

DCGAN, PGGAN, StyleGAN

DCGAN, PGGAN, StyleGAN 논문리뷰

2022.10.18 학부연구소모임 우아라

1. **DCGAN의 경우**, 기존 GAN 에서 fully-connected layer 구조를 사용하지 않고 convolutional and deconvolutional layer로 대체했다는 구조를 이해하고, 한계점으로 (1) 고해상도 영상을 생성하기 어려움, (2) min-max problem을 해결하는 과정에서 학습이 불안정함 (3) GAN에서 BCE loss를 사용하면서 생성자가 유사 영상만 만들어 내는 model collapse (모드 붕괴) 현상 발생 등이 있다.
2. **PGGAN의 경우**, 점진적으로 layer를 추가하여 low resolution에서 high resolution 영상을 생성가능하게 하여 1024x1024 영상까지 생성하게 함. 점진적으로 layer를 추가하여 학습하기 때문에 학습이 안정적임. 고해상도 영상을 상대적으로 안정적으로 생성할 수 있는 장점이 있는 반면 원하는 특정 feature에 대한 생성은 어려워서 style transfer에 기반한 generator 구조를 제시한게 StyleGAN 이 됨.
3. **StyleGAN의 경우**, 최종 생성된 영상을 style들의 조합으로 생각해서 generator의 각 layer 마다 style을 입히는 방식으로 영상을 만들어 내는 것으로 초기 layer는 coarse feature를 만들어내고 뒤로 갈수록 fine detail을 만드는 등 visual attribute를 조절할 수 있다는게 가장 큰 차이점임.
4. 위 사항을 참고하여 의료영상 분류 문제에서 데이터증강을 위해 사용한 DCGAN, PGGAN 에서의 한계점을 기재해 놓은 것이 있으면 해당 사항이 함께 논문에 기재되는게 좋단다.
5. 코드리뷰를 해보겠다는 사항은 1x512 input latent vector를 입력받아 mapping network (fully-connected layer 8개 통과)에 넣어서 disentanglement 한 intermediate latent vector W를 만들어내고, 각기 다른 w 벡터를 AdaIN layer에 넣어서 style을 입히는데 이 때 세부 사항들을 변경시키기 위해서 noise를 추가한다고 생각하면 되고 최종으로 PGGAN과 동일하게 1024x1024 영상을 생성함.

DCGAN

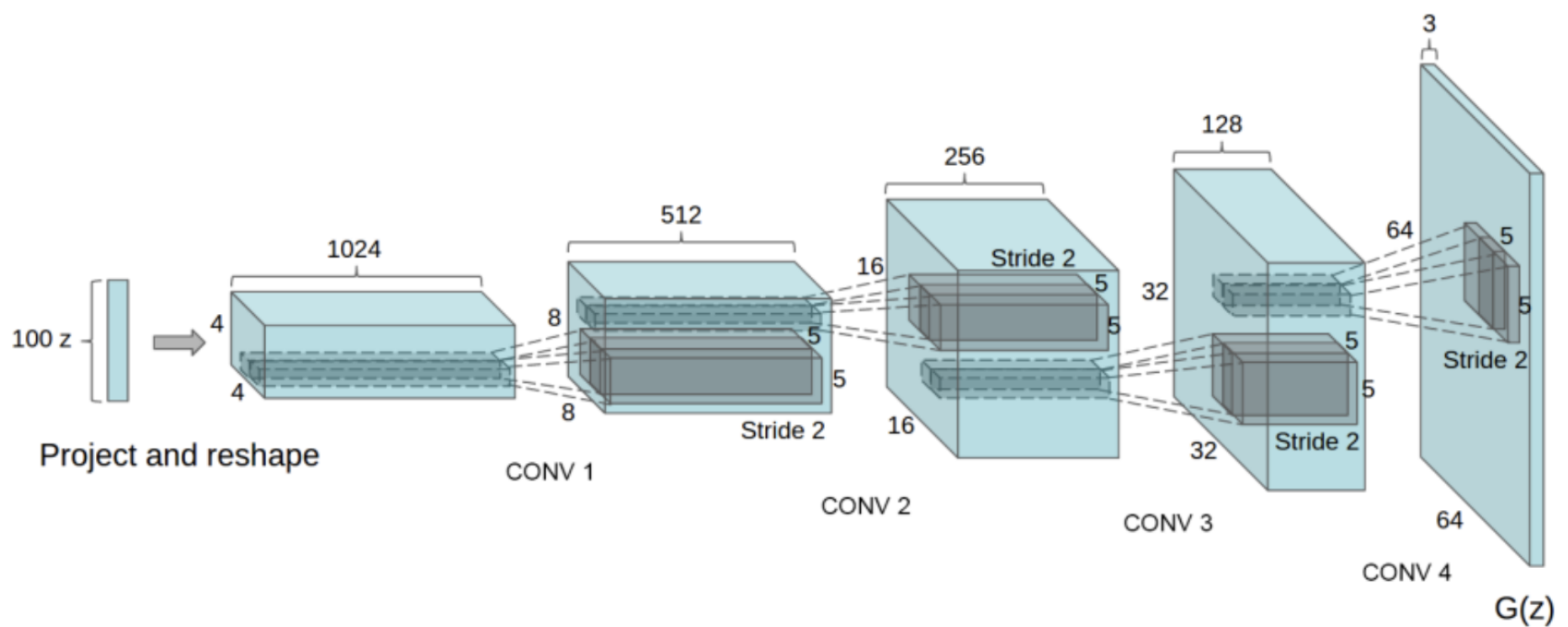
[Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks]

