

StyleGAN

A Style-Based Generator Architecture for Generative Adversarial Networks (CVPR 2019)

Tero Karras, Samuli Laine, Timo Aila (NVIDIA)

[A Style-Based Generator Architecture for Generative Adversarial Networks \(CVPR 2019\)](#)

[Intro](#)

[Abstract](#)

[1. Introduction](#)

[2. Style-based generator](#)

[2.1. Quality of generated images](#)

[2.2. Prior art](#)

[3. Properties of the style-based generator](#)

[3.1. Style mixing](#)

[3.2. Stochastic variation](#)

[3.3. Separation of global effects from stochasticity](#)

[4. Disentanglement studies](#)

[4.1. Perceptual path length](#)

[4.2. Linear separability](#)

[5. Conclusion](#)

[Reference](#)

Intro

StyleGAN을 온전히 이해하려면 사전적으로 **Deep Convolutional GAN, PGGAN, Style Transfer**,

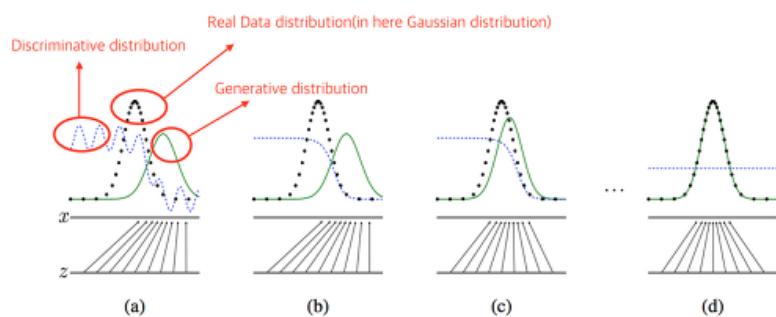
Feature Extracting, Instance Normalization 등의 개념을 모두 알고 있어야 함

▼ Generative Adversarial Networks (=GAN)

적대적 시스템으로 이미지를 생성하는 인공 신경망

생성자(generator)와 판별자(discriminator) 두 개의 네트워크를 활용한 생성 모델

- 최적화 과정



초록색 - Generator가 생성해 낸 이미지의 분포

파란색 - Discriminator의 값

(a)에서는 Generator가 원본 이미지의 분포를 잘 설명하지 못해 차이가 생김

→ 이에 따라 Discriminator가 Generator 부분에서는 낮은 확률, 반대는 높은 확률로 진짜/가짜를 구분해 냄

(d) 진짜 이미지와 생성해 낸 이미지의 분포가 같아짐

→ discriminator는 0.5가 되어 진짜/가짜를 구분해 낼 수 없음

→ 랜덤하게 선택하라 수 밖에 없음

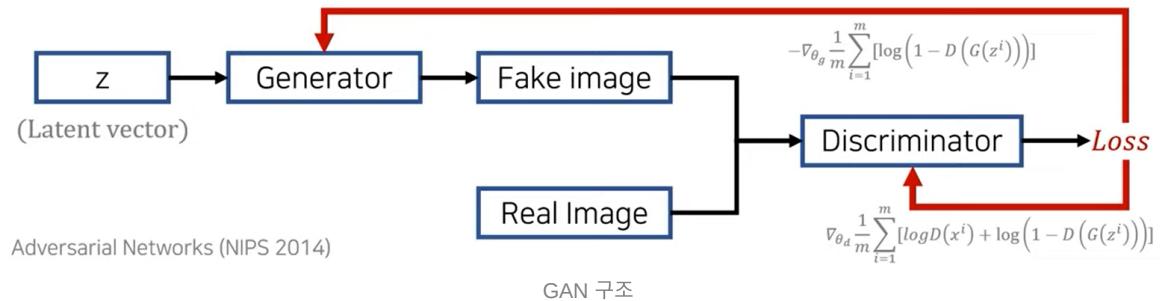
- **Loss Function**

$$\min_G \max_D V(D, G) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

$-\infty \leq$ 목적함수 ≤ 0 , $D_{\min} \leq$ 목적함수 $\leq D_{\max}$

Generator G(z) : new data instance

Discriminator D(x) (=Probability) : a simple came from the real distribution (Real:1 ~ Fake:0)



▼ Deep Convolutional GAN (=DCGAN)

DGGAN (ICLR 2016)

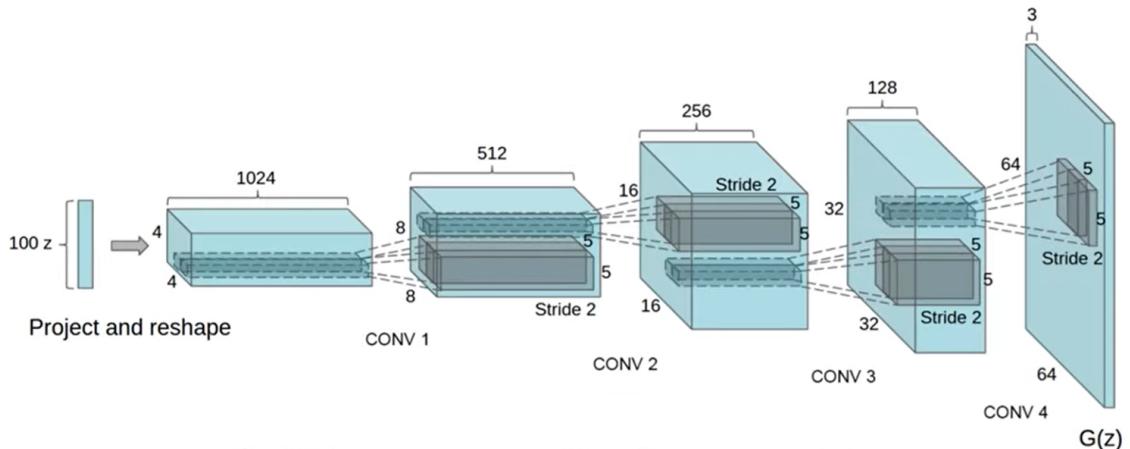


Fig. DCGAN generator used for LSUN scene modeling.

