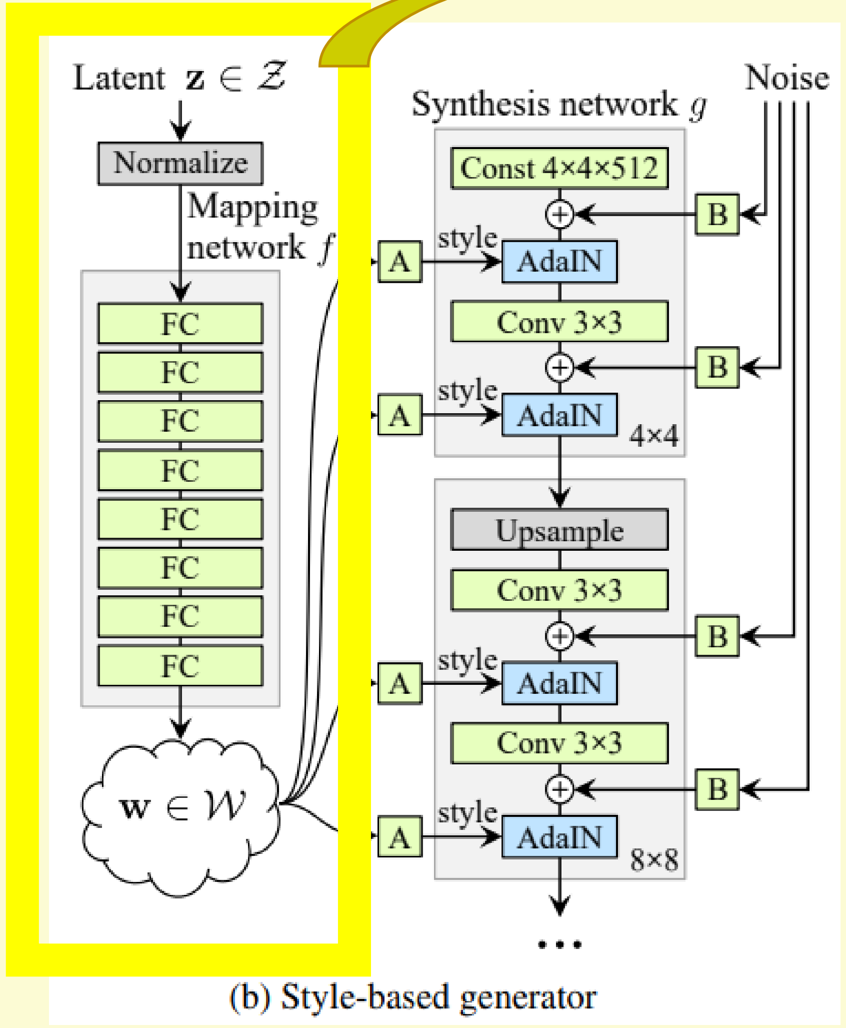
① Z도메인에서 W도메인으로 매핑 수행

② W vector를 각각의 Block에 넣어줌

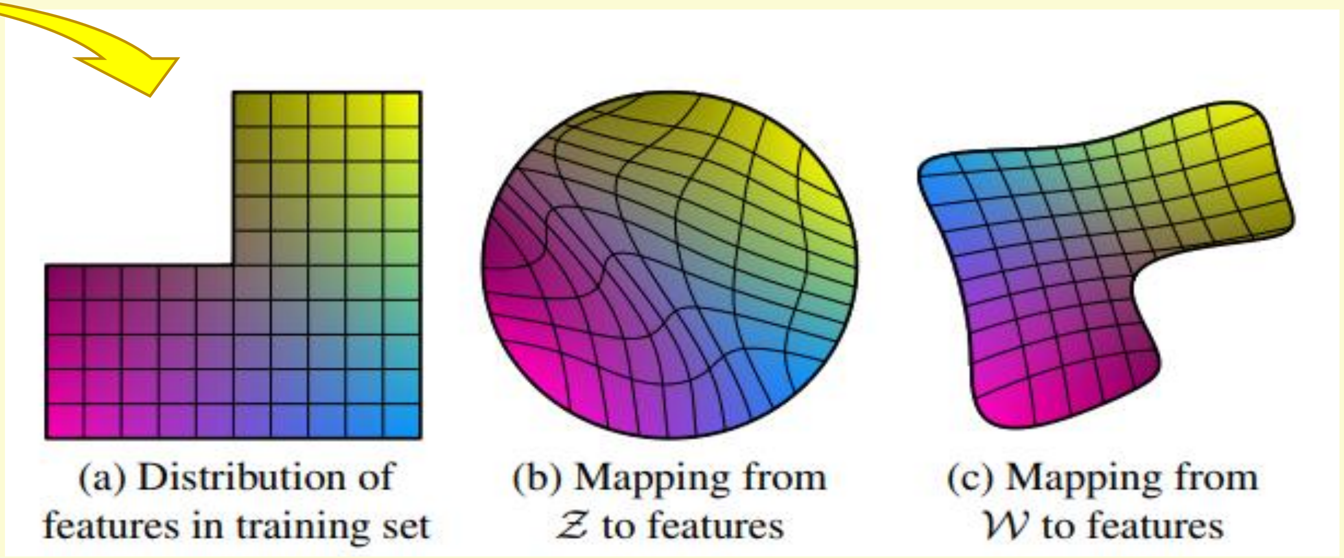
③ Affine transformation을 해준 후 AdaIN에 넣음

④ Noise를 추가

StyleGAN Architecture



(b) Style-based generator


(a) Distribution of features in training set

(b) Mapping from $\mathcal{Z}$ to features

(c) Mapping from $\mathcal{W}$ to features

(a) 실제 학습데이터가 가지는 분포

ex) 세로축: 남성, 여성

가로축: 머리 길이

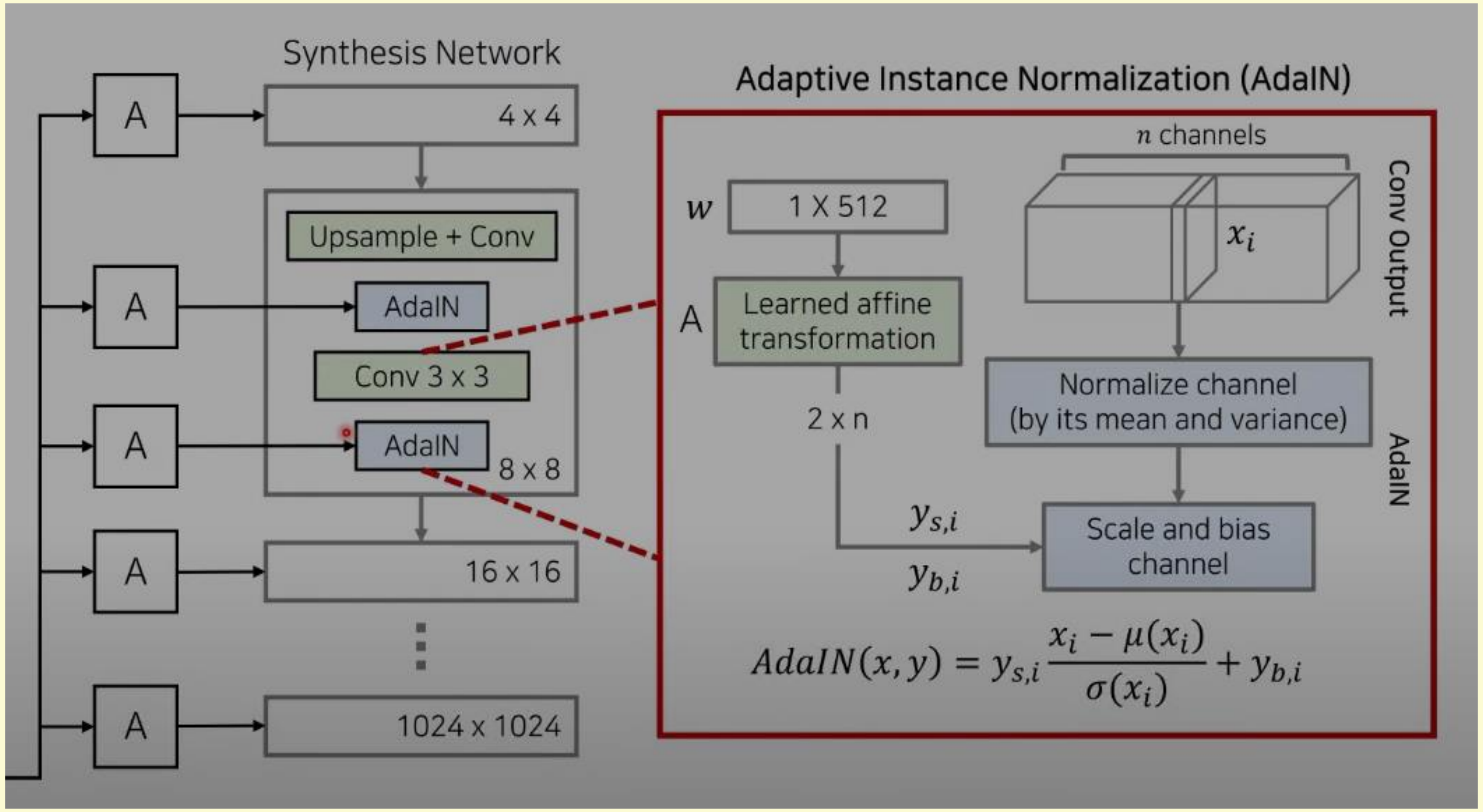
(b) 가우시안 분포를 따르는 *latent vector*를 샘플링

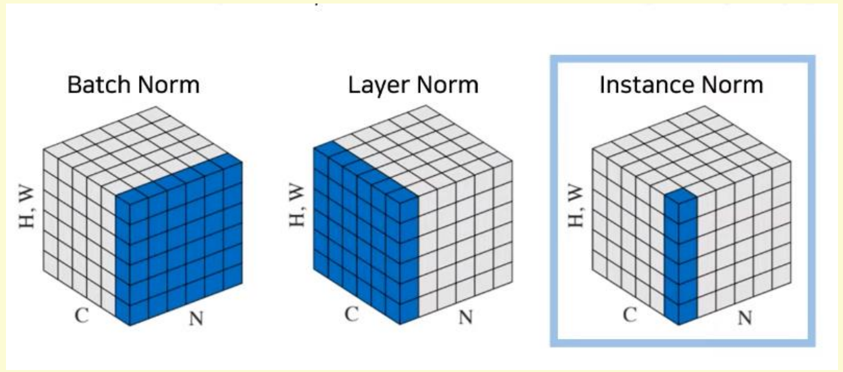
-> entangled됨

(c) *Mapping network*를 거쳐 *latent vector w*

-> 각각의 특징에 대해서 interpolate을 수행할 때 linear한 상태가 될 확률 up

StyleGAN Architecture – *Adaptive Instance Normalization* ***(AdaIN)***

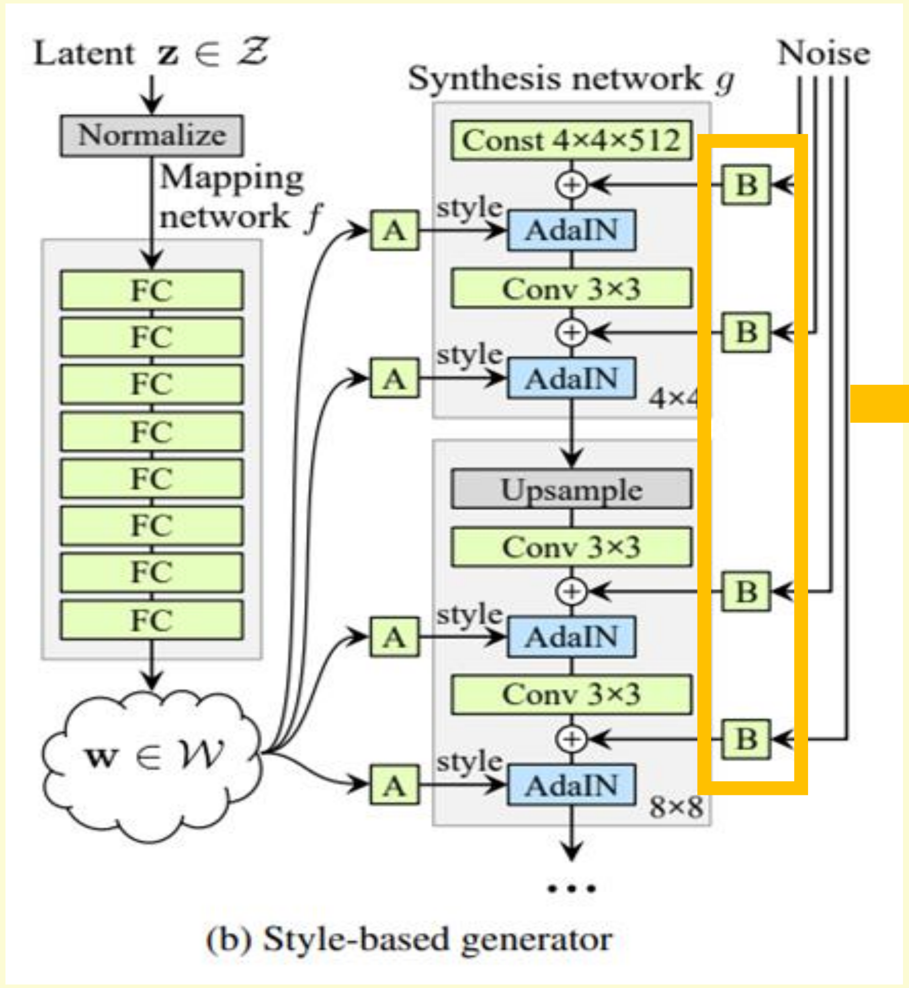

Synthesis Network
A
4 x 4
Upsample + Conv
AdaIN
Conv 3 x 3
AdaIN
8 x 8
16 x 16
1024 x 1024
Adaptive Instance Normalization (AdaIN)
n channels
w
1 X 512
A
Learned affine transformation
2 x n
x_i
Conv Output
Normalize channel (by its mean and variance)
AdaIN
$y_{s,i}$
$y_{b,i}$
Scale and bias channel
$AdaIN(x,y) = y_{s,i}\frac{x_i - \mu(x_i)}{\sigma(x_i)} + y_{b,i}$



$$\mathrm{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i}\frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i}$$

