

# Analyzing and Improving the Image Quality of StyleGAN

arras, Tero, et al. "Analyzing and improving the image quality of stylegan." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020.

## Abstract

### 1. Introduction

기존 stylegan에서 characteristic artifacts(물방울 모양의 반점)가 발견됨

그 이유는

1. generator가 자체의 구조적 결함 문제를 회피하기 위해서 생성함.  
artifacts를 제거하기 위해 generator의 normalization을 다시 설계함.
2. progressive growing으로 인해 저해상도에서 생성된 artifact가 고해상도에서도 유지됨.  
progressive growing을 유지하면서 network를 재설계함.

### 2. Removing normalization artifacts(droplet)



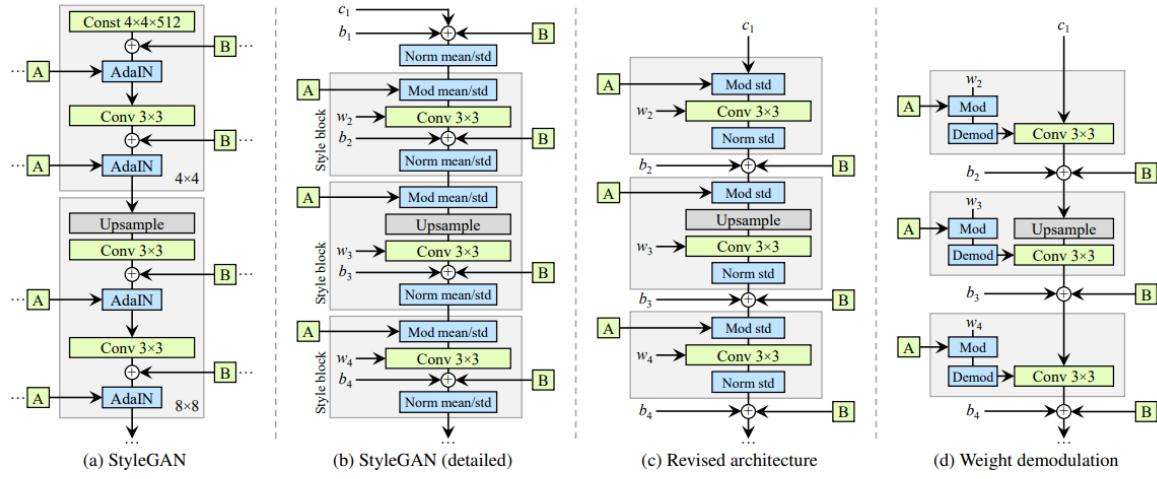
Figure 1. Instance normalization causes water droplet -like artifacts in StyleGAN images. These are not always obvious in the generated images, but if we look at the activations inside the generator network, the problem is always there, in all feature maps starting from the 64x64 resolution. It is a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

위 사진과 같이 stylegan1에서 생성된 이미지중, artifacts들이 생성되었고, 이러한 artifacts들은 64x64 resolution을 생성할 때 시작된 것이 관찰되었다. 64x64에서 생성된 artifacts들은 그보다 고해상도로 갈수록 점점 강해졌다고 한다. 하지만 이러한 artifacts들은 discriminator에서 발견하고 수정되었어야 하지만 그렇지 못했다고 함.

이 문제의 원인으로는 AdaIN(Adaptive Instance Normalization)이 각 feature map을 style의 평균과 분산으로 정규화하면서 feature 사이의 크기차이에서 발견되는 정보들이 사라질 수 있기 때문이라고 한다.

generator의 normalization을 제거하였더니 artifacts가 사라진 것을 발견하였음.

#### 2.1 Generator architecture revisited



위 그림에서 (a)와 (b)는 기존 styleGAN의 구조를 보여주고 있다. 이 구조에서 한 block을 살펴보면 styleGAN에서 AdaIN 적용 방법(normalization, modulation)을 볼 수 있는데, 이전 결과에서 style에 대한 통계 정보(평균, 표준편차)를 입혀줌으로 style을 변경할 수 있는 Mod mean/std block => convolution => Noise => Norm mean/std로 이루어져 있다.