DCGAN

DCGAN은 StyleGAN의 base가 되는 모델

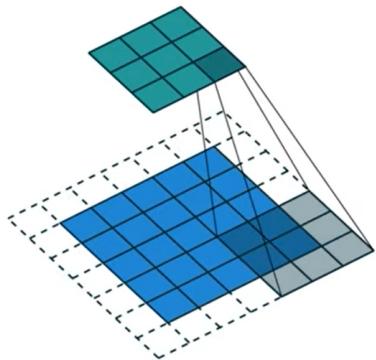
Deep Convolutional Layers를 이용하여 이미지 도메인에서의 높은 성능을 보임

기본적인 NLP를 사용한 GAN과는 다르게 Convolutional Layer로 고해상도의 이미지 생성

Width, Height를 증가시키고 채널은 감소시키는 방향으로 결과 이미지를 만들어 냄

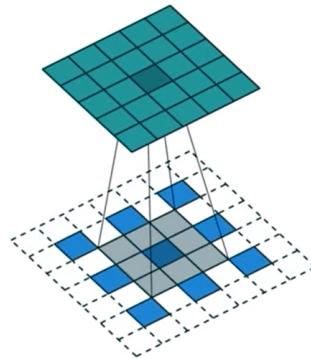
(일반적인 분류 네트워크와 반대)

판별자(Discriminator)



Strided Convolution: 너비와 높이가 감소

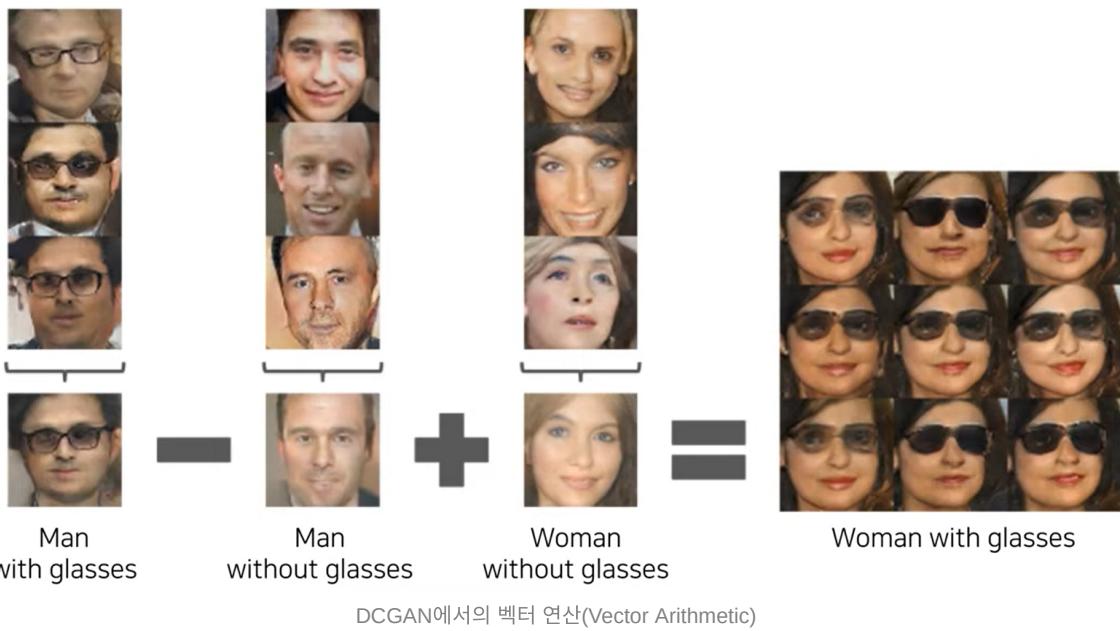
생성자(Generator)



Transposed Convolution: 너비와 높이가 증가

DCGAN에서의 Convolutional 필터

- **Discriminator(Downsampling)** : Layer가 깊어질수록 너비와 높이는 감소, 채널 증가
- **Generator(Upsampling)** : Layer가 깊어질수록 너비와 높이 증가, 채널 감소



하나의 생성자를 잘 학습한 뒤, 많은 횟수로 Sampling 진행

각 특징 latent vector 의 평균값

▼ Progressive GAN (=PGGAN)