각 채널 단위로 정규화를 수행한 뒤에 별도의 스타일 정보를 입력 받아서 feature상의 statistics을 바꾸는 방식

StyleGAN Architecture – *Stochastic Variation*



(b) Style-based generator

Stochastic Variation (확률론적 변이)

- 다양한 **확률적인 측면**을 컨트롤 할 수 있게 해줌

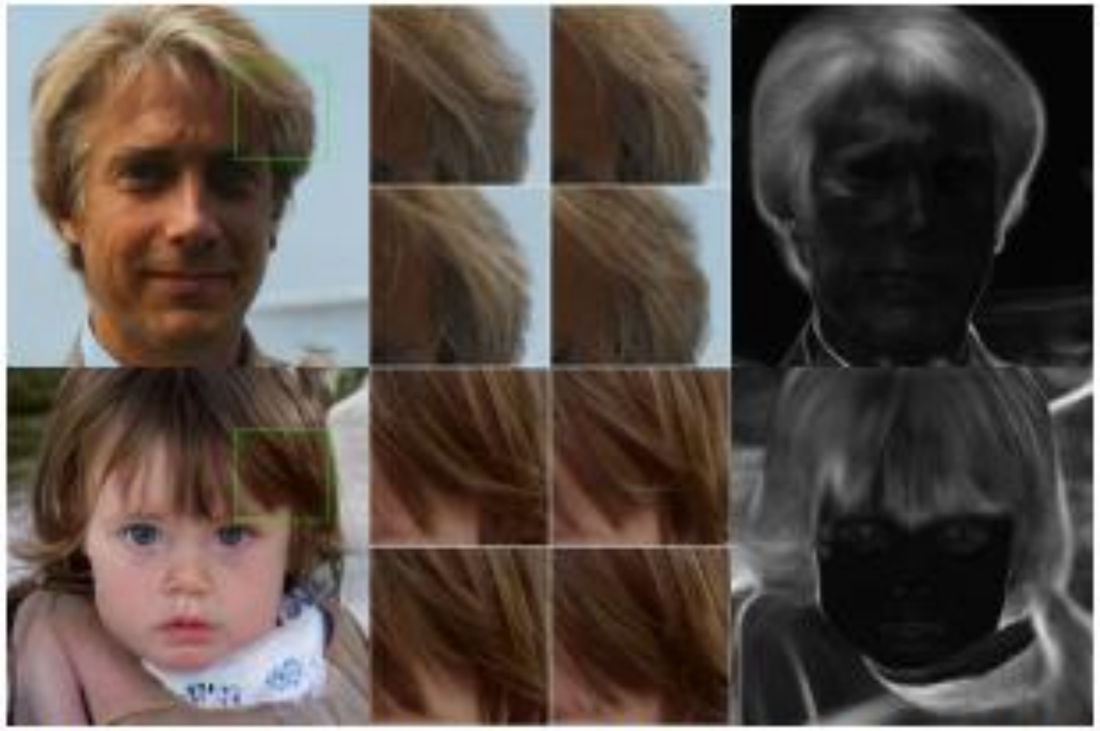
  ex) 주근깨, 머리카락의 배치, 여드름 등

StyleGAN Architecture – Stochastic Variation

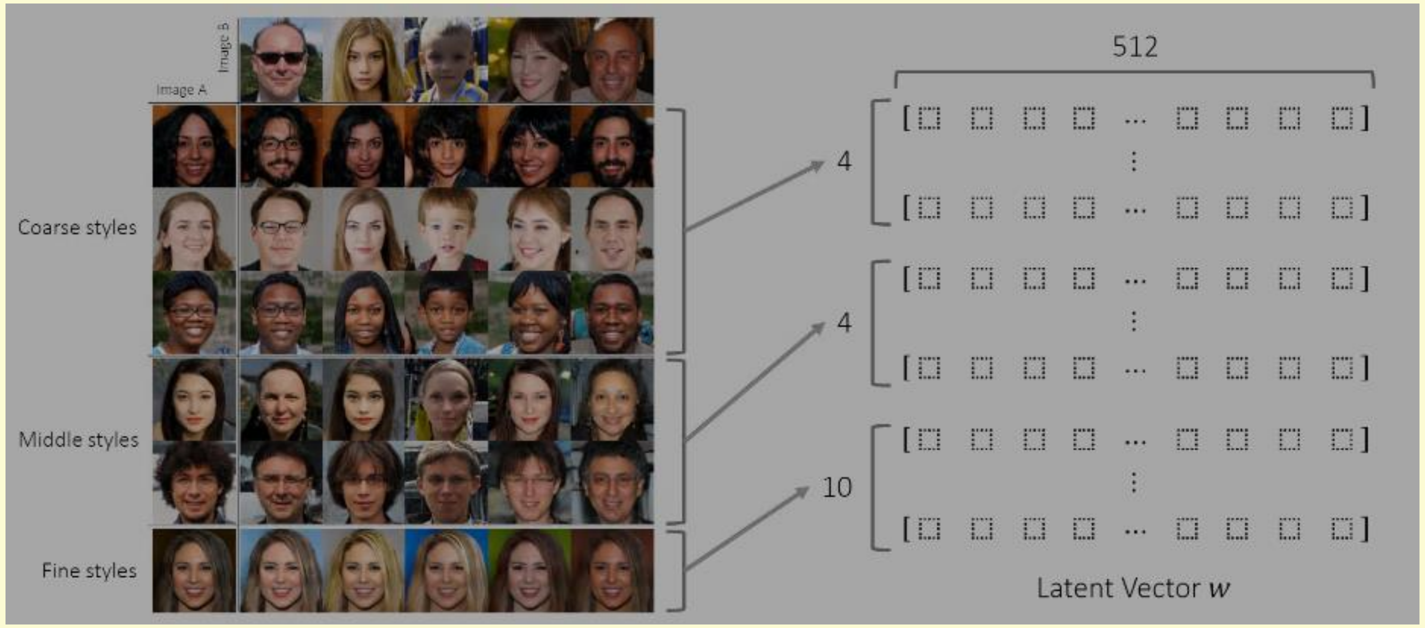
Stochastic Variation (확률론적 변이)

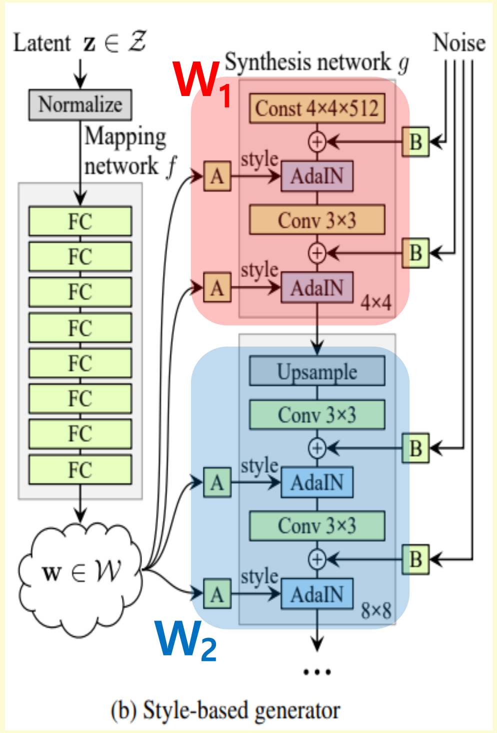
- 다양한 확률적인 측면을 컨트롤 할 수 있게 해줌

  ex) 주근깨, 머리카락의 배치, 여드름 등


(a) Generated image (b) Stochastic variation (c) Standard deviation

StyleGAN Architecture – *Mixing Regularization*





인접한 layer 간의 style 상관관계를 줄여 나가는 것

- 각각의 스타일이 잘 localize되어서 다른 layer에 관여하지 않도록

crossover point 설정

StyleGAN 한계점

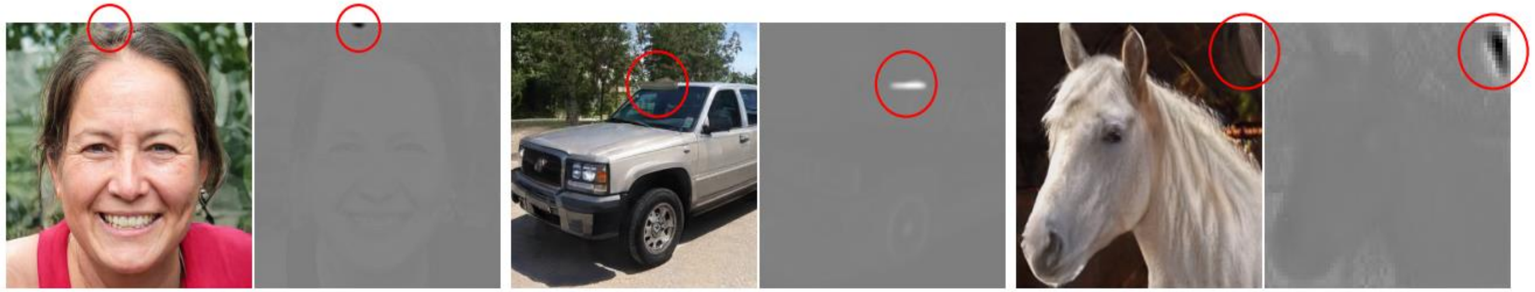
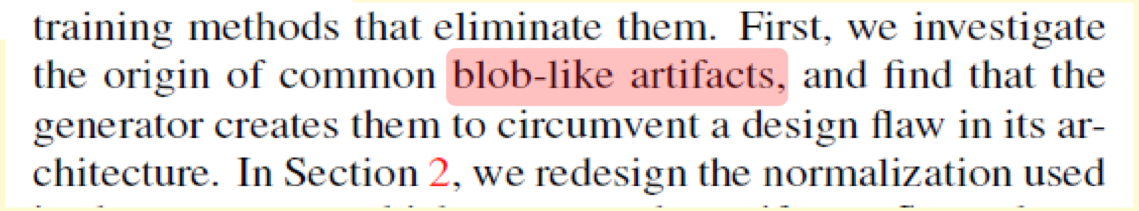


Figure 1. Instance normalization causes water droplet -like artifacts in StyleGAN images. These are not always obvious in the generated images, but if we look at the activations inside the generator network, the problem is always there, in all feature maps starting from the 64x64 resolution. It is a systemic problem that plagues all StyleGAN images.