이후 그림 (c)와 같이 bias와 noise의 적용을 style block 외부인 normalization 뒤로 이동(normalization 이전에 noise를 적용하면 noise도 같이 정규화 되기 때문에 영향이 적어짐)하고, normalization과 modulation에서 평균을 사용하지 않는 방식으로 단순화 했음에도 충분한 결과가 나올 수 있다는 것을 관찰할 수 있었다고 한다.

## 2.2 Instance normalization revisited

기존 styleGAN에서는 AdaIN이 모든 feature map을 normalize 하지만 분산만 사용하면 feature map을 normalize하는 것보다 convolution의 weight를 normalize함으로 같은 효과를 얻을 수 있다.

$$1. \text{AdaIN}(x_i, y) = \sigma(y) \left( \frac{x - \mu(x)}{\sigma(x)} \right) + \mu(y)$$

$$2. \text{without mean} = \sigma(y) \left( \frac{x}{\sigma(x)} \right)$$

위는 기존의 AdaIN과 평균을 사용하지 않을 때의 normalization이다. 이때 x는 convolution을 통해 나온 feature map으로 볼 수 있는데, 2번 식에서 볼 수 있다시피 단순 곱의 형태로 진행되고, convolution은 weighted sum의 형태로 되어 있다. 그렇기 때문에 style에 대한 표준편차 정보는 feature map 직접 적용 하는 것이 아니라, 가중치에만 곱해서 적용되어도 같은 효과를 얻을 수 있다.

$$y = \sum_{i,j} w_{i,j} x_{i,j}, \sigma y$$

$$\sigma y = \sum_{i,j} \sigma w_{i,j} x_{i,j}$$

이를 바탕으로 그림 (d)에서의 modulation과 demodulation을 구성할 수 있다.

$$1. w'_{ijk} = s_i \cdot w_{ijk}$$

$$2. \sigma_j = \sqrt{\sum_{i,k} w'_{ijk}^2}$$

$$3. w''_{ijk} = w'_{ijk} / \sqrt{\sum_{i,k} w'_{ijk}^2 + \epsilon}$$

1번 식은 modulation을 feature map이 아닌 convolution weights에 직접 적용하는 modulation이라고 볼 수 있다.

하지만 instance normalization은 output feature map의 통계량에서 scaling의 영향을 제거하는 것이 목적이다. 그렇기 때문에 output feature map의 분산도 1이 되도록 scaling을 해야한다.

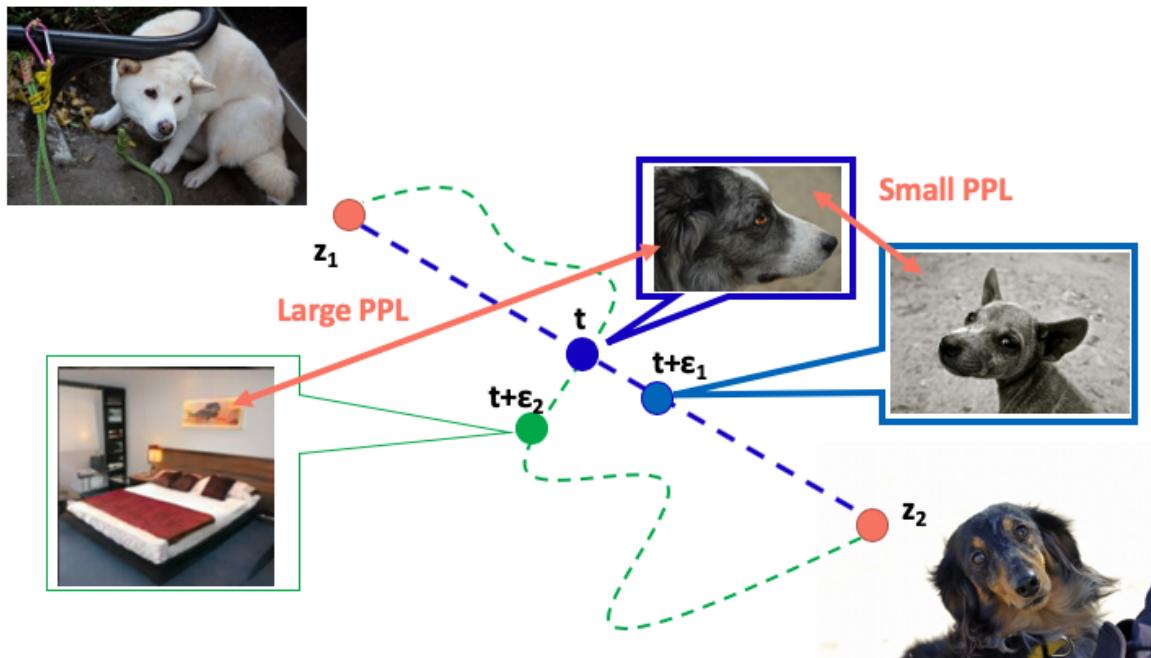
2번 식에서는 input의 표준편차가 i.i.d random variables일 때 output feature map의 표준편차의 기댓값이라고 볼 수 있는데, 이를 다시 나눠줌으로 최종적으로 목표인 output feature map의 표준편차를  $1/(\sigma)$ 로 맞추어 줄 수 있다(3.). 이 때 zero division error 때문에 epsilon값을 더해준다.