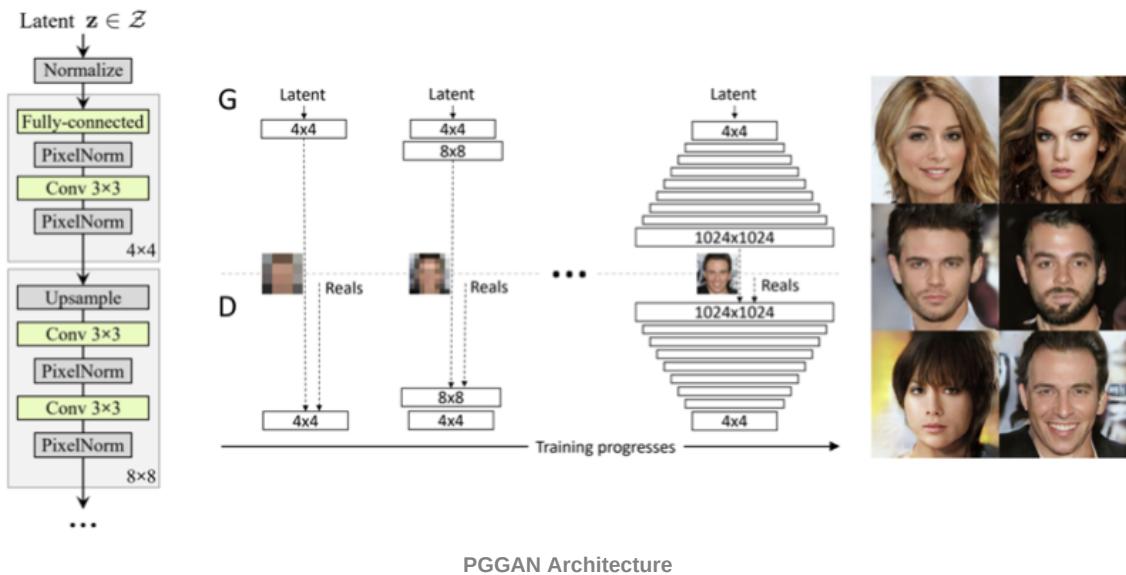


학습 과정에서 레이어를 추가함

고해상도 이미지 학습을 성공시킴

but, 이미지의 특징 제어가 어려움 (ex. 안경을 씌우거나, 얼굴의 방향을 바꾸는 과정에서 다른 특징들이 개입)

→ StyleGAN에서 개선함



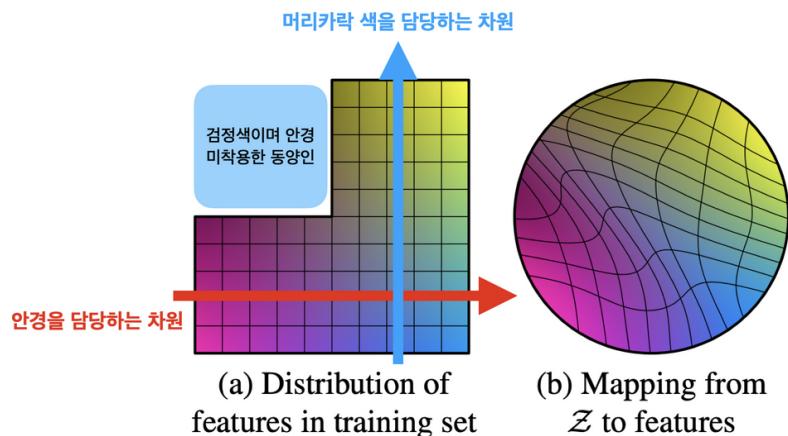
PGGAN은 StyleGAN의 base가 되는 모델

낮은 해상도부터 높은 해상도까지 점진적(progressively) 으로 생성하는 대표적인 생성 모델

latent vector z 가 Normalize를 거쳐 모델에 바로 입력이 되는 형태로 학습이 진행됨

이렇게 z 가 Generator에 바로 입력으로 들어가면 GAN은 latent space가 무조건 **학습 데이터 셋의 확률 분포와 비슷한 형태로 만들어지도록 학습을 하게 됨**

but, 이렇게 되면 latent space가 **entangle**하게 만들어 지게 됨



(a)는 학습 데이터셋을 시각적으로 보여주는 그림

학습 데이터 셋에는 검정 머리이며 안경을 착용한 동양인으로 구성되었다고 가정

붉은색 화살표는 안경을 담당하는 차원이고 파란색 화살표는 머리카락 색을 담당하는 차원

PGGAN처럼 latent variable을 기반으로 학습하는 생성 모델은 noise 상태에 있는 latent space \mathcal{Z}

\mathcal{Z} 를 학습 데이터 분포와 비슷하게 변화시키는 mapping을 학습하는 것을 목표로 함!



(a)에서 (b)로 mapping 하는 이유

latent vector를 샘플링해서 생성자에 넣을 때 Gaussian Distribution를 따르기 때문에
(=구의 형태에서 Sampling을 하는 것과 같음)

학습 데이터 셋에 없는 데이터인 머리카락 색이 검정색이며 안경을 미착용한 동양인의 경우

- 학습 데이터의 고정된 분포를 latent space가 **non-linear하게 mapping**이 되어버린 상태
- 생성하기 매우 어려움

PGGAN의 단점

눈동자 색만 다른 색으로 색칠하고, 머리카락 색만 다른 색으로 색칠하고자 할 때,
Generator에 **latent vector z가 바로 입력**되게 때문에 entangle하게 되어서 불가능
다양한 특징들이 적절히 분리되어 있지 않기 때문에 안경을 바꾸거나,
얼굴의 각도를 조절할 때 다른 특징들이 많이 개입되는 문제

Abstract

제안 네트워크

StyleGAN은 PGGAN 구조에서 **Style Transfer** 개념을 적용해 **Generator architecture**를 재구성



High-level attribute : human pose or identity on human face

Stochastic Variation : skin or hair (동일한 사람의 사진이어도, 사람을 구성하는 요소는 확률적으로 다양한 방식으로 달라질 수 있음)

새로운 기능

1. 새로운 아키텍처는 자동으로 학습되고 high-level attributes (e.g. pose and identity)와 생성된 이미지의 stochastic variation(e.g. freckles, hair)을 separation
→ 직관적으로 scale-specific control 이 가능하게 함 (PGGAN에선 불가능)
2. 새로운 generator는 traditional **distribution quality metrics** 측면에서 SOTA를 향상
3. 더 나은 **interpolation properties** 입증하며 latent factor of variation을 잘 disentangled 하게 함
interpolation quality와 disentanglement을 정량화(quantify)하기 위해 모든 generator 아키텍처에 사용할 수 있는 두 가지 새로운 방법 제안