1) blob-like artifact 발생



training methods that eliminate them. First, we investigate the origin of common blob-like artifacts, and find that the generator creates them to circumvent a design flaw in its architecture. In Section 2, we redesign the normalization used

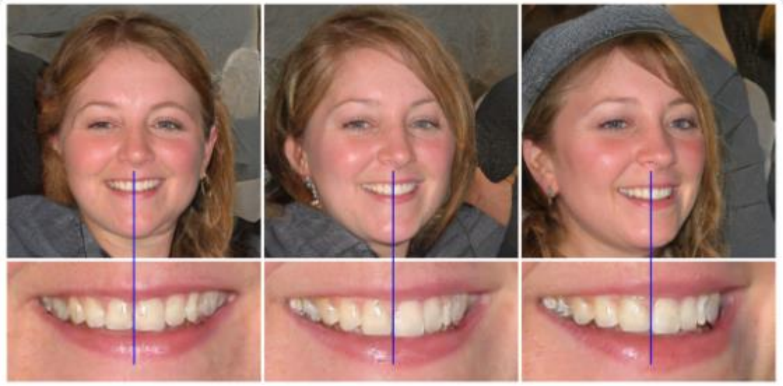


Figure 6. Progressive growing leads to "phase" artifacts. In this example the teeth do not follow the pose but stay aligned to the camera, as indicated by the blue line.

2) phase artifact (fixed position)

StyleGAN2 등장

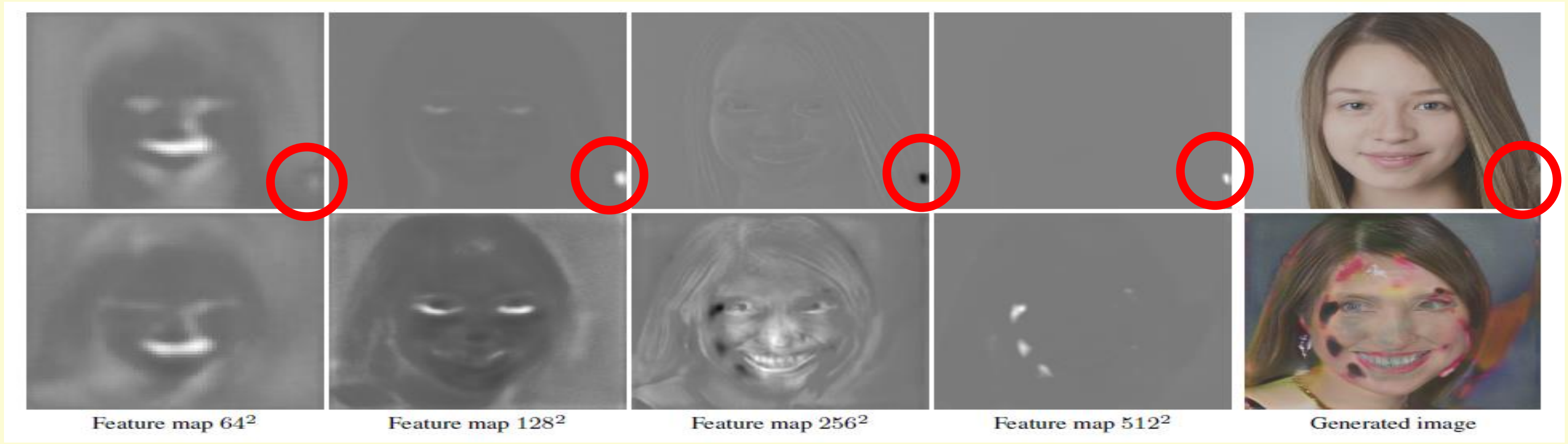
① 이전과 마찬가지로 Z → **W 매핑 네트워크**를 사용

② 기존 아키텍처의 **문제점** 지적

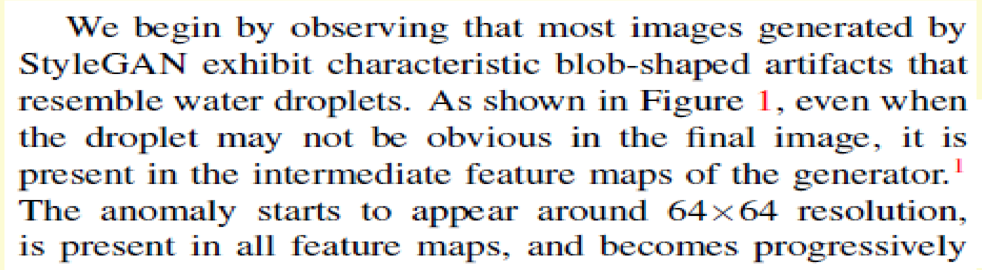
1) **blob-like artifact** 발생

2) **phase artifact** 발생

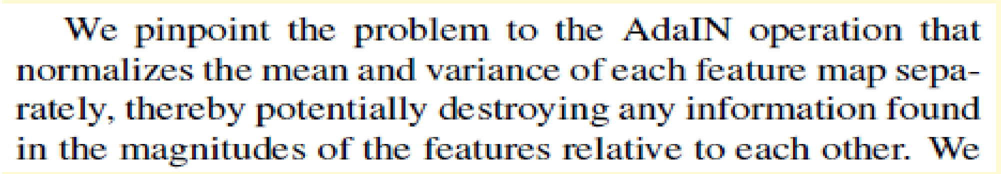
StyleGAN2 1) blob-like artifact 발생



- AdaIN이 원인으로 추정
- 대략 64 x 64 해상도에서부터 이러한 artifact가 보이기 시작



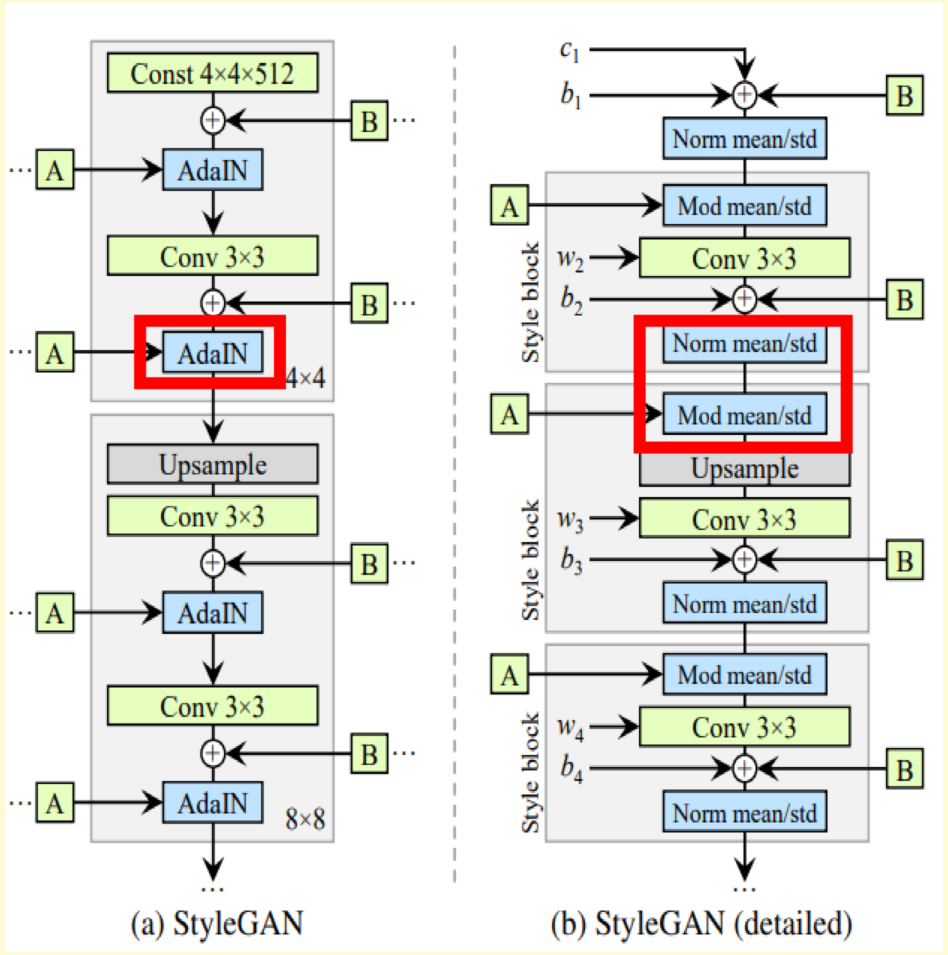
We begin by observing that most images generated by StyleGAN exhibit characteristic blob-shaped artifacts that resemble water droplets. As shown in Figure 1, even when the droplet may not be obvious in the final image, it is present in the intermediate feature maps of the generator.[1] The anomaly starts to appear around 64×64 resolution, is present in all feature maps, and becomes progressively

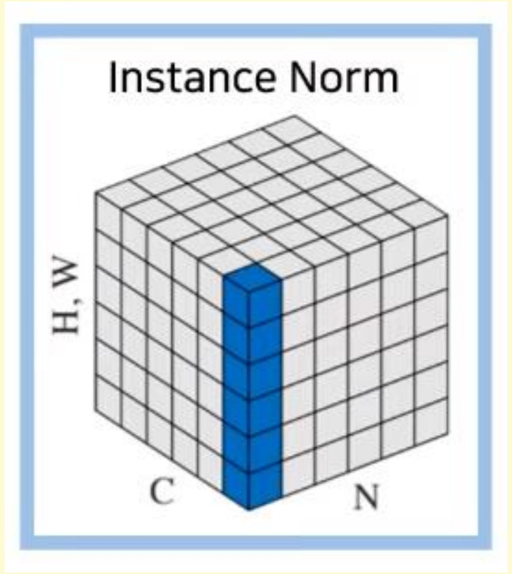


We pinpoint the problem to the AdaIN operation that normalizes the mean and variance of each feature map separately, thereby potentially destroying any information found in the magnitudes of the features relative to each other. We

