StyleGAN2 논문리뷰

https://arxiv.org/pdf/1912.04958v2.pdf

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten

23 Mar 2020

<Contribution>

- original StyleGAN과 마찬가지로 W space에서의 analysis를 진행한다.

  (이전 연구와 마찬가지로 Z → W 매핑 네트워크를 사용한다.)
- 기존 아키텍처의 문제점을 지적한다.
  - AdaIN style transfer → blob-like artifact 발생
  - Progressive growing → 얼굴의 특정 부분이 fixed position(고정된 위치)을 갖는 문제 발생 (phase artifact)

    (치아나 눈동자 등 일부 위치가 변경되지 않는 문제)
- 성능을 개선하기 위한 방법을 제안한다.
  - Path length **regularization**
    - A fixed-size step in W results in a fixed-magnitude change in the image (W space에서 고정된 만큼 latent vector를 를 이동시키면 그것이 이미지 space에서도 고정된 만큼의 변화를 일으킬 수 있도록 네트워크를 incluse 하는 방법인데, **regularization**을 사용하게 되면 .. ~~결과적으로 ~~
    - 결과적으로 latent interpolation에 따라서 이미지가 부드러운(smooth) 변화를 보이게 된다.
  - 이미지를 latent vector로 inversion하는 새로운 알고리즘을 제안한다.

<StyleGAN의 문제 1. Blob-like (Droplet) Artifacts>

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.

- 물방울(water droplet) 형태의 인공물(artifact)
  - Adaptive instance normalization(AdaIN)이 원인으로 추정됨
  - 생성자의 activation에서는 더욱 확실히 드러남

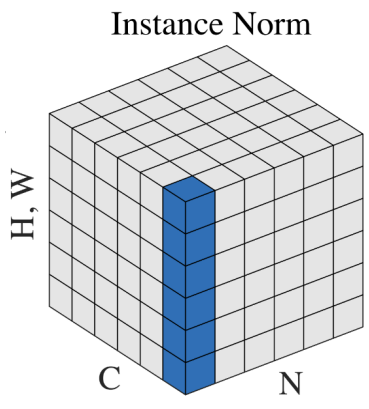
<StyleGAN 문제2. Phase Artifact>

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.

→ (사진에서 얼굴의 방향은 변하지만 치아의 위치는 비교적 변하지 않음)