이를 통해서 물방울 모양의 artifacts(droplet artifacts)를 제거할 수 있었다고 한다.

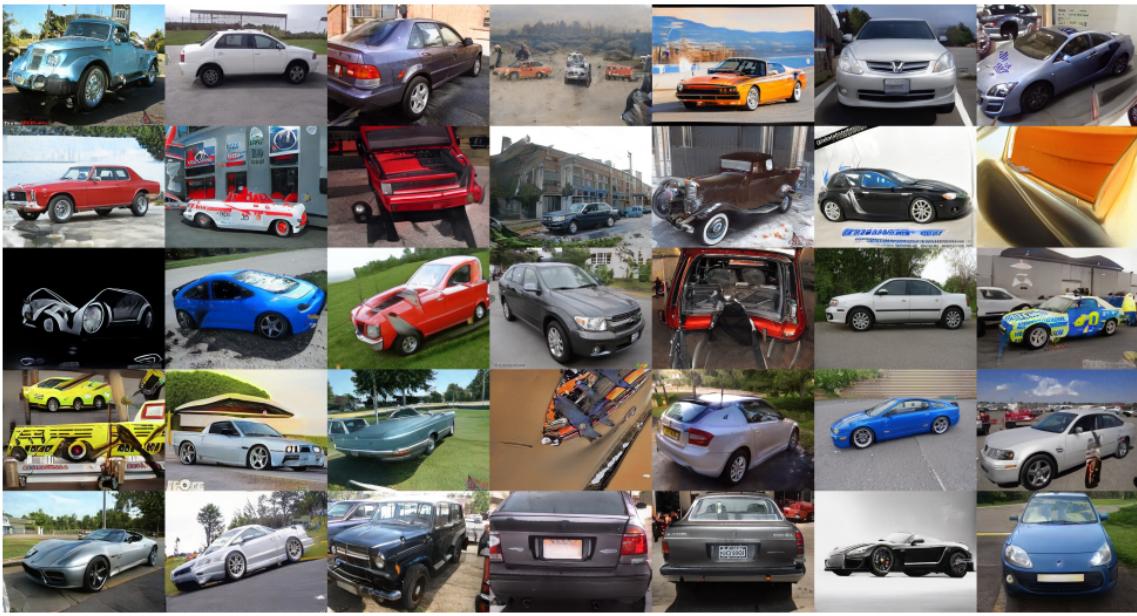
### 3. Image quality and generator smoothness