StyleGAN2

1) blob-like artifact 발생



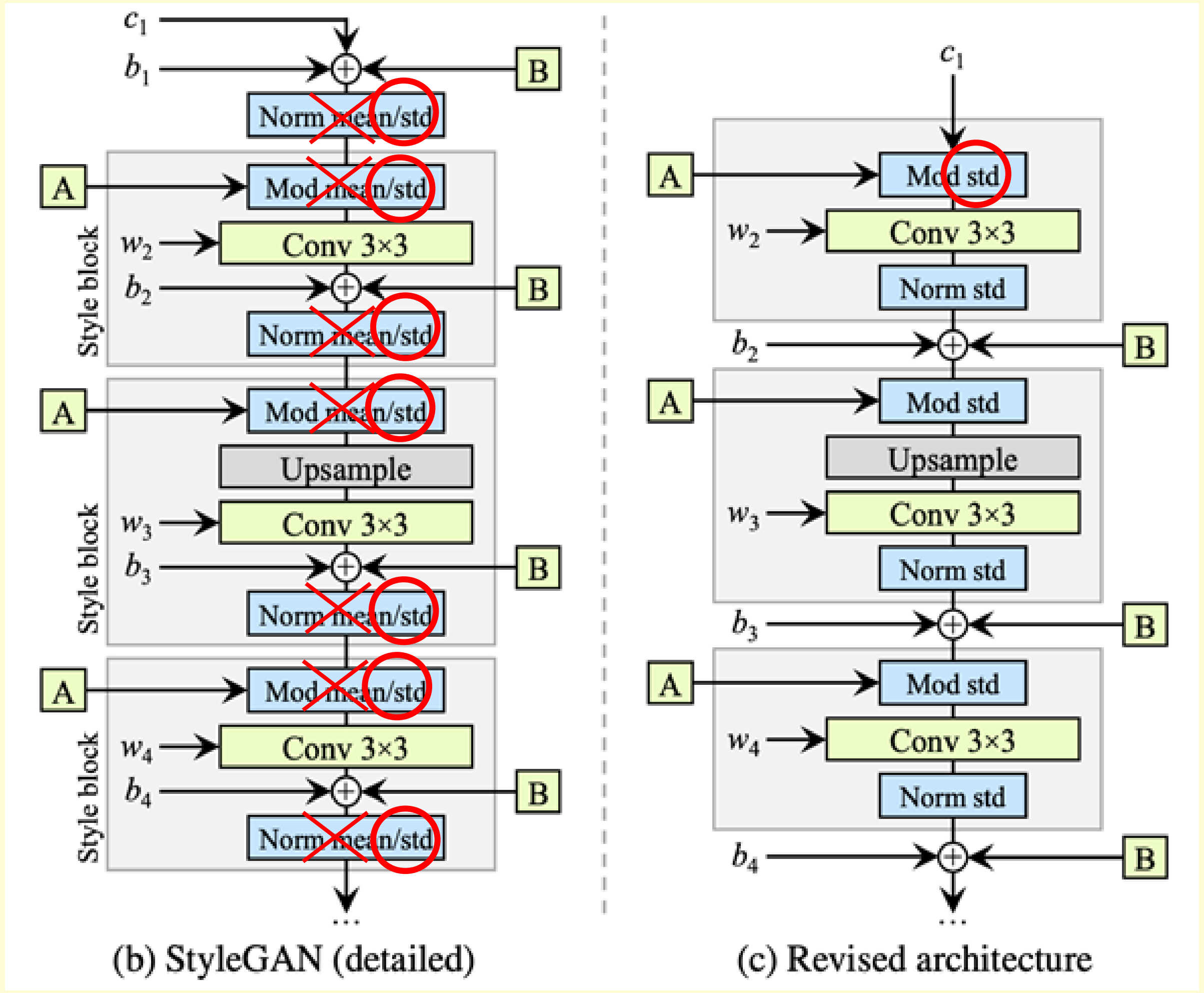


원인) AdaIN

각 feature map의 평균과 분산을 개별적으로normalize 하여 서로 연관된 feature들의 정보가 소실될 수 있음

StyleGAN2

1) blob-like artifact 해결책



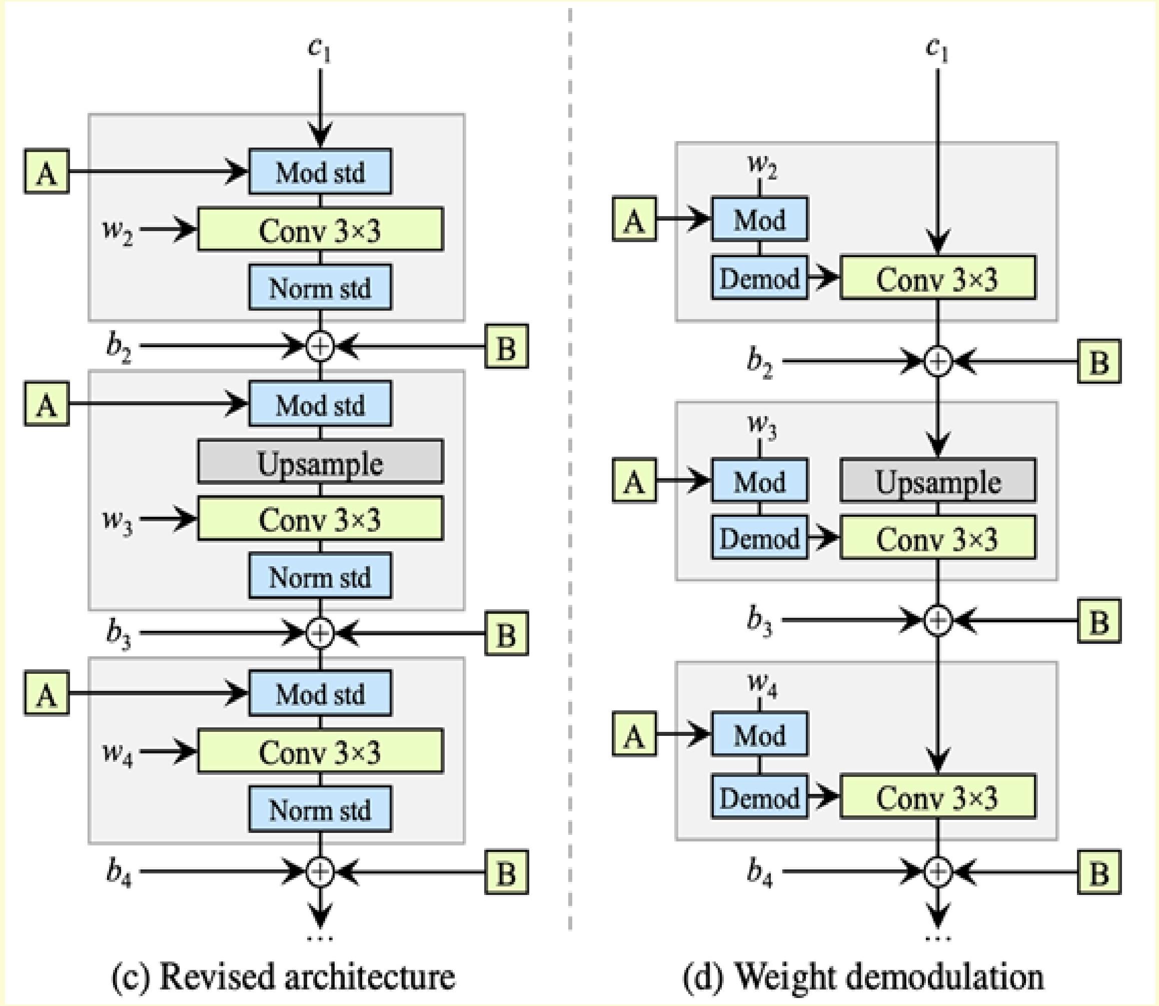
Normalization 단계를 제거

① Mean은 건드리지 말고 standard deviation(표준편차)만 변경해도 충분

② 최대한 style block 외부에서 feature map들의 값을 변경하고자 함

StyleGAN2

1) blob-like artifact 해결책



(c) Revised architecture (d) Weight demodulation

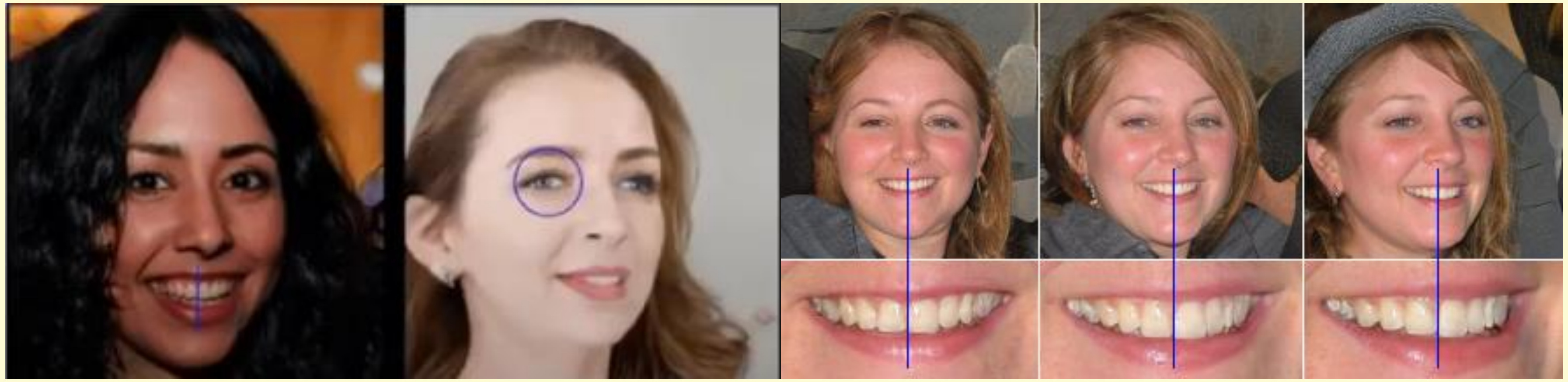
Normalization 단계를 제거
Demodulation 대체

① Weight 값에 대하여 modulation 진행

② 직접적으로 정규화를 하지 않고 feature map의 statistics를 예측하여 정규화를 적용

StyleGAN2

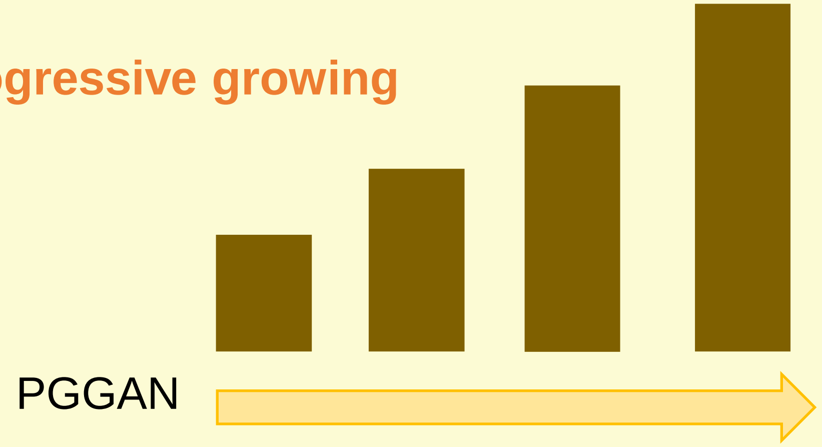
2) phase artifact 발생



Phase artifact란?

얼굴의 특정 부분이 **고정된 위치**를 갖는 문제 발생

원인) Progressive growing

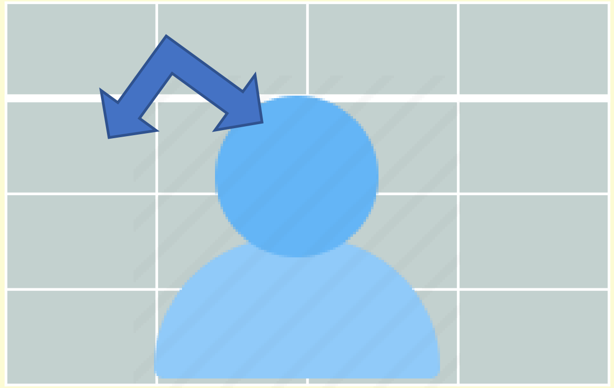


① 각 resolution이 output resolution에 즉시 영향을 미침

② 최대한 높은 frequency detail을 유발함

StyleGAN2

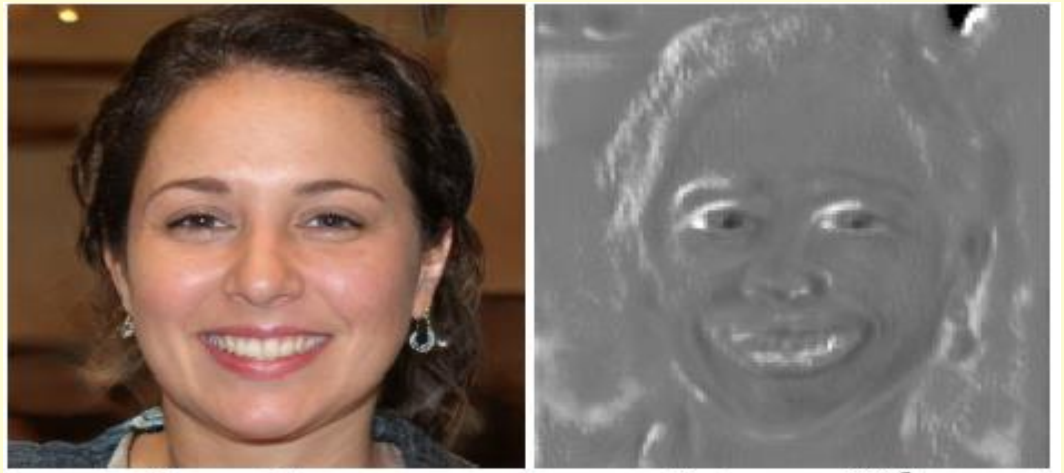
2) phase artifact 발생



높은 frequency detail 란?