Entangle

서로 얹혀 있는 상태여서 특징 구분이 어려운 상태
즉, 각 특징들이 서로 얹혀있어서 구분이 안됨

Disentangle

여러 개의 feature가 서로 얹혀있지 않고 linear하게 분포가 되어있는 것을 의미
선형적으로 변수를 변경했을 때 어떤 결과물의 feature인지 예측할 수 있는 상태
각 style들이 잘 구분 되어있는 상태여서 어느 방향으로 가면 A라는 특징이 변하고 B라는 특징이 변하게 되어서 특징들이 잘 분리가 되어있다는 의미
cf) 다양한 특징들이 연관관계를 가지고 있으면(Disentangle이 부족하다면) 요소를 변경해서 Image를 만들고 싶을 때 다른 요소까지 변경됨

새로운 데이터 기법 & 데이터셋

2개의 새로운 method를 제시하고, 새로운 데이터셋 배포

1. Introduction

GAN 기반의 기술은 PGGAN, BEGAN 등 발전해가고 있지만 Generator의 과정은 아직도 **Black Box**
이미지를 생성할 때 stochastic feature를 control하는 방법과 latent space에 대한 연구 역시 더 필요함
→ 본 논문에선 **style transfer**에 대해 영향을 받았고 이를 토대로 StyleGAN이 나오게 된 것

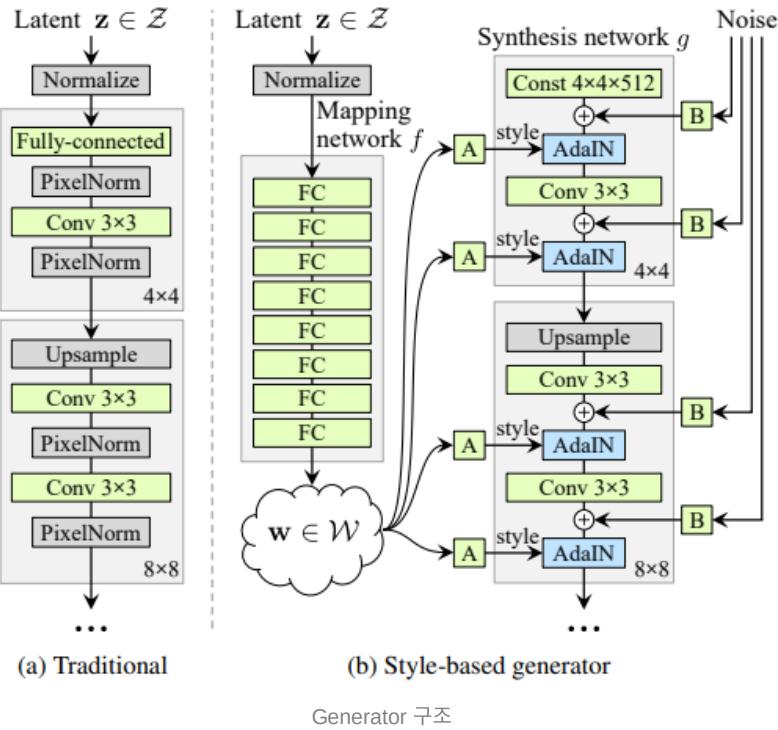
실제로 본 network에서는 학습된 상수 값으로부터 각각의 convolution layer를 거치면서 style값을 조절 및 추가
(본 논문은 이미지를 여러 style의 조합으로 보기 때문에 각각의 layer를 거쳐가며 style정보를 입히는 방식으로 이미지를 합성하는 것)

또한 discriminator나 loss function은 수정하지 않고, **generator**를 더욱 효과적으로 만드는데 집중

새로운 2가지의 metric (**perceptual path length & linear separability**)를 사용했을 때 기존의 generator architecture보다 조금 더 linear하게 동작하면서 특징들을 control하기에 간편했다고 주장

FFHQ라는 1024*1024 size의 고해상도 image dataset을 배포

2. Style-based generator



Mapping Network에서 만들어진 latent vector w 가 입력 값으로 들어가 다양한 이미지가 생성

저해상도 이미지에서 시작하여 너비와 높이를 증가시키는 방향

2개의 Convolution Layer와 2개의 AdaIN Layer를 가진 총 9개의 블록으로 구성

초기 입력은 상수(constant)로 대체 (경험적으로 성능 향상)

다양한 Style에 대한 정보가 Layer를 거치면서 적용될 수 있게 함으로써 이미지의 다양성이 보장

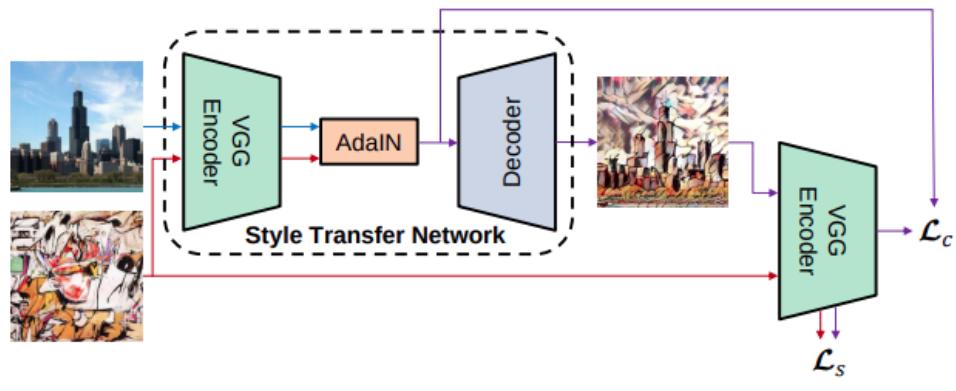
→ 굳이 하나의 latent vector로부터 input image가 시작할 필요는 없음!