#### Perceptual Path Length

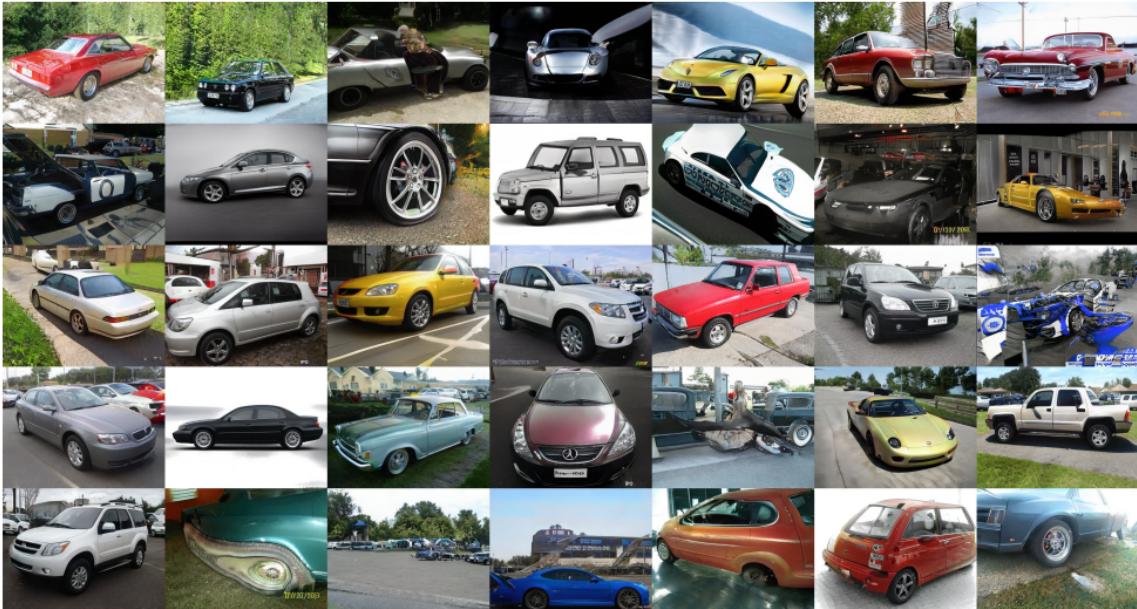
$$l_{\mathcal{W}} = \mathbb{E} \left[ \frac{1}{\epsilon^2} d(G(\text{lerp}(\mathbf{z}_1, \mathbf{z}_2; t)), G(\text{lerp}(\mathbf{z}_1, \mathbf{z}_2; t + \epsilon)), ) \right]$$



기존의 gan 평가 지표에는 FID, Precision & Recall score로 사용되었지만, FID나 P&R이 같더라도 PPL(perceptual path length)에 따라서 image quality가 달라질 수 있다. PPL이 낮을 수록 대체로 더 좋은 결과를 내는 것을 확인할 수 있다.



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

FID나 P&R같은 경우는 neural network를 사용하여 측정하기 때문에 대략적인 형태만 가지고 사물인지 판단하는 사람과는 다르게, 형태가 아닌 texture에 기반하여 평가되기 때문이라고 한다. 하지만 PPL의 경우는 latent vector가 조금 바뀌어도 artifact가 갑자기 생성된다면 더 높은 PPL score가 나오기 때문에 PPL이 낮을 수록 전반적으로 더 좋은 결과를 보여줄 수 있다고 한다.

### Lazy regularization

Regularization을 매 iteration마다 적용시킨다면 computational resource가 너무 많이 필요하기 때문에 매번 적용하는게 아니라 가끔 적용함으로 computational resource를 아낄 수 있다.

### Path length regularization

PPL을 줄이는 방법으로는 style에 대한 vector  $w$ 가 일정 크기만큼 증가하였을 때, 생성되는 이미지 또한 일정하게 변하게 해야한다.

$$\begin{aligned} g(w) : \mathcal{W} &\mapsto \mathcal{Y} \\ \mathbf{J}_w = \delta g(\mathbf{w})/\delta(\mathbf{w}) \\ \mathbb{E}_{\mathbf{w}, \mathbf{y} \sim \mathcal{N}(0, \mathbf{I})} \left( \|\mathbf{J}_w^T \mathbf{y}\|_2 - a \right)^2 \end{aligned}$$

위처럼  $w$ 로 이미지를 생성하는 generator를  $g(w)$ 라 할 때, 이에 대한 자코비언 행렬  $J_w$ 로 구성되어 있다.