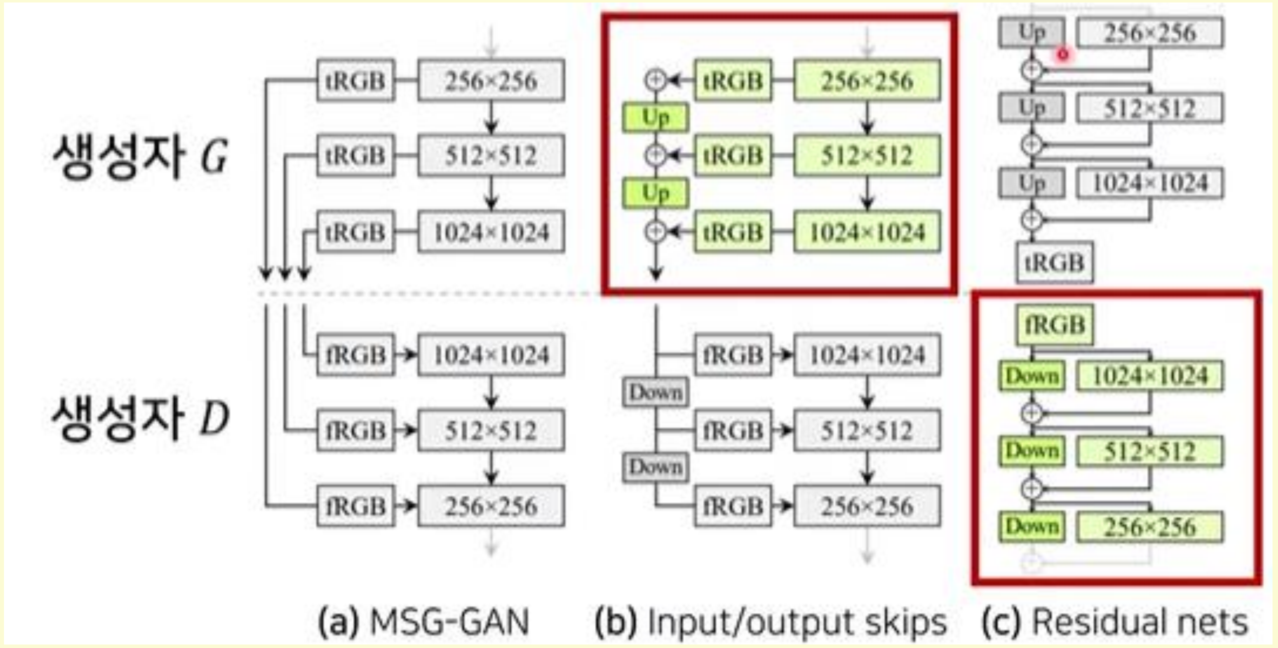
인접한 픽셀 간의 값의 변화가 크다는 것

저해상도 부분에서 높은 frequency detail을 갖는 이미지를 만들어버렸다면
후반부에 있는 layer는 이미지 detail을 살리는 방식으로 결과 생성


Generated image Feature map 128^2

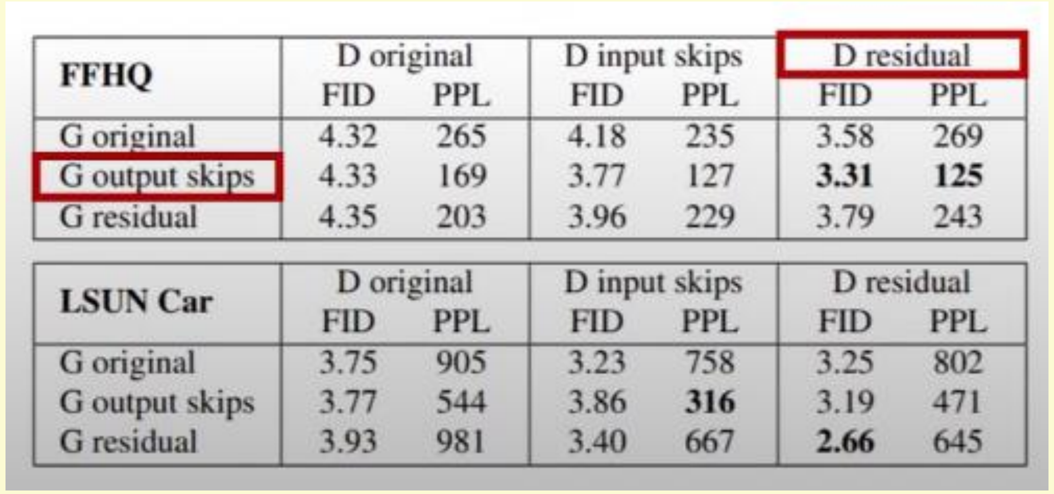
StyleGAN2

2) phase artifact 해결책



(a) MSG-GAN (b) Input/output skips (c) Residual nets

단순한 feedforward 네트워크를 사용하는 방식을 다시 사용

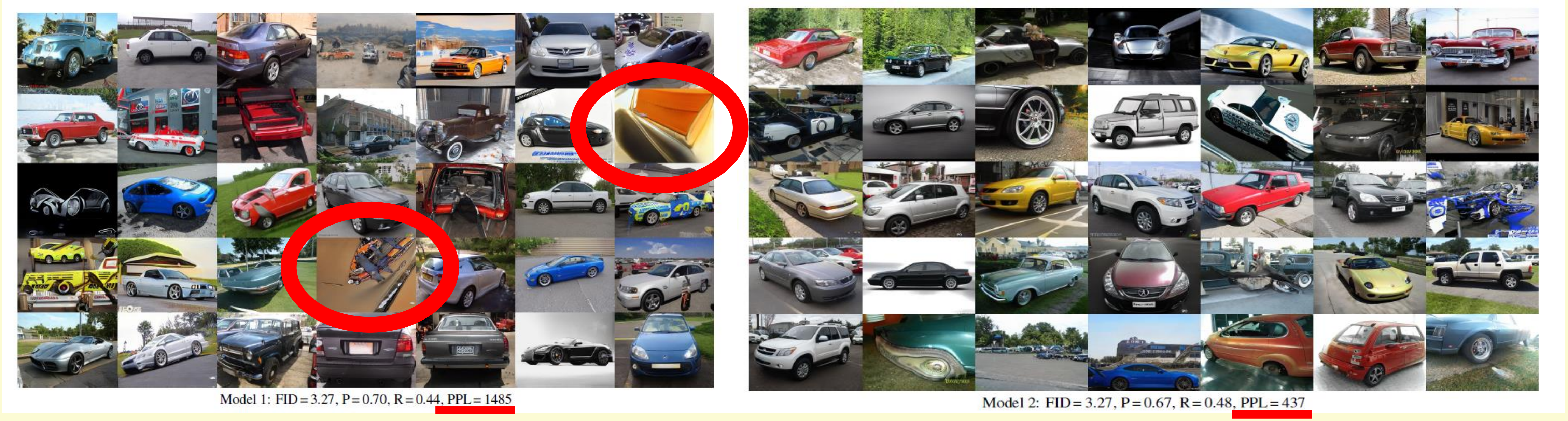


FFHQ	D original		D input skips		D residual	
	FID	PPL	FID	PPL	FID	PPL
G original	4.32	265	4.18	235	3.58	269
G output skips	4.33	169	3.77	127	**3.31**	**125**
G residual	4.35	203	3.96	229	3.79	243

LSUN Car	D original		D input skips		D residual	
	FID	PPL	FID	PPL	FID	PPL
G original	3.75	905	3.23	758	3.25	802
G output skips	3.77	544	3.86	**316**	3.19	471
G residual	3.93	981	3.40	667	**2.66**	645

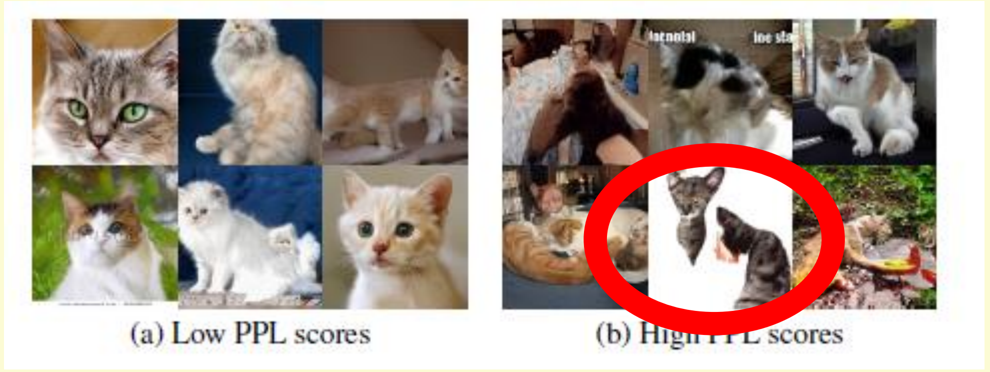
[결과]
G는 output skips , D는 residual net을 쓸 때 우수

StyleGAN2 *Image Quality and Generator Smoothness*


Model 1: FID = 3.27, P = 0.70, R = 0.44, PPL = 1485
Model 2: FID = 3.27, P = 0.67, R = 0.48, PPL = 437

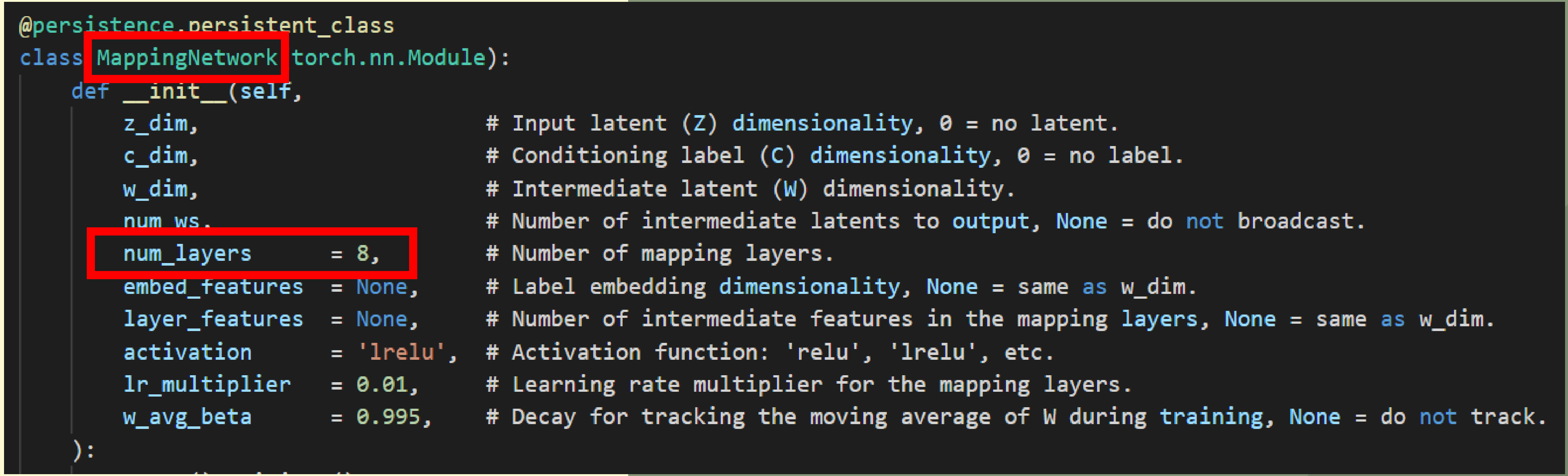
① FID, Precision & Recall score이 같더라도 **PPL**에 따라서 image quality가 달라질 수 있음