또한 stochastic variation을 처리할 수 있도록 별도의 noise에 대한 vector 역시 추가로 넣어줌

ex. 주근깨, 머리카락의 배치

▼ Adaptive Instance Normalization (AdaIN)이란?

하나의 이미지가 여러 개의 semantic한 style 정보로 구성되는 형태로 이미지를 생성할 수 있도록 architecture 고안



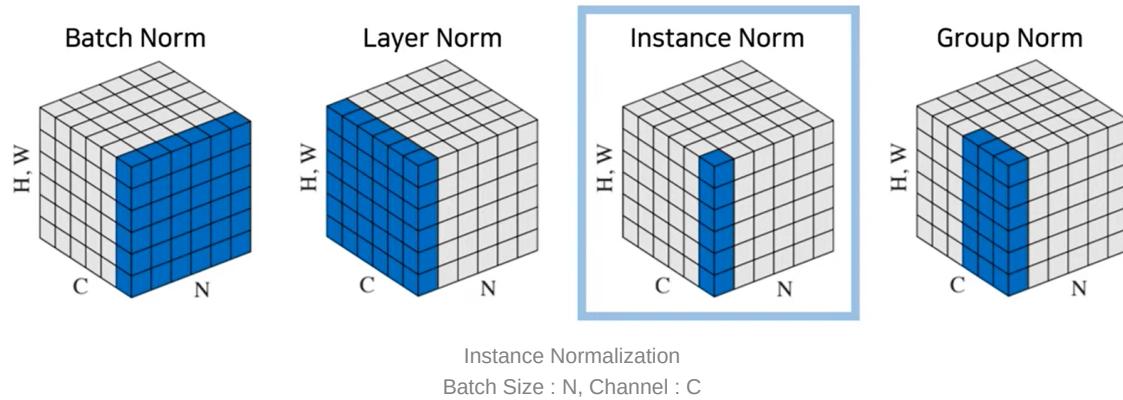
AdaIN Network 구조

AdaIN을 이용하면 다른 데이터로부터 스타일 정보를 가져와 적용할 수 있음

다른 데이터로부터 style 정보를 가져오기 때문에 AdaIN Layer에서 학습 시킬 별도의 파라미터가 필요하지 않음
(Batch Norm에서 사용되는 γ 와 β 사용하지 않음)

feed-forward 방식의 style transfer 네트워크에서 사용되어 좋은 성능을 보임

→ 다양한 style transfer 논문에서 사용되다가 StyleGAN이라는 이름이 붙여짐



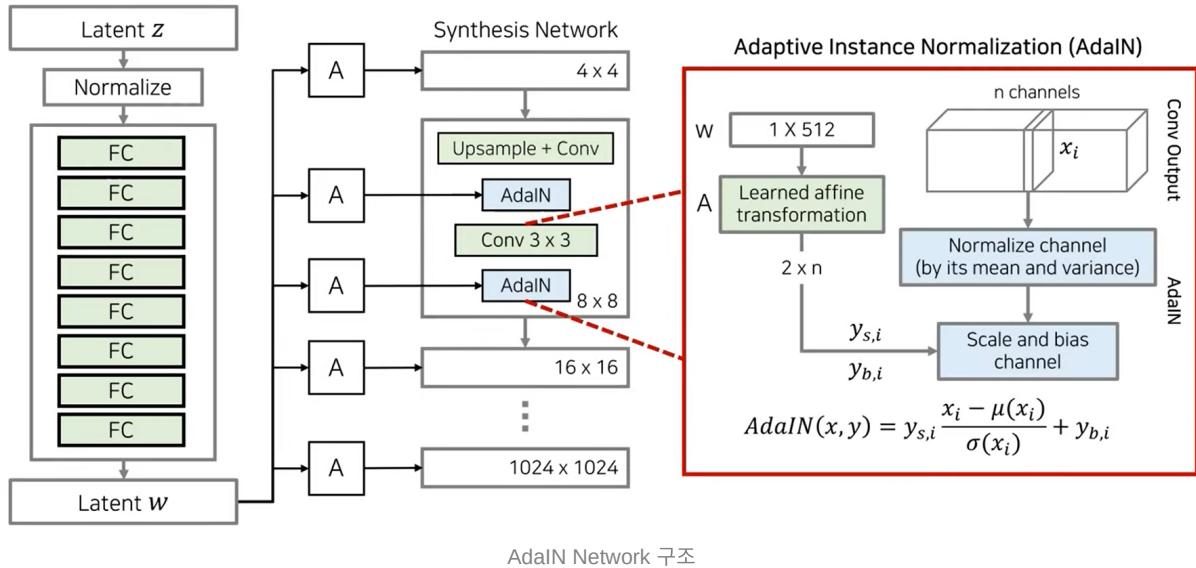
각각의 데이터 **Instance** 단위로 Normalization을 수행한다는 특징

하나의 이미지에 대해서 각각의 채널 단위로 정규화 수행

cf) Batch Norm : 하나의 Batch에 포함되어 있는 N개의 이미지에 걸쳐서 각 channel 마다 정규화

AdaIN을 거치게 된다면, 하나의 feature에 대한 정보를 scaling과 bias를 적용하면서 입력 받은 스타일의 statistics를 이용하여 하나의 feature에 대한 statistics를 바꿀 수 있도록 만듦