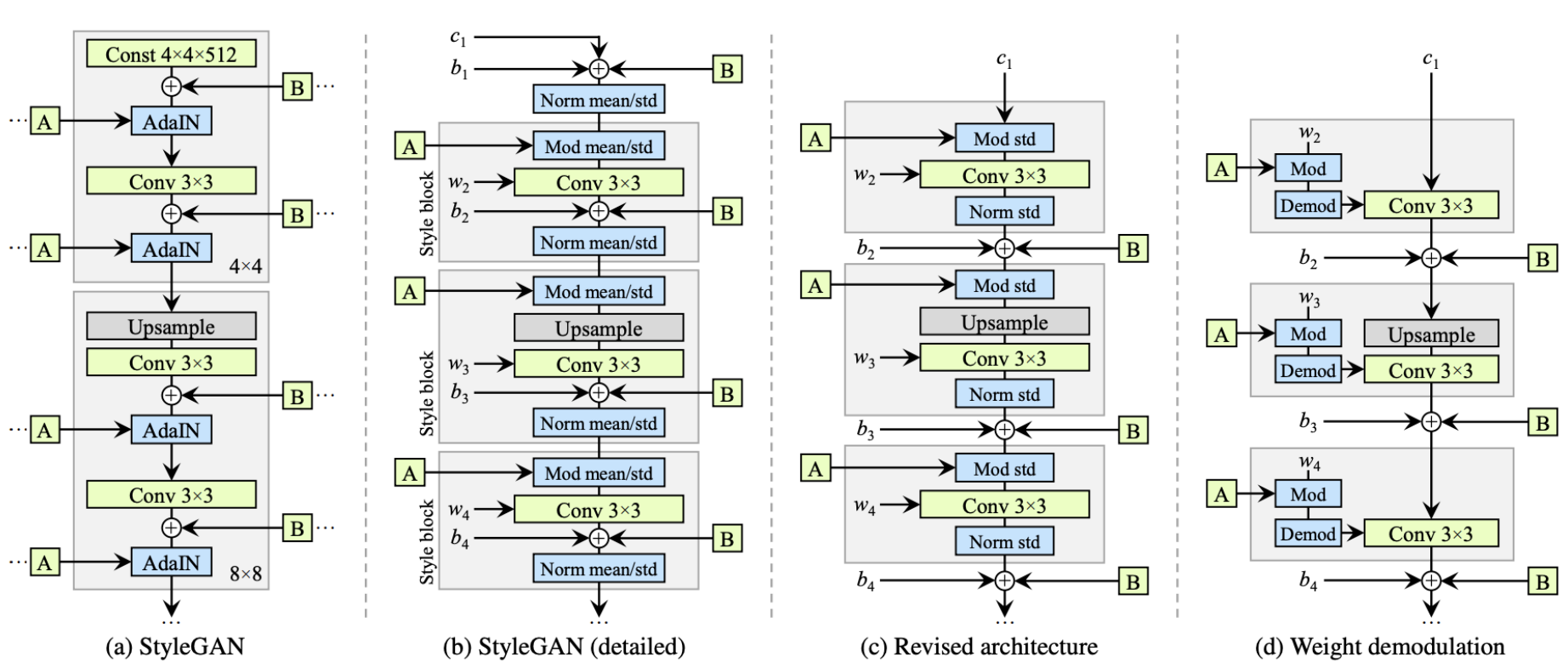
- Latent manipulation 과정에서 특정 요소가 고정되는 문제
- **Progressive growing이 원인**으로 추정됨
  - 각 resolution이 output resolution에 즉시 영향을 미침
    - 최대한 높은 frequency detail을 생성하고자 함

<StyleGAN 문제1 해결 - Removing Normalization Artifacts>



- AdaIN은 각 feature map의 mean/std를 개별적으로 정규화해 서로 연관된 feature들의 정보가 소실될 수 있다.

  Normalize mean/std → Modulate mean/std
- 또한 본 논문의 저자들은 standard deviation만 변경해도 충분하다는 것을 알아냈다.
  - 결과적으로 standard deviation만 변경하면서, 최대한 style block의 외부에서 feature map들의 값을 변경하고자 한다.



(b) :

Normalize → modulate 진행해서 feature map마다 statistics를 변경하는 방식으로 style 정보를 넣어서 이러한 블럭들을 거치면서 결과 이미지가 만들어질 수 있도록 한다. 즉, convolution 연산을 수행하고, noise를 반영하고, statistics를 변경하고, upsampling한 뒤에, 다시 convolution 연산 수행하고, noise 반영하고, statistics를 변경하고 .. 이러한 방식을 반복하는데, 이 때 기본적으로 본 논문에서는 mean값에 대해서는 건드리지 않고, standard deviation만 변경해도 기능상 충분하다는 것을 알아냈고, 결과적으로 본 논문에서는 feature map에 standard deviation만 변경하면서, 최대한 style block의 외부에서 feature map들의 값을 변경하고자 했다.

(c) :

즉, convolution 연산을 수행한 결과에 대해서 직접적으로 normalization을 진행하는 것이 아니라, 가능한 block의 외부에서 feature map들의 statistics들을 변경할 수 있도록 한다. 그래서 하나의 block이 있을 때, normalization을 수행한 뒤에 noize가 들어가는 형태로 순서를 바꾸었고, mean 값은 건드리지 않고 standard deviation 값만 변경하는 형태로 architecture를 수정했다. 다만 이렇게 하더라도 여전히 convolution 연산의 결과로 나온 feature map에 대해 직접적으로 statistics를 변경하기 때문에 서로 연관된 feature 들의 정보를 손상시킬 수 있다.

(d) :

더 나아가 본 논문에서는 convolution 연산을 거친 feature map에 대해 바로(직접적으로) modulation을 적용하지 않고, 대신에 (convolution 연산에 쓰이는) weight 값에 대해 modulation을 진행한다. 즉, convolution 연산을 수행할 때 커널을 쭉 슬라이딩 하면서 합성곱을 수행하는데, 그 때 곱하는 양, 즉 weight 값에 대해 modulation을 대신 진행한다.