② **PPL**이 낮을수록 자동차 모양의 object를 잘 생성함



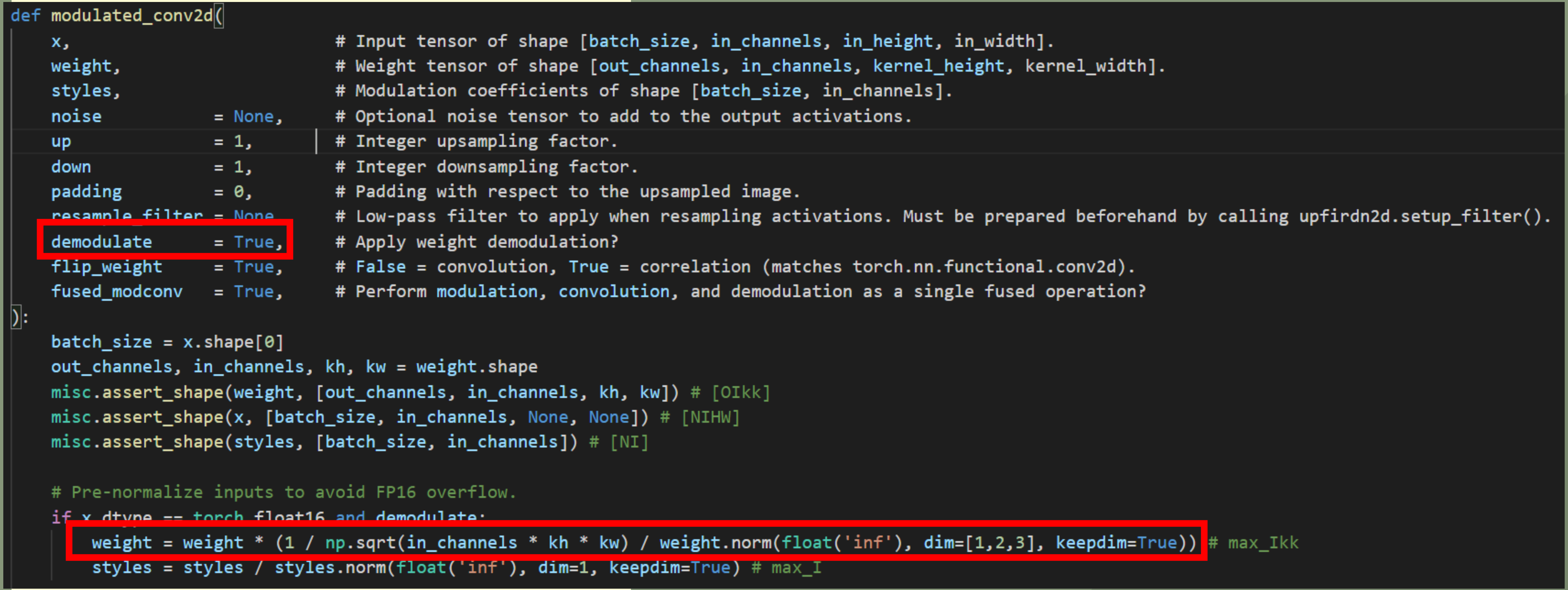
Code

code



```
@persistence.persistent_class
class MappingNetwork(torch.nn.Module):
    def __init__(self,
        z_dim,                      # Input latent (Z) dimensionality, 0 = no latent.
        c_dim,                      # Conditioning label (C) dimensionality, 0 = no label.
        w_dim,                      # Intermediate latent (W) dimensionality.
        num_ws,                     # Number of intermediate latents to output, None = do not broadcast.
        num_layers      = 8,        # Number of mapping layers.
        embed_features  = None,     # Label embedding dimensionality, None = same as w_dim.
        layer_features  = None,     # Number of intermediate features in the mapping layers, None = same as w_dim.
        activation      = 'lrelu',  # Activation function: 'relu', 'lrelu', etc.
        lr_multiplier   = 0.01,     # Learning rate multiplier for the mapping layers.
        w_avg_beta      = 0.995,    # Decay for tracking the moving average of W during training, None = do not track.
    ):
```

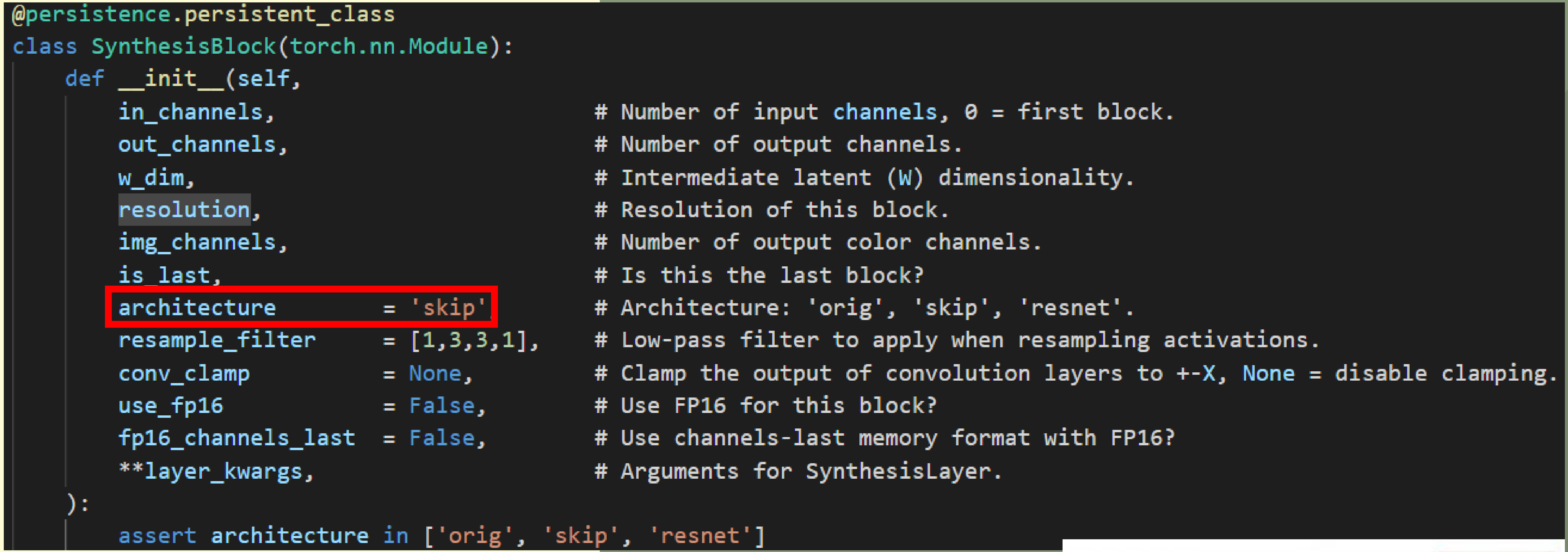
code



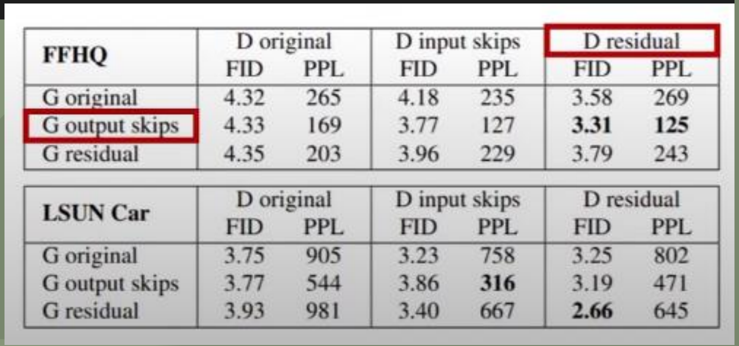
```
def modulated_conv2d(
    x,                          # Input tensor of shape [batch_size, in_channels, in_height, in_width].
    weight,                     # Weight tensor of shape [out_channels, in_channels, kernel_height, kernel_width].
    styles,                     # Modulation coefficients of shape [batch_size, in_channels].
    noise           = None,     # Optional noise tensor to add to the output activations.
    up              = 1,        # Integer upsampling factor.
    down            = 1,        # Integer downsampling factor.
    padding         = 0,        # Padding with respect to the upsampled image.
    resample_filter = None,     # Low-pass filter to apply when resampling activations. Must be prepared beforehand by calling upfirdn2d.setup_filter().
    demodulate      = True,     # Apply weight demodulation?
    flip_weight     = True,     # False = convolution, True = correlation (matches torch.nn.functional.conv2d).
    fused_modconv   = True,     # Perform modulation, convolution, and demodulation as a single fused operation?
):
    batch_size = x.shape[0]
    out_channels, in_channels, kh, kw = weight.shape
    misc.assert_shape(weight, [out_channels, in_channels, kh, kw]) # [OIkk]
    misc.assert_shape(x, [batch_size, in_channels, None, None]) # [NIHW]
    misc.assert_shape(styles, [batch_size, in_channels]) # [NI]

    # Pre-normalize inputs to avoid FP16 overflow.
    if x.dtype == torch.float16 and demodulate:
        weight = weight * (1 / np.sqrt(in_channels * kh * kw) / weight.norm(float('inf'), dim=[1,2,3], keepdim=True)) # max_Ikk
        styles = styles / styles.norm(float('inf'), dim=1, keepdim=True) # max_I
```

code



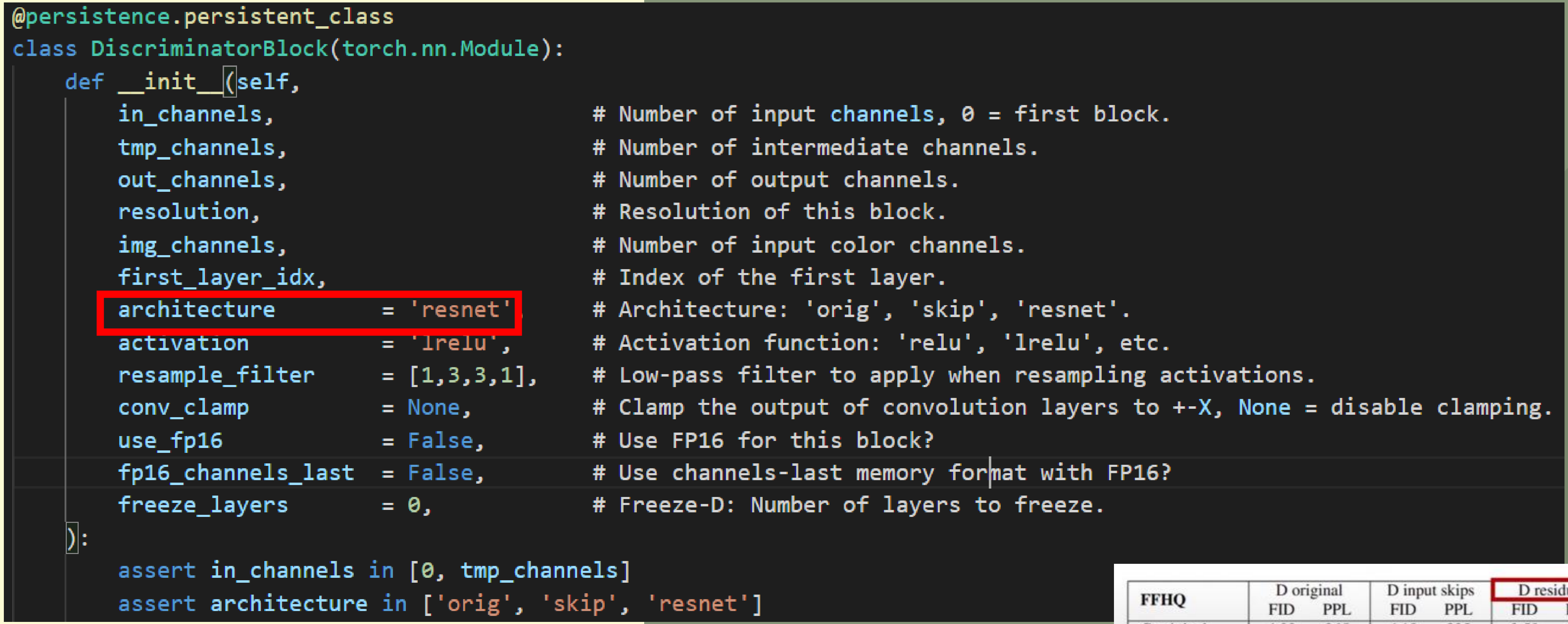
```python
@persistence.persistent_class
class SynthesisBlock(torch.nn.Module):
    def __init__(self,
        in_channels,                            # Number of input channels, 0 = first block.
        out_channels,                           # Number of output channels.
        w_dim,                                  # Intermediate latent (W) dimensionality.
        resolution,                             # Resolution of this block.
        img_channels,                           # Number of output color channels.
        is_last,                                # Is this the last block?
        architecture        = 'skip',           # Architecture: 'orig', 'skip', 'resnet'.
        resample_filter     = [1,3,3,1],        # Low-pass filter to apply when resampling activations.
        conv_clamp          = None,             # Clamp the output of convolution layers to +-X, None = disable clamping.
        use_fp16            = False,            # Use FP16 for this block?
        fp16_channels_last  = False,            # Use channels-last memory format with FP16?
        **layer_kwargs,                         # Arguments for SynthesisLayer.
    ):
        assert architecture in ['orig', 'skip', 'resnet']
```