AdaIN



AdaIN Network 구조

기본적인 Neural Network에서 각 layer를 지나가며 scale, variance의 변화가 생김

- 이는 빈번하게 발생하며 학습이 불안정해지는 현상 발생
- 이를 방지하기 위해 Batch Normalization 방법과 같은 normalization 기법을 각 layer에 사용

\mathcal{W} 가 latent vector의 style로 각 scale을 담당하는 layer에 입력으로 들어가게 됨

본 논문에서는 \mathcal{W} 가 style에 영향을 주면서 동시에 normalization 해주는 방법으로 사용

AdaIN 수식

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

x 에 대해서 평균 값을 빼주고 표준편차로 나누어 주는 것은 정규화 수행

이후 추가적으로 s (=scaling)과 b (bias)를 적용함으로써 feature space에서의 statistics를 변경

$y_{s,i}$ 와 $y_{b,i}$ 는 \mathcal{W} 를 Affine Transformation을 거쳐서 shape을 맞추고 style을 입혀주게 됨

x 는 AdaIN network를 거치기 전의 input 값이고, y 에 의해서 매번 style이 바뀌는 것

또한 noise를 추가해서 stochastic한 detail를 변경할 수 있음

(이때 noise는 feature map과 비례한 크기를 가져야 함)

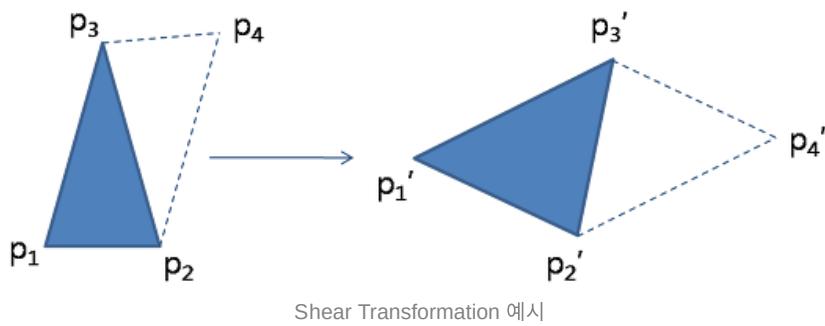
▼ Affine Transformation 이란?

아핀 변환은 선형 변환 중 하나로 점, 직선, 평면을 보존하며 평행선도 보존됨

어떤 도형이 있을 때 한 dimension의 길이에 비례하는 값을 다른 dimension에 더하는 변형

x, y 축에 독립적으로 translation이 이루어지기 때문에 점 사이의 상대적 거리도 유지하고, 평행선도 보존함

Affine Transform	Example	Transformation Matrix	
Translation		$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$	t_x specifies the displacement along the x axis t_y specifies the displacement along the y axis.
Scale		$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	s_x specifies the scale factor along the x axis s_y specifies the scale factor along the y axis.
Shear		$\begin{bmatrix} 1 & sh_y & 0 \\ sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	sh_x specifies the shear factor along the x axis sh_y specifies the shear factor along the y axis.
Rotation		$\begin{bmatrix} \cos(q) & \sin(q) & 0 \\ -\sin(q) & \cos(q) & 0 \\ 0 & 0 & 1 \end{bmatrix}$	q specifies the angle of rotation.



Method	CelebA-HQ	FFHQ
A Baseline Progressive GAN [30]	7.79	8.04
B + Tuning (incl. bilinear up/down)	6.11	5.25
C + Add mapping and styles	5.34	4.85
D + Remove traditional input	5.07	4.88
E + Add noise inputs	5.06	4.42
F + Mixing regularization	5.17	4.40

PGGAN Baseline부터 점차적으로 추가해가면서 FID를 비교한 표

C는 AdaIN Network 적용 / E는 noise를 추가 / F는 모든 method를 적용

▼ Frechet Inception Distance (=FID)이란?

실제 이미지와 생성된 이미지에 대해 통계 측면에서 두 그룹이 얼마나 유사한지 즉, 벡터 사이의 거리를 계산하는 metric

이 점수가 낮을수록 두 그룹의 이미지가 더 유사하거나 통계가 더 유사하다는 뜻이기 때문에 점수가 낮을수록 성능이 좋음 (만점은 0.0)

기존의 IS(Inception Score)를 개선시키기 위해 오직 **GAN의 성능 평가**를 위해 특별히 개발된 것!

Inception Score는 생성된 이미지만 사용하여 성능을 평가하는 반면,

FID는 대상 도메인의 **실제 이미지 모음 통계와 생성된 이미지 모음 통계**를 비교해 평가 진행

pretrained된 Inception v3에서 출력 레이어를 제거하고 **마지막 풀링 레이어의 활성화(activation)**를 사용했다고 해서 Frechet Inception Distance!