 <https://arxiv.org/pdf/1511.06434.pdf>

DCGAN?

- 기존 GAN의 이미지 퀄리티를 높임
- Black-box method 문제를 어느정도 해결
- Generator가 단순 기억으로 generate 하지 않는다는 것을 보여줌
- z의 미세한 변동에 따른 generate 결과가 연속적으로 부드럽게 이루어져야 한다. (walking in the latent space)
- 벡터 연산 가능

DCGAN 구조



Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

- pooling layer 대신 **stride & fractional-stride convolution layer** 사용
- Generator와 Discriminator 에 **Batch normalization layer** 사용
 - 깊은 신경망일수록 같은 Input 값을 갖더라도, 가중치가 조금만 달라지면 완전히 다른 값을 얻을 수 있다.
 - 이를 해결하기 위해, 각 layer에 배치 정규화 과정을 추가해준다면, 가중치의 차이를 완화하여 보다 안정적인 학습이 이루어질 수 있다.
- Fully connected layer 제거
 - 파라미터를 너무 많이 잡아먹어 계산량이 늘어나기 때문
 - 3차원 이미지를 1차원으로 평탄화하면 공간 정보가 소실되기 때문
- Generator에 **ReLU** 사용 + 마지막에 **Tanh** 사용
 - ReLU : 0보다 작으면 0, 0보다 크면 그대로 출력 (Gradient vanishing 현상 해결)
- Discriminator에 **Leaky ReLU** 사용 + 마지막에 **Sigmoid** 사용

DCGAN 한계

- 고해상도 영상을 생성하기 어려움,
- min-max problem을 해결하는 과정에서 학습이 불안정함
- GAN에서 BCE loss를 사용하면서 생성자가 유사 영상만 만들어 내는 model collapse (모드 붕괴) 현상 발생

PGGAN(=ProGAN)

[PROGRESSIVE GROWING OF GANS FOR IMPROVED QUALITY, STABILITY, AND VARIATION]