이때  $J_w^T y$ 는  $y$ 일 때 생성되는 이미지의 변화량 즉, 미분값으로 추정될 수 있는데, 이를 a(학습 동안 변화량의 평균)값에 근사시킴으로 Path length regularization을 할 수 있다고 한다.

이를 통해서 ppl을 줄이도록 하는 정규화를 할 수 있었다고 한다.

#### 4. Progressive growing revisited(phase artifacts)

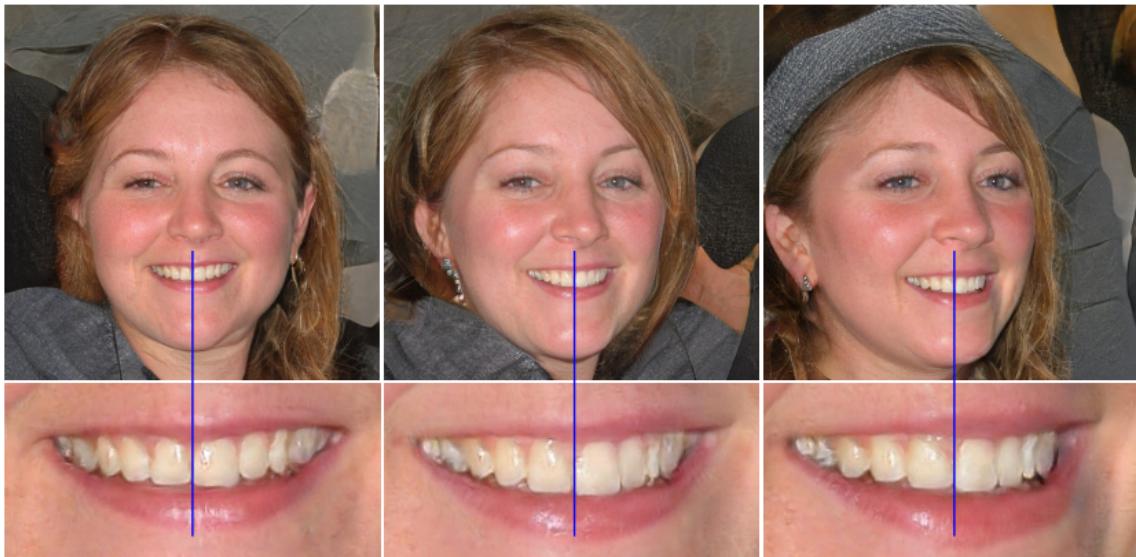
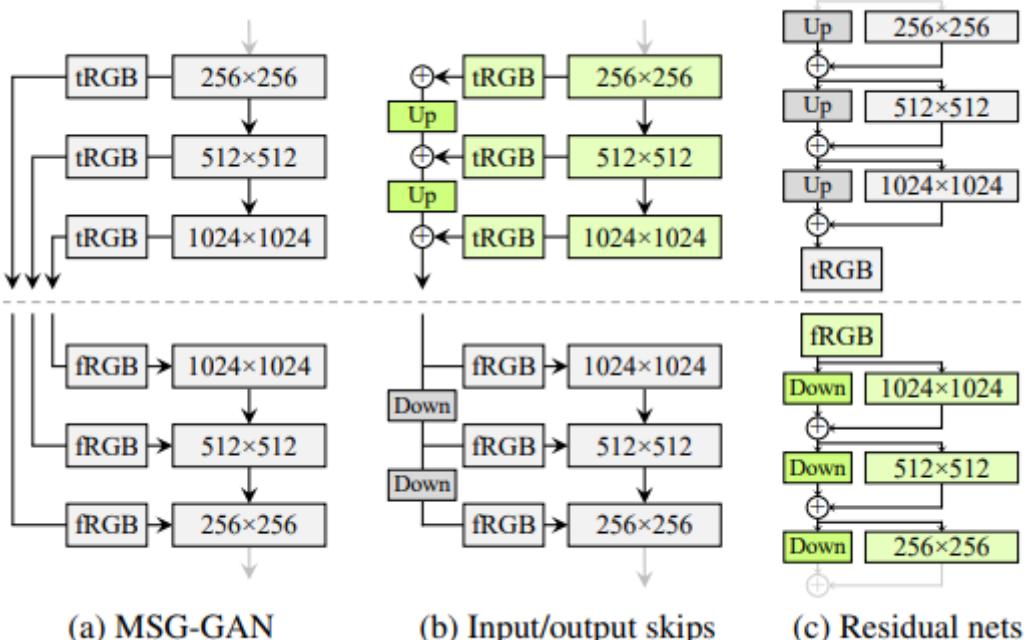