즉, feature map에 대해 직접적으로 standard deviation 값을 바꾸는게 아니라 convolution에 사용되는 weight 값을 변경할 수 있도록 해서, feature map을 직접적으로 정규화하지 않고 특정 layer에서 feature map이 어떠한 statistics을 가질 것이다라고 예측해서 적절한 weight 값으로 convolution 연산을 수행하는 형태로 정규화의 기능을 대신 수행하도록 한다.

기본적으로 우리가 standard deviation 값을 바꾸기 위해서 modulation을 할 때 weight을 대신 scaling하는 방법을 사용한다.

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

49:49

https://www.youtube.com/watch?v=5UlTdngEdas

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

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

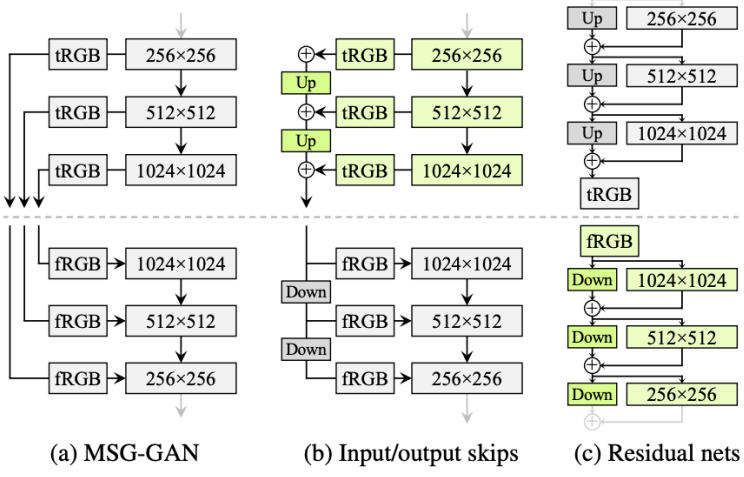


Figure 7. Three generator (above the dashed line) and discriminator architectures. Up and Down denote bilinear up and downsampling, respectively. In residual networks these also include 1×1 convolutions to adjust the number of feature maps. tRGB and fRGB convert between RGB and high-dimensional per-pixel data. Architectures used in configs E and F are shown in green.

Abstract

latent code -> image로 맵핑하는 데 더 좋은 영향을 주는 요소들

- **generator의 normalization** 부분을 수정했고,
- **progressive growing**(기존 stylegan에서 해상도를 한 step씩 올려가며 학습하는 것)을 **재고**하고,
- **generator를 정규화**했다.

출력물의 해상도를 얼마나 잘 활용하는지 시각화하여 수용량의 문제를 발견했고, 이에 따라 **추가적인 품질 향상을 위해 수용력이 더 큰 모델을 이용해 학습**

1. Introduction

stylegan이 생성한 이미지에서 발견되는, 인공적으로 보이는 **특징적인 결점이 발생하는 원인 2가지**

- **common blob-like artifacts, and find that the generator creates them to circumvent a design flaw in its architecture**
  - redesign the normalization used in the generator, which removes the artifacts
- 모양보단 텍스쳐에 집중하는 metric인 FID and P&R은 이미지의 품질을 정확하게 평가하기 어렵다.
  - latent space interpolation 품질을 측정하기 위해 고안된 PPL(Pereceptual Path Length)이 형태의 일관성이나 안정성과도 상관관계가 있다.
  - 위 사실에 근거해서, synthesis network를 정규화(regularize)해서 좀 더 mapping이 부드럽게 되게 했고, 품질에서 분명한 향상을 이룸
- original stylegan의 generator에 비해 path-length로 정규화된 stylegan2의 [projection : image -> W (latent Space)] 이 훨씬 더 성능이 좋음을 발견했다.
  - 이미지가 소스로부터 생성되기 쉽게 만듦

2. Removing normalization artifacts

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.

3. Image quality and generator smoothness

4. Progressive growing revisite

5. Projection of images to latent space