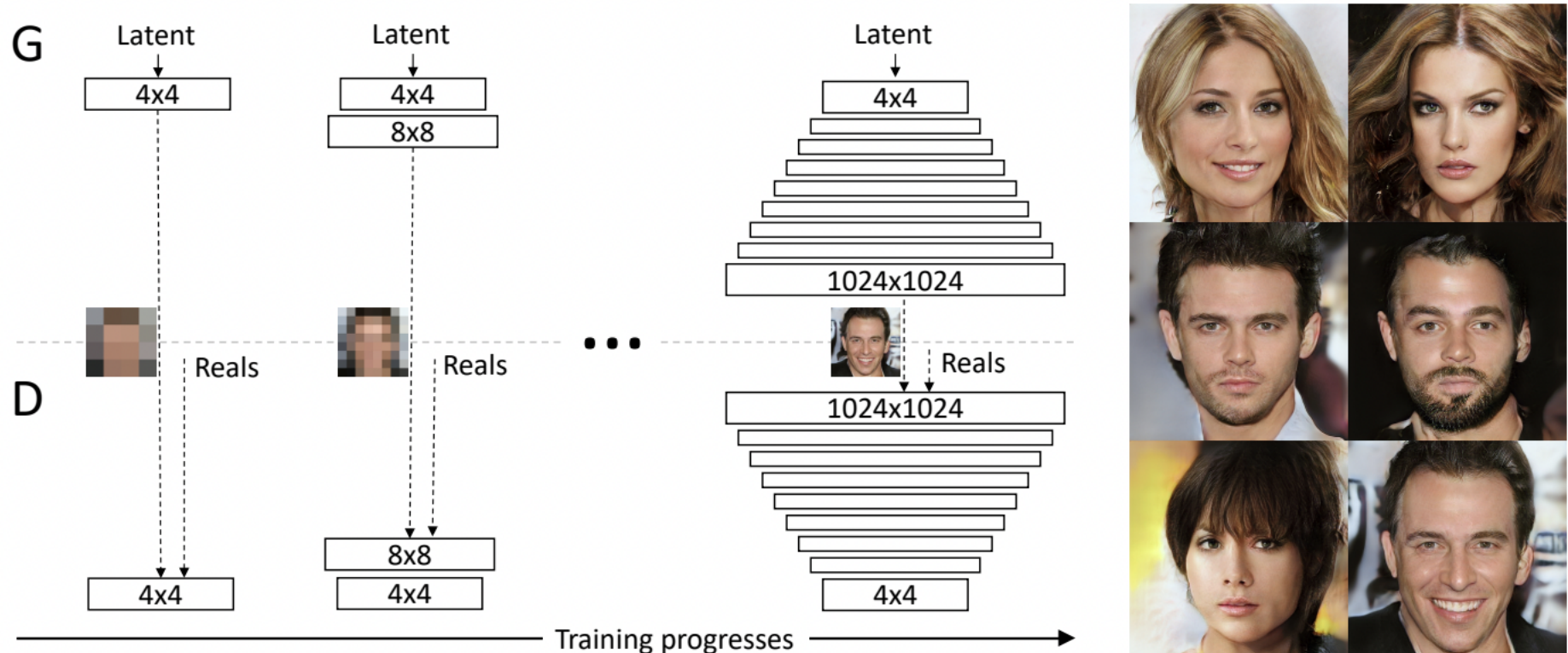
PGGAN?

- 그 전의 GAN들은 고해상도 이미지를 만드는 것이 매우 힘들었는데, 새로운 접근 방법으로 **고해상도 이미지 생성**에 성공
- 학습이 안정적(**WGAN-GP Loss** 적용)
- 학습 시간 절약