Figure 6. Progressive growing leads to “phase” artifacts. In this example the teeth do not follow the pose but stay aligned to the camera, as indicated by the blue line.

phase artifacts란 위처럼 사람의 얼굴 방향이 돌아감에도 치아의 방향은 변하지 않는 문제를 말한다.

이는 high resolution을 생성하는데 효과적이었던 progressive growing의 문제로 보이는데, 특정 layer에서 특징들에 대해 강하게 학습되어 위치를 고정하게 된다.

##### 4.4. Alternative network architectures



그래서 progressive growing 대신 단순 feed forward 방식으로 고해상도의 이미지를 생성할 수 있는 다른 구조들을 살펴본다.

(a) 는 MSG-GAN으로 generator의 skip connection이 discriminator의 같은 resolution에 적용되는 방식으로 U-net과 비슷한 구조를 가지고 있다.

(b) 는 각각의 다른 resolution의 영상들을 upsampling, downsampling을 하여 이미지를 생성하는 방식이다.

(c) 는 각 resolution에 residual connection을 적용시켜 이미지를 생성한다.

(a) 는 generator와 discriminator가 각 resolution마다 skip connection으로 묶여 있어, generator와 discriminator의 구조를 같게 해주어야 하지만, (b), (c)의 경우는 upsampling이나 downsampling이 한 block으로 이루어져 본 논문의 최종 아키텍쳐와 같이 generator를 (b)로 사용하고, discriminator를 (c)로 사용할 수도 있다.

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	<b>3.31</b>	<b>125</b>
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	<b>316</b>	3.19	471
G residual	3.93	981	3.40	667	<b>2.66</b>	645

Table 2. Comparison of generator and discriminator architectures without progressive growing. The combination of generator with output skips and residual discriminator corresponds to configuration E in the main result table.

#### 4.2 Resolution usage

progressive growing을 통해 저자가 얻고 싶었던 특성은 generator가 처음엔 저해상도에 초점을 맞추고 생성되는 이미지의 세부적인 요소를 고해상도에서 학습하는 것이다. 이 특성이 새로 구성한 generator에도 나타나는지 알아보기 위해서는 학습 시간에 따라 generator가 어떤 해상도에 강한 영향을 끼치는지에 대해 정량화할 필요가 있다.