- **다변량 정규분포(Multivariate Normal Distribution) X, Y 사이의 거리**

$$d(X, Y) = (\mu_X - \mu_Y)^2 + (\sigma_X - \sigma_Y)^2$$

Inception 네트워크를 사용하여 중간 layer에서 feature를 추출

이 feature에서 평균 μ 와 공분산 Σ 를 추출계산

- **실제 이미지 X 와 생성된 이미지 Y 사이의 FID**

$$FID = \| \mu_X - \mu_Y \|^2 - Tr(\Sigma_X + \Sigma_Y - 2\Sigma_X \Sigma_Y)$$

<https://velog.io/@viriditass/GAN%EC%9D%80-%EC%95%8C%EA%B2%A0%EB%8A%94%EB%8D%B0-%EA%B7%B8%EB%9E%98%EC%84%9C-%EC%96%B4%EB%96%A4-GAN%EC%9D%B4-%EB%8D%94-%EC%A2%8B%EC%9D%80%EA%B1%B4%EB%8D%B0-How-to-evaluate-GAN>

2.1. Quality of generated images

CELEBA-HQ의 경우 **WGAN-GP** 사용

FFHQ는 **WGAN-GP(A)**와 **R1 regularization(B-F)** 사용

4x4x512 constant tensor에서 이미지 합성을 시작하여 architecture를 단순화

synthesis network가 AdaIN 연산을 제어하는 스타일만 입력을 받음에도 의미있는 결과를 생성

▼ WGAN-GP 이란?

기존의 GAN은 Loss Term이 안정적으로 학습이 안됨

WGAN은 함수가 **1-Lipshitz 조건**을 만족하도록 하여 안정적인 학습을 유도

본래 WGAN 논문은 weight clipping을 이용하여 제약 조건을 만족하도록 함

WGAN-GP에서는 **gradient penalty**를 이용하여 WGAN의 성능을 개선

$$L = \underbrace{\mathbb{E}_{\tilde{\mathbf{x}} \sim \mathbb{P}_g} [D(\tilde{\mathbf{x}})] - \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_r} [D(\mathbf{x})]}_{\text{Original critic loss}} + \lambda \underbrace{\mathbb{E}_{\hat{\mathbf{x}} \sim \mathbb{P}_{\hat{\mathbf{x}}}} [(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1)^2]}_{\text{Gradient penalty}}$$

WGAN-GP Loss

NIPS 2017에서 WGAN-GP가 나온 이후 대부분의 GAN에서 baseline으로 사용하게 됨

Method	Score
ALI [8] (in [27])	$5.34 \pm .05$
BEGAN [4]	5.62
DCGAN [22] (in [11])	$6.16 \pm .07$
Improved GAN (-L+HA) [23]	$6.86 \pm .06$
EGAN-Ent-VI [7]	$7.07 \pm .10$
DFM [27]	$7.72 \pm .13$
WGAN-GP ResNet (ours)	$7.86 \pm .07$

Inception Score 측정 결과



Style-based generator에서 선별되지 않은 FFHQ Dataset

style-based generator가 traditional generator에 비해 FID를 거의 20% 개선

평균 품질이 높고 안경, 모자와 같은 악세서리도 성공적으로 합성됨

모든 이미지는 1024x1024 해상도로 생성됨

2.2. Prior art

기존의 GAN Architecture에 대한 work는 **Discriminator**를 개선하는 데 초점

ex. multiple discriminators, multiresolution discrimination, self-attention

반면, Generator 측의 연구는 주로 input latent space의 정확한 distribution 또는 Gaussian Mixture Model, Clustering, encouraging convexity를 통한 **input latent space** 형성에 초점

conditional generator는 별도의 embedding network를 통해 **identity**를 generator의 많은 계층에 제공하는 반면, **latent**는 여전히 **input layer**를 통해 제공

기존에는 병렬 작업에서 AdaIN을 사용해 자체 변조(self modulate)