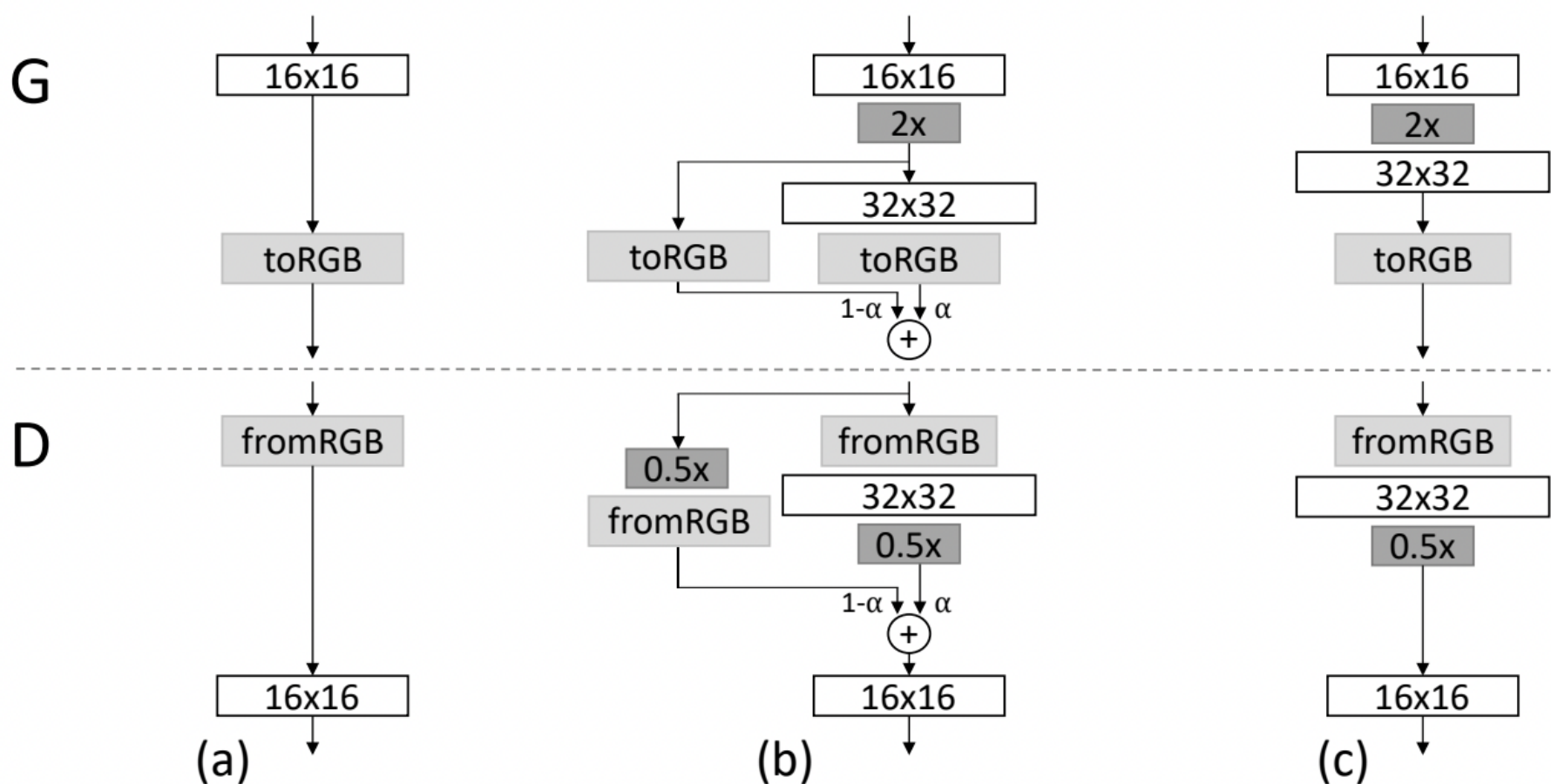
기존 GAN들이 고해상도 이미지를 만들기 어려웠던 이유

- High Resolution 일수록 Discriminator는 Generator가 생성한 Image가 Fake Image인지 아닌지를 구분하기가 쉬워진다.
- 고해상도 이미지를 잘 만든다고 해도, 메모리 제약 조건 때문에 mini-batch size를 줄여야 한다. batch size를 줄이면 학습과정 중 학습이 불안정해지는 현상이 발생한다.