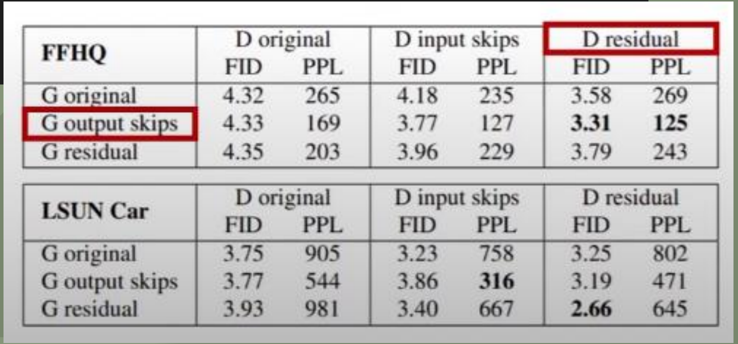
FFHQ	D original		D input skips		D residual	
	FID	PPL	FID	PPL	FID	PPL
G original	4.32	265	4.18	235	3.58	269
G output skips	4.33	169	3.77	127	**3.31**	**125**
G residual	4.35	203	3.96	229	3.79	243

LSUN Car	D original		D input skips		D residual	
	FID	PPL	FID	PPL	FID	PPL
G original	3.75	905	3.23	758	3.25	802
G output skips	3.77	544	3.86	**316**	3.19	471
G residual	3.93	981	3.40	667	**2.66**	645

code



```
@persistence.persistent_class
class DiscriminatorBlock(torch.nn.Module):
    def __init__(self,
        in_channels,                        # Number of input channels, 0 = first block.
        tmp_channels,                       # Number of intermediate channels.
        out_channels,                       # Number of output channels.
        resolution,                         # Resolution of this block.
        img_channels,                       # Number of input color channels.
        first_layer_idx,                    # Index of the first layer.
        architecture        = 'resnet',     # Architecture: 'orig', 'skip', 'resnet'.
        activation          = 'lrelu',      # Activation function: 'relu', 'lrelu', etc.
        resample_filter     = [1,3,3,1],    # Low-pass filter to apply when resampling activations.
        conv_clamp          = None,         # Clamp the output of convolution layers to +-X, None = disable clamping.
        use_fp16            = False,        # Use FP16 for this block?
        fp16_channels_last  = False,        # Use channels-last memory format with FP16?
        freeze_layers       = 0,            # Freeze-D: Number of layers to freeze.
    ):
        assert in_channels in [0, tmp_channels]
        assert architecture in ['orig', 'skip', 'resnet']
```

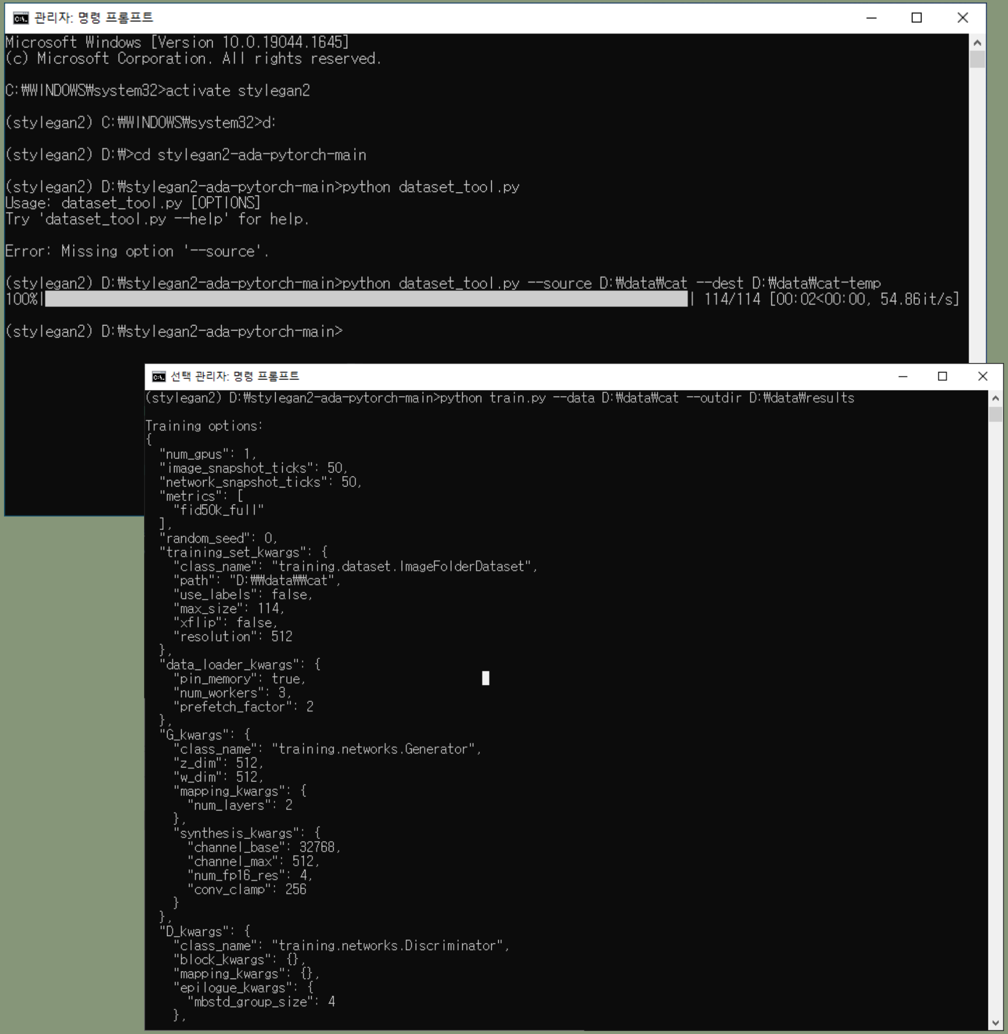


FFHQ	D original		D input skips		D residual	
	FID	PPL	FID	PPL	FID	PPL
G original	4.32	265	4.18	235	3.58	269
G output skips	4.33	169	3.77	127	**3.31**	**125**
G residual	4.35	203	3.96	229	3.79	243

LSUN Car	D original		D input skips		D residual	
	FID	PPL	FID	PPL	FID	PPL
G original	3.75	905	3.23	758	3.25	802
G output skips	3.77	544	3.86	**316**	3.19	471
G residual	3.93	981	3.40	667	**2.66**	645

output

output



관리자: 명령 프롬프트

```
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. All rights reserved.

C:₩WINDOWS₩system32>activate stylegan2

(stylegan2) C:₩WINDOWS₩system32>d:

(stylegan2) D:₩>cd stylegan2-ada-pytorch-main

(stylegan2) D:₩stylegan2-ada-pytorch-main>python dataset_tool.py
Usage: dataset_tool.py [OPTIONS]
Try 'dataset_tool.py --help' for help.

Error: Missing option '--source'.

(stylegan2) D:₩stylegan2-ada-pytorch-main>python dataset_tool.py --source D:₩data₩cat --dest D:₩data₩cat-temp
100%|████████████████████████████████████████████████████| 114/114 [00:02<00:00, 54.86it/s]

(stylegan2) D:₩stylegan2-ada-pytorch-main>
```

선택 관리자: 명령 프롬프트

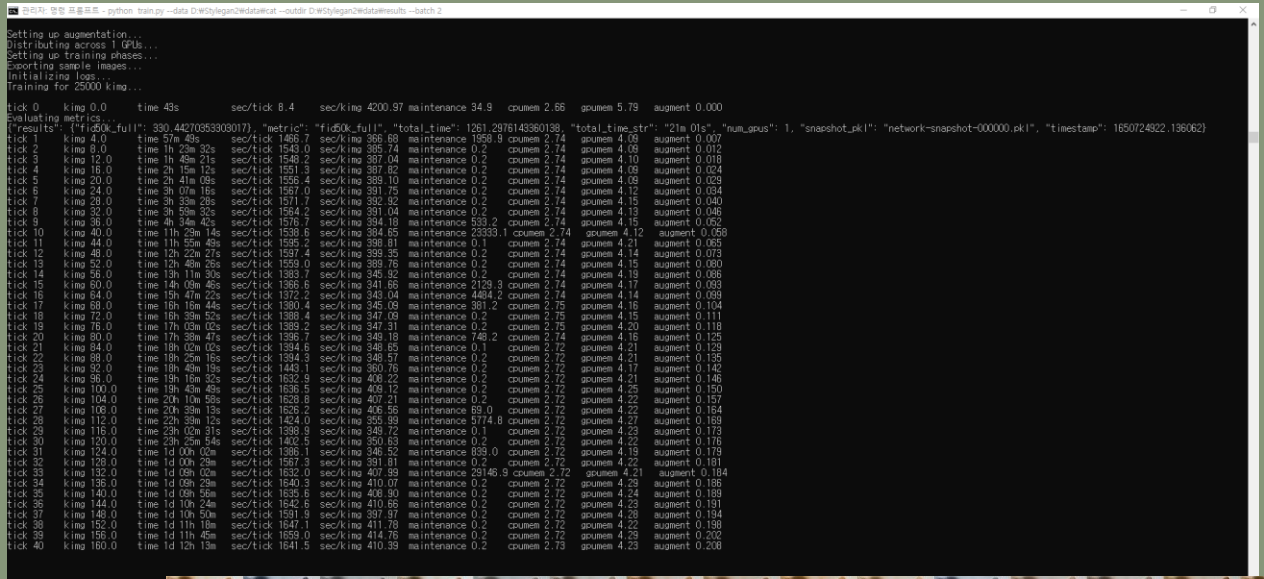
```
(stylegan2) D:₩stylegan2-ada-pytorch-main>python train.py --data D:₩data₩cat --outdir D:₩data₩results

Training options:
{
  "num_gpus": 1,
  "image_snapshot_ticks": 50,
  "network_snapshot_ticks": 50,
  "metrics": [
    "fid50k_full"
  ],
  "random_seed": 0,
  "training_set_kwargs": {
    "class_name": "training.dataset.ImageFolderDataset",
    "path": "D:₩₩data₩₩cat",
    "use_labels": false,
    "max_size": 114,
    "xflip": false,
    "resolution": 512
  },
  "data_loader_kwargs": {
    "pin_memory": true,
    "num_workers": 3,
    "prefetch_factor": 2
  },
  "G_kwargs": {
    "class_name": "training.networks.Generator",
    "z_dim": 512,
    "w_dim": 512,
    "mapping_kwargs": {
      "num_layers": 2
    },
    "synthesis_kwargs": {
      "channel_base": 32768,
      "channel_max": 512,
      "num_fp16_res": 4,
      "conv_clamp": 256
    }
  },
  "D_kwargs": {
    "class_name": "training.networks.Discriminator",
    "block_kwargs": {},
    "mapping_kwargs": {},
    "epilogue_kwargs": {
      "mbstd_group_size": 4
    },
```