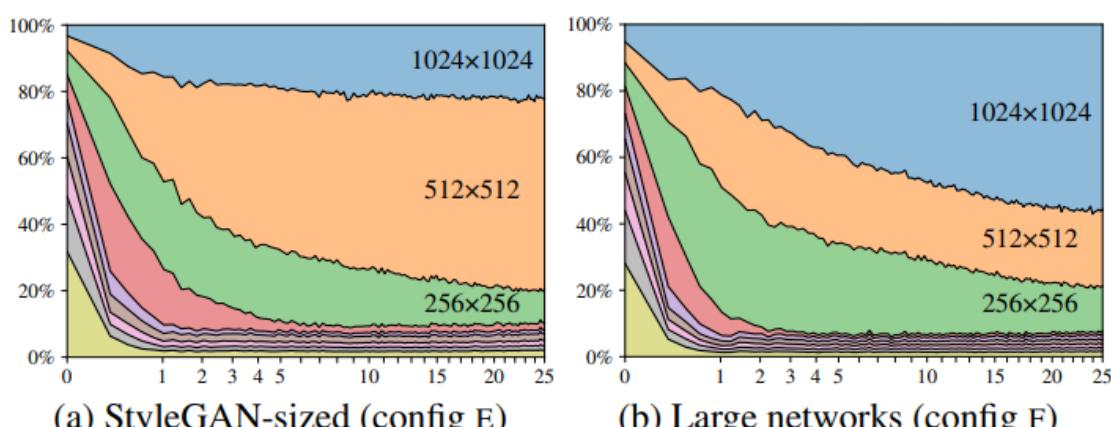


그림 (a)의 config E는 generator를 input/output skips로, discriminator를 residual로 구성한 network이며, (b)에서의 config F는 E에서 network의 size를 증가시킨 network이다.

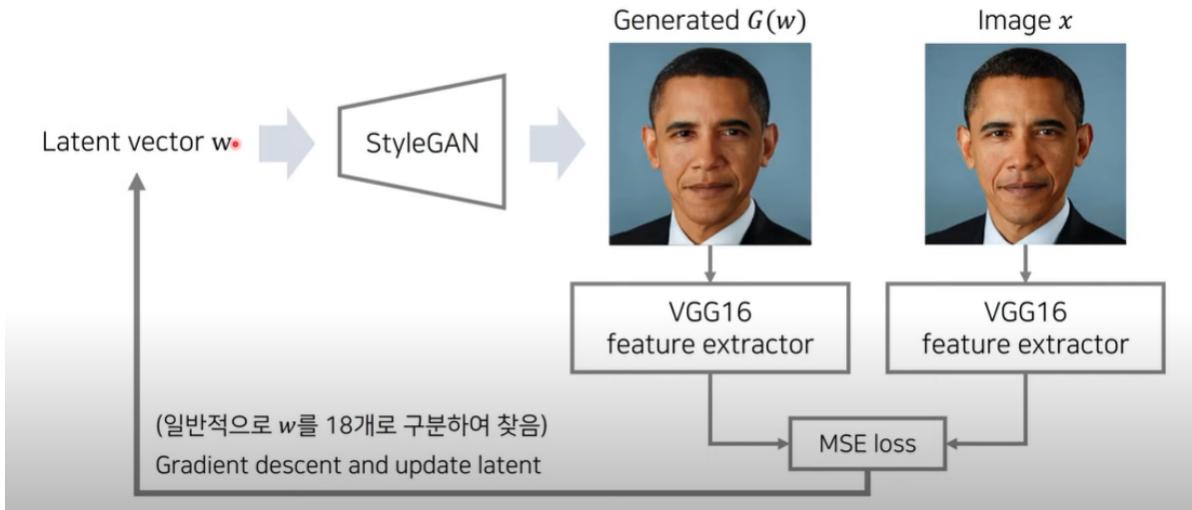
그림 (a)에서는  $512 \times 512$ 까지는 generator가 학습이 진행됨에 따라 더 높은 해상도에 영향을 받음을 확인하는 것을 보아 이전의 progressive growing에서의 특성을 가졌다고 볼 수 있다.

하지만  $1024 \times 1024$ 에서는 영향이 점점 늘지 않는 것을 확인할 수 있는데, 저자가 생성된 이미지를 살펴보았을 때, 생성된  $1024 \times 1024$ 의 이미지는  $512 \times 512$ 에서 좀 더 선명화되기만 했었다고 한다.

이런 문제는 model의 capacity 문제로 보아  $64 \sim 1024$ 의 feature map을 2배 늘렸고, 이에 대한 결과로 그림 (b)와 같은 결과를 얻을 수 있다.

## 5. Projection of images to latent space

projection이란 generator를 역변환하여 image를 다시 latent code  $z$ 를 추정하는 방법으로 이를 활용하면 각 특징(style)에 맞는  $w$ 를 추정할 수 있을 것이다.