PGGAN 구조



- 점진적으로 Generator와 Discriminator를 키운다. (Low resolution 에서 High resolution으로 키우기 위해서 layer를 점진적으로 add해주는 것)
 - 점진적으로 layer를 add 할 때의 장점
Image Distribution(각 이미지마다 분포가 다 다름)에서 Large scale 구조(전반적인 전체 데이터 형태)를 먼저 발견하도록 도움을 준다.
점차 레이어를 쌓아 올라갈수록 세부적인 Feature들을 보면서 학습을 진행한다.
- 새로 추가하는 레이어를 smooth하게, fade in 하게 넣어준다
(이미 잘 학습된 이전 단계의 layer(n-1 layer)들에 대한 sudden shock를 방지)



- 2x : 이미지의 해상도 두 배 늘림
- toRGB : Feature Vector를 RGB Color로 만듦

PGGAN 한계

- 원하는 특정 feature에 대한 생성은 어려움 (이미지의 특징 제어가 어려움)
- 이미지의 특징이 잘 분리되어 있지 않기 때문에 다른 특징까지 개입되는 문제 발생

(Generator에 latent vector z 가 바로 입력되게 때문에 entangle하게 되어서 style의 변형이 불가능 하다.)

StyleGAN

코드리뷰에서는 input vector의 크기, 출력영상의 크기, AdaIN의 역할, noise의 역할 및 noise를 어떻게 주었는지 등에 대해서 확인 및 질문

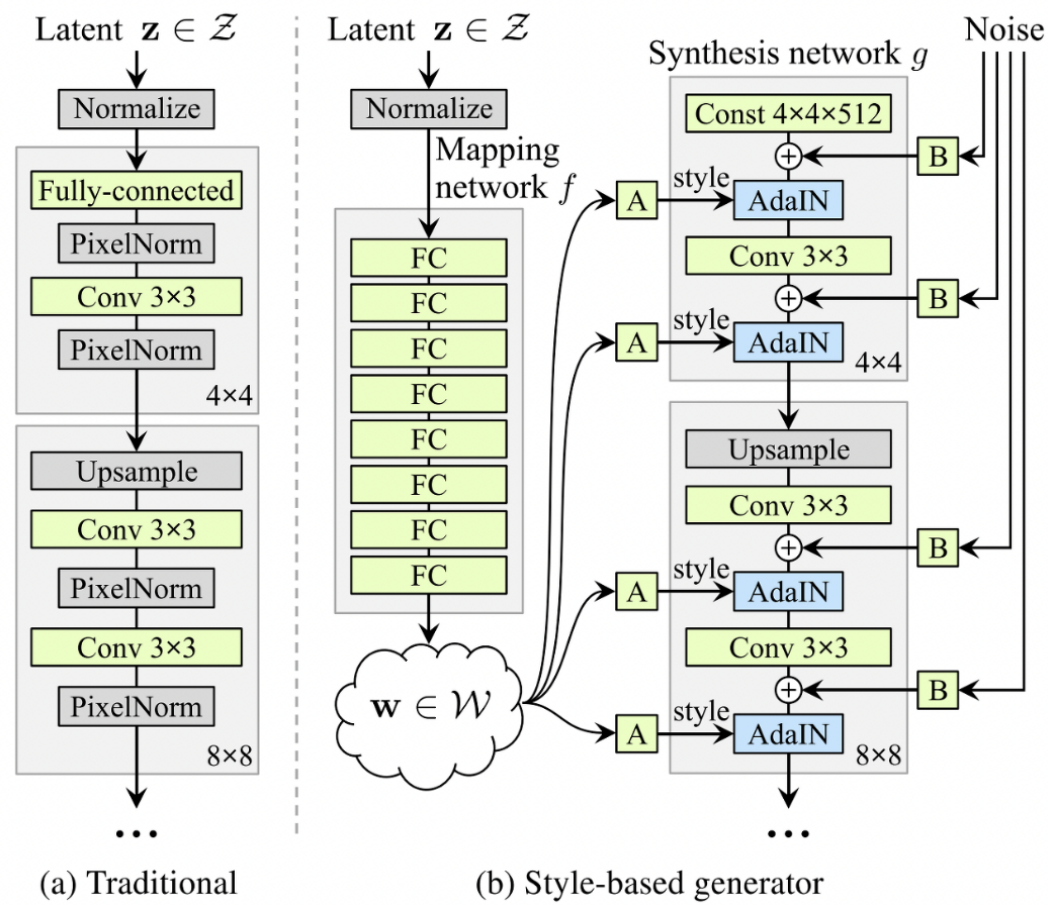
[A Style-Based Generator Architecture for Generative Adversarial Networks]

<https://arxiv.org/pdf/1812.04948.pdf>

StyleGAN?