Real Images

output

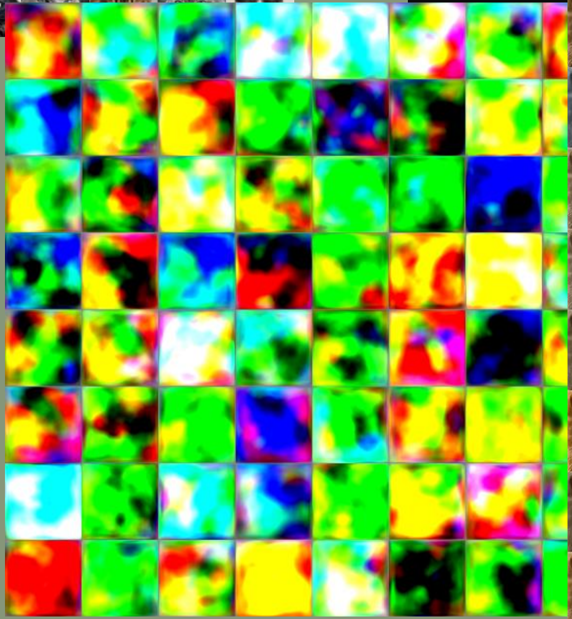
Real Images

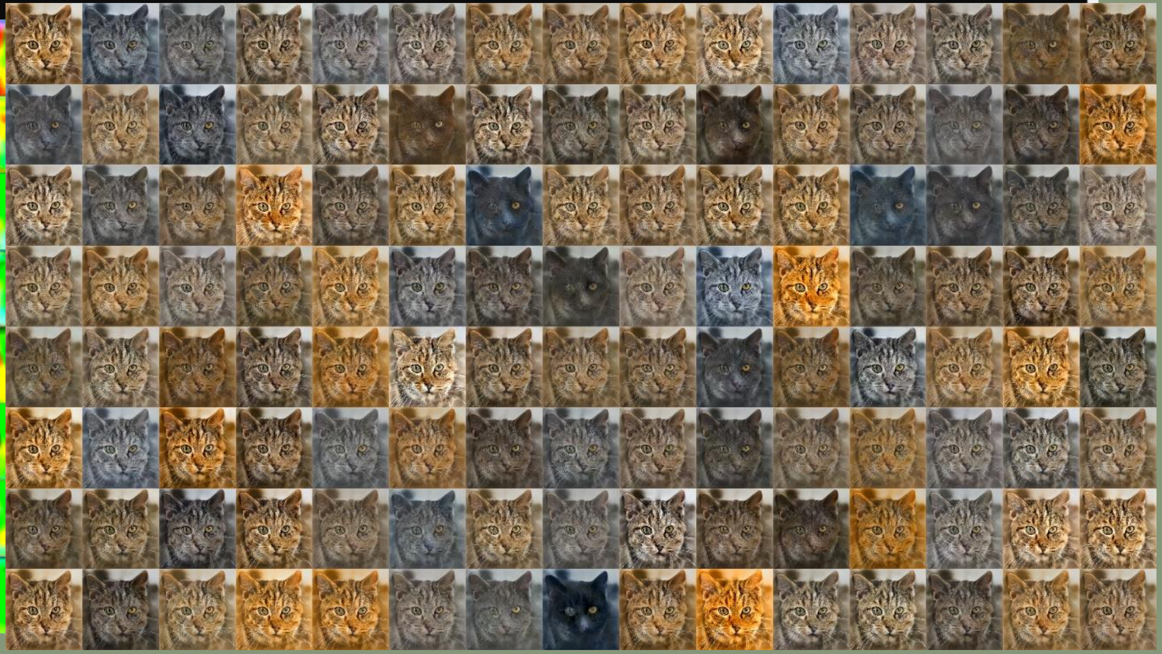


```
관리자: 명령 프롬프트 - python  train.py --data D:\Stylegan2\data\cat --outdir D:\Stylegan2\data\results --batch 2

Setting up augmentation...
Distributing across 1 GPUs...
Setting up training phases...
Exporting sample images...
Initializing logs...
Training for 25000 kimg...

tick 0     kimg 0.0      time 43s          sec/tick 8.4     sec/kimg 4200.97 maintenance 34.9   cpumem 2.66   gpumem 5.79   augment 0.000
Evaluating metrics...
{"results": {"fid50k_full": 330.44270353303017}, "metric": "fid50k_full", "total_time": 1261.2976143360138, "total_time_str": "21m 01s", "num_gpus": 1, "snapshot_pkl": "network-snapshot-000000.pkl", "timestamp": 1650724922.136062}
tick 1     kimg 4.0      time 57m 49s      sec/tick 1466.7  sec/kimg 366.68  maintenance 1958.9 cpumem 2.74   gpumem 4.09   augment 0.007
tick 2     kimg 8.0      time 1h 23m 32s   sec/tick 1543.0  sec/kimg 385.74  maintenance 0.2    cpumem 2.74   gpumem 4.09   augment 0.012
tick 3     kimg 12.0     time 1h 49m 21s   sec/tick 1548.2  sec/kimg 387.04  maintenance 0.2    cpumem 2.74   gpumem 4.10   augment 0.018
tick 4     kimg 16.0     time 2h 15m 12s   sec/tick 1551.3  sec/kimg 387.82  maintenance 0.2    cpumem 2.74   gpumem 4.09   augment 0.024
tick 5     kimg 20.0     time 2h 41m 09s   sec/tick 1556.4  sec/kimg 389.10  maintenance 0.2    cpumem 2.74   gpumem 4.09   augment 0.029
tick 6     kimg 24.0     time 3h 07m 16s   sec/tick 1567.0  sec/kimg 391.75  maintenance 0.2    cpumem 2.74   gpumem 4.12   augment 0.034
tick 7     kimg 28.0     time 3h 33m 28s   sec/tick 1571.7  sec/kimg 392.92  maintenance 0.2    cpumem 2.74   gpumem 4.15   augment 0.040
tick 8     kimg 32.0     time 3h 59m 32s   sec/tick 1564.2  sec/kimg 391.04  maintenance 0.2    cpumem 2.74   gpumem 4.13   augment 0.046
tick 9     kimg 36.0     time 4h 34m 42s   sec/tick 1576.7  sec/kimg 394.18  maintenance 533.2  cpumem 2.74   gpumem 4.15   augment 0.052
tick 10    kimg 40.0     time 11h 29m 14s  sec/tick 1538.6  sec/kimg 384.65  maintenance 23333.1 cpumem 2.74   gpumem 4.12   augment 0.058
tick 11    kimg 44.0     time 11h 55m 49s  sec/tick 1595.2  sec/kimg 398.81  maintenance 0.1    cpumem 2.74   gpumem 4.21   augment 0.065
tick 12    kimg 48.0     time 12h 22m 27s  sec/tick 1597.4  sec/kimg 399.35  maintenance 0.2    cpumem 2.74   gpumem 4.14   augment 0.073
tick 13    kimg 52.0     time 12h 48m 26s  sec/tick 1559.0  sec/kimg 389.76  maintenance 0.2    cpumem 2.74   gpumem 4.15   augment 0.080
tick 14    kimg 56.0     time 13h 11m 30s  sec/tick 1383.7  sec/kimg 345.92  maintenance 0.2    cpumem 2.74   gpumem 4.19   augment 0.086
tick 15    kimg 60.0     time 14h 09m 46s  sec/tick 1366.6  sec/kimg 341.66  maintenance 2129.3 cpumem 2.74   gpumem 4.17   augment 0.093
tick 16    kimg 64.0     time 15h 47m 22s  sec/tick 1372.2  sec/kimg 343.04  maintenance 4484.2 cpumem 2.74   gpumem 4.14   augment 0.099
tick 17    kimg 68.0     time 16h 16m 44s  sec/tick 1380.4  sec/kimg 345.09  maintenance 381.2  cpumem 2.75   gpumem 4.16   augment 0.104
tick 18    kimg 72.0     time 16h 39m 52s  sec/tick 1388.4  sec/kimg 347.09  maintenance 0.2    cpumem 2.75   gpumem 4.15   augment 0.111
tick 19    kimg 76.0     time 17h 03m 02s  sec/tick 1389.2  sec/kimg 347.31  maintenance 0.2    cpumem 2.75   gpumem 4.20   augment 0.118
tick 20    kimg 80.0     time 17h 38m 47s  sec/tick 1396.7  sec/kimg 349.18  maintenance 748.2  cpumem 2.74   gpumem 4.16   augment 0.125
tick 21    kimg 84.0     time 18h 02m 02s  sec/tick 1394.6  sec/kimg 348.65  maintenance 0.1    cpumem 2.72   gpumem 4.21   augment 0.129
tick 22    kimg 88.0     time 18h 25m 16s  sec/tick 1394.3  sec/kimg 348.57  maintenance 0.2    cpumem 2.72   gpumem 4.21   augment 0.135
tick 23    kimg 92.0     time 18h 49m 19s  sec/tick 1443.1  sec/kimg 360.76  maintenance 0.2    cpumem 2.72   gpumem 4.17   augment 0.142
tick 24    kimg 96.0     time 19h 16m 32s  sec/tick 1632.9  sec/kimg 408.22  maintenance 0.2    cpumem 2.72   gpumem 4.21   augment 0.146
tick 25    kimg 100.0    time 19h 43m 49s  sec/tick 1636.5  sec/kimg 409.12  maintenance 0.2    cpumem 2.72   gpumem 4.25   augment 0.150
tick 26    kimg 104.0    time 20h 10m 58s  sec/tick 1628.8  sec/kimg 407.21  maintenance 0.2    cpumem 2.72   gpumem 4.22   augment 0.157
tick 27    kimg 108.0    time 20h 39m 13s  sec/tick 1626.2  sec/kimg 406.56  maintenance 69.0   cpumem 2.72   gpumem 4.22   augment 0.164
tick 28    kimg 112.0    time 22h 39m 12s  sec/tick 1424.0  sec/kimg 355.99  maintenance 5774.8 cpumem 2.72   gpumem 4.27   augment 0.169
tick 29    kimg 116.0    time 23h 02m 31s  sec/tick 1398.9  sec/kimg 349.72  maintenance 0.1    cpumem 2.72   gpumem 4.23   augment 0.173
tick 30    kimg 120.0    time 23h 25m 54s  sec/tick 1402.5  sec/kimg 350.63  maintenance 0.2    cpumem 2.72   gpumem 4.22   augment 0.176
tick 31    kimg 124.0    time 1d 00h 02m   sec/tick 1386.1  sec/kimg 346.52  maintenance 839.0  cpumem 2.72   gpumem 4.19   augment 0.179
tick 32    kimg 128.0    time 1d 00h 29m   sec/tick 1567.3  sec/kimg 391.81  maintenance 0.2    cpumem 2.72   gpumem 4.22   augment 0.181
tick 33    kimg 132.0    time 1d 09h 02m   sec/tick 1632.0  sec/kimg 407.99  maintenance 29146.9 cpumem 2.72   gpumem 4.21   augment 0.184
tick 34    kimg 136.0    time 1d 09h 29m   sec/tick 1640.3  sec/kimg 410.07  maintenance 0.2    cpumem 2.72   gpumem 4.29   augment 0.186
tick 35    kimg 140.0    time 1d 09h 56m   sec/tick 1635.6  sec/kimg 408.90  maintenance 0.2    cpumem 2.72   gpumem 4.24   augment 0.189
tick 36    kimg 144.0    time 1d 10h 24m   sec/tick 1642.6  sec/kimg 410.66  maintenance 0.2    cpumem 2.72   gpumem 4.23   augment 0.191
tick 37    kimg 148.0    time 1d 10h 50m   sec/tick 1591.9  sec/kimg 397.97  maintenance 0.2    cpumem 2.72   gpumem 4.28   augment 0.194
tick 38    kimg 152.0    time 1d 11h 18m   sec/tick 1647.1  sec/kimg 411.78  maintenance 0.2    cpumem 2.72   gpumem 4.22   augment 0.198
tick 39    kimg 156.0    time 1d 11h 45m   sec/tick 1659.0  sec/kimg 414.76  maintenance 0.2    cpumem 2.72   gpumem 4.29   augment 0.202
tick 40    kimg 160.0    time 1d 12h 13m   sec/tick 1641.5  sec/kimg 410.39  maintenance 0.2    cpumem 2.73   gpumem 4.23   augment 0.208
```





Q
A