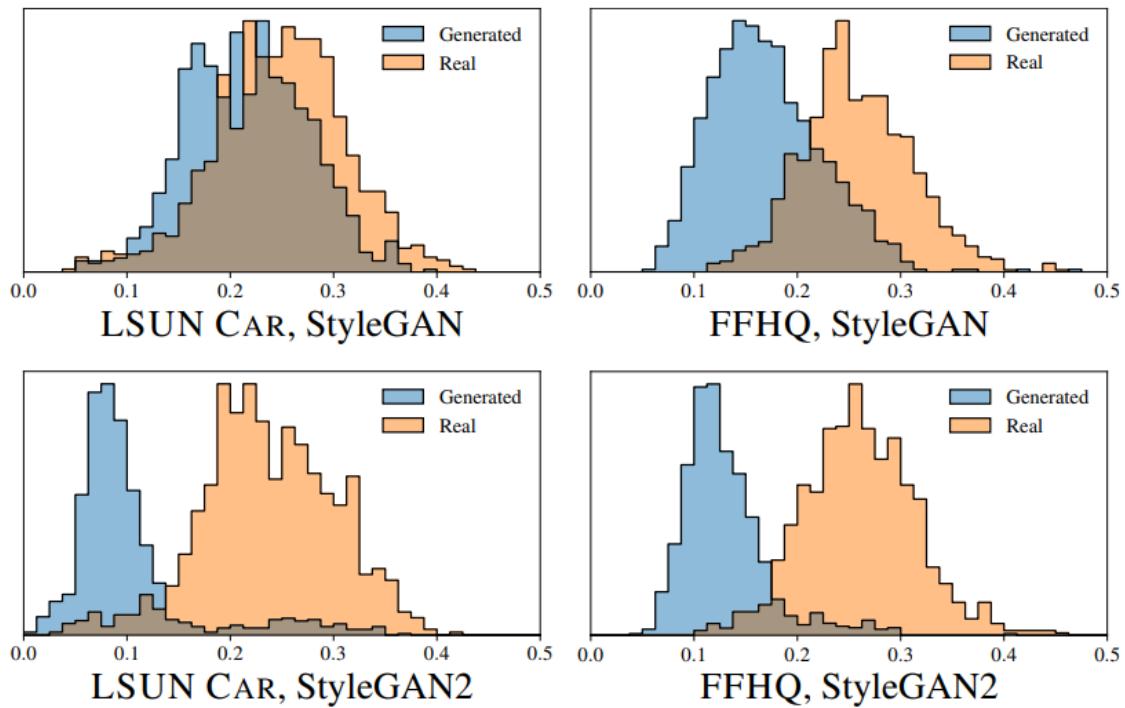


위처럼 생성된 이미지와 실제 이미지에서 vgg16를 사용하여 feature를 추출하고 이와의 mse loss를 통해  $w$ 를 업데이트하는 방식으로 원하는 이미지에 대한 latent vector인  $w$ 를 추정할 수 있다.



Figure 9. Example images and their projected and re-synthesized counterparts. For each configuration, top row shows the target images and bottom row shows the synthesis of the corresponding projected latent vector and noise inputs. With the baseline StyleGAN, projection often finds a reasonably close match for generated images, but especially the backgrounds differ from the originals. The images generated using StyleGAN2 can be projected almost perfectly back into generator inputs, while projected real images (from the training set) show clear differences to the originals, as expected. All tests were done using the same projection method and hyperparameters.

이와 같은 방법을 수행하여 해당 이미지에 맞는 latent vector  $w$ 를 추정하고 이를 통해 생성한 결과이다. 여기서 특이점으로는 생성된 이미지에 대해서 수행한 결과는 거의 완전히 같은 결과를 보임을 확인할 수 있다.



$$D_{LPIPS}[x, g(\tilde{g}^{-1}(x))]$$

이는 위처럼 LPIPS distance의 분포를 보여주는 결과로 stylegan에 비해서 stylegan2가 실제 이미지와, 생성된 이미지로 수행한 결과가 잘 분리됨을 확인할 수 있고, 더 0에 가까운(원래 이미지와 유사한) 결과를 보여준다.

이를 통해서 generator의 projection을 통한 latent vector w를 찾는 성능이 개선됨을 보이고 있다.