- 고화질 이미지 생성에 적합한 아키텍처 제안
 - PGGAN 베이스라인 아키텍처의 성능 향상
 - Disentanglement 특성 향상 (다양한 특징들이 잘 분리되어 있는 것)
- 특징 제어가 어려웠던 PGGAN의 단점 개선

StyleGAN 구조



- **Style Mixing (Mixing regularization)**

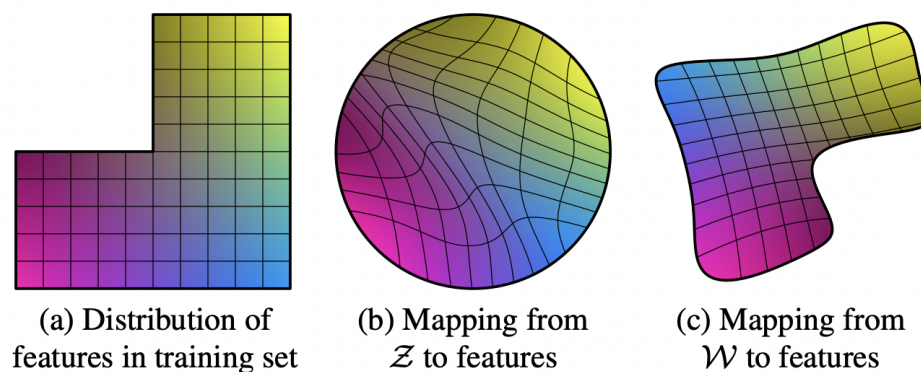
latent z 를 바로 학습시키는 것이 아니라, multi latent $z(w_1, w_2)$ 를 이용하여 w 벡터를 만든 후 학습시킨다.

→ 다양한 style이 섞여서 synthesis network 학습이 된다.

인접한 layer 간의 style 상관관계를 줄인다.

- **Mapping Network**

가우시안 분포에서 샘플링한 z 벡터를 직접 사용하지 않고, 매핑 네트워크를 거쳐 w 벡터로 바꾼 후 이미지를 만들면 (b) 처럼 특정 분포를 따라야 하는 제한이 사라지기 때문에 (c) 처럼 각각의 특징을 훨씬 잘 분리할 수 있다.



- **AdaIN(Adaptive Instance Normalization)**

layer을 거치면 학습이 불안정해져서 normalization을 각 layer에 추가

다른 원하는 데이터로부터 style 정보를 가져와 현재 이미지 style에 적용할 수 있음 (학습시킬 추가적인 파라미터가 필요하지 않고, 성능이 더 좋음)

+) Instance Normalization

채널마다 정규화를 수행했던 Batch Normalization과 달리, 하나의 이미지에 대해 정규화를 수행

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

▼ AdaIN code

```
def AdaIN(x):
    #Normalize x[0] (image representation)
    mean = K.mean(x[0], axis = [0,1], keepdims = True)
```

```
std = K.std(x[0], axis = [0,1], keepdims = True) + 1e-7
y = (x[0] - mean) / std

#Reshape scale and bias parameters
pool_shape = [-1, 1, 1, y.shape[-1]]
scale = K.reshape(x[1], pool_shape)
bias = K.reshape(x[2], pool_shape)

#Multiply by x[1] (GAMMA) and add x[2] (BETA)
return y * scale + bias
```

- **Noise**

다양한 확률적인 측면 컨트롤 (ex. 주근깨, 머리카락 등)

별도의 affine transformation인 B를 거쳐 각각의 feature map 마다 noise가 적용될 수 있도록 한다.

StyleGAN 한계

물방울 형태의 blob 들이 관측



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.