하지만 본 논문처럼 **intermediate latent space**나 **noise input**을 고려하지는 않음

3. Properties of the style-based generator

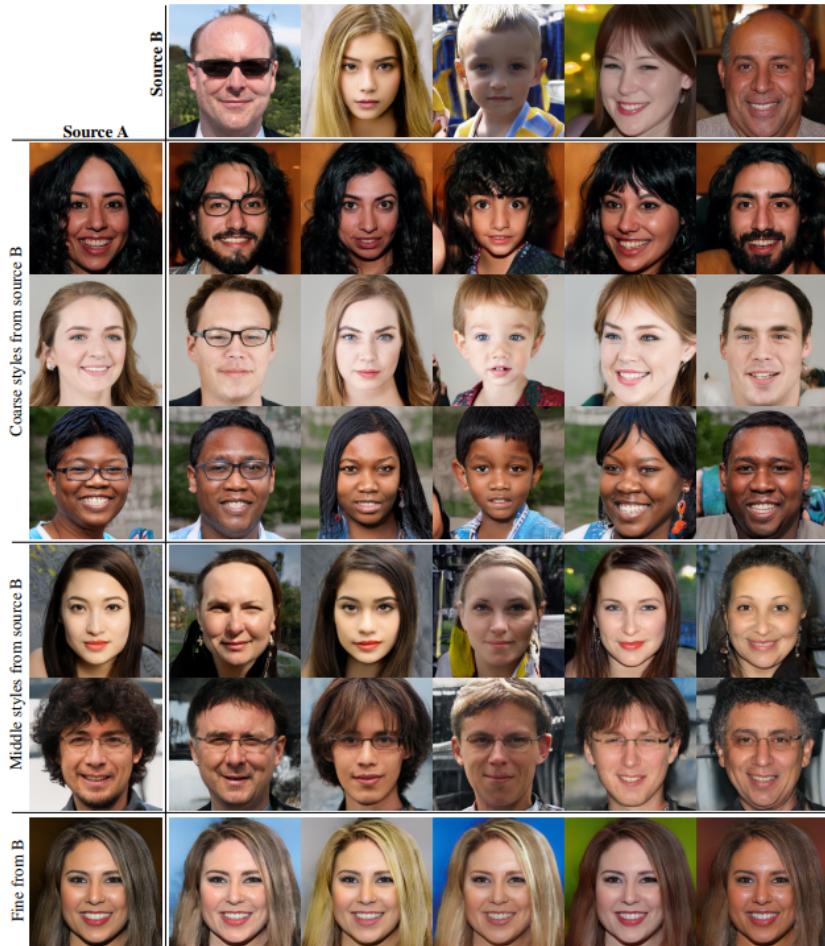
3.1. Style mixing

이 부분은 간단히 말하면 **인접한 layer간의 style 상관 관계를 줄이는 것!**

각 style이 잘 localize되어서 다른 layer에 관여하지 않도록 만들기 위해 style mixing을 제안하고 있음

2개의 latent vector가 있을 때 서로 섞어서 이미지를 만들 수 있도록 하는 것

여기서는 crossover point를 설정하여 이전은 w1으로 이후는 w2로 사용



- **윗부분 (Coarse style)** : 앞쪽 4개의 layer에 적용한 것이며 이미지의 큼직한 변화 (ex. 포즈나 안경) 가 생김
- **중간 부분 (Middle style)** : 중간 부분의 4개 layer에 적용한 부분이며 hair나 눈의 모양(뜨고 감은 부분)들의 조금 더 세밀해진 변화를 볼 수 있음
- **아랫부분(Fine)** : 마지막 10개의 layer를 변경했을 때의 결과를 본다면 색상이나 그림의 미세한 detail (ex. 배경) 이 변한 것을 볼 수 있음

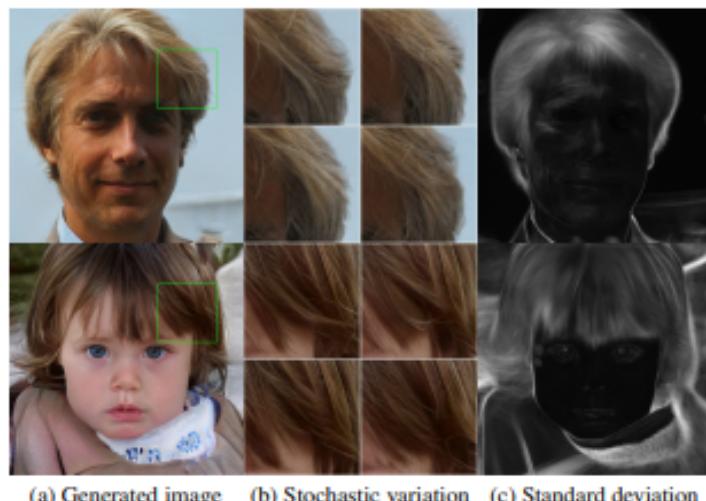
→ convolution layer에서 뒤로 갈수록 **patch의 크기가 작아지기 때문에 세밀한 변화가 가능해지는 것**



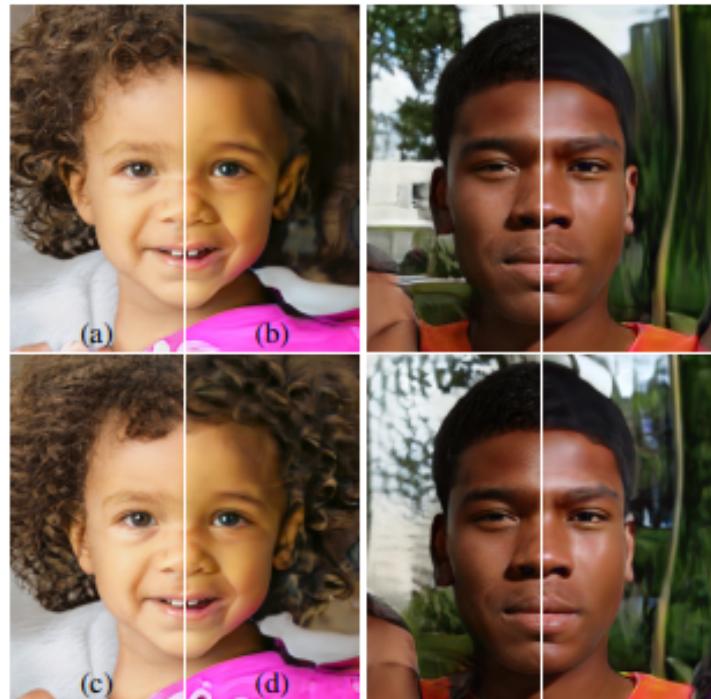
3.2. Stochastic variation

본 논문에서 나오는 style은 high-level global attribute(얼굴형이나 포즈, 안경)를 말하며 noise는 stochastic variation(머릿결, 주근깨나 피부 모공)을 말하는 것

noise vector의 변화를 준다면 아래의 그림과 같이 세밀한 부분들이 변경되는 것을 볼 수 있음



- **Coarse noise** : 큰 크기의 머리 곱슬거림, 배경 등
- **Fine noise** : 세밀한 머리 곱슬거림, 배경 등



(a) 모든 레이어에 노이즈 적용

(b) 노이즈 적용하지 않음

(c) Fine Layer에 적용

(d) Coarse Layer에 적용

3.3. Separation of global effects from stochasticity

feature map이 동일한 값으로 scaled 되고 biased 되기 때문에 style은 전체 이미지에 영향을 줌

포즈, 조명, 배경과 같은 global effect를 일관적이게 제어할 수 있음

반면, noise는 각 픽셀에 독립적으로 추가되므로 stochastic variation을 제어하는 데 이상적

noise를 사용하여 포즈를 제어하려고 하면 공간적으로 inconsistent decision이 발생하여 discriminator에 의해 불이익을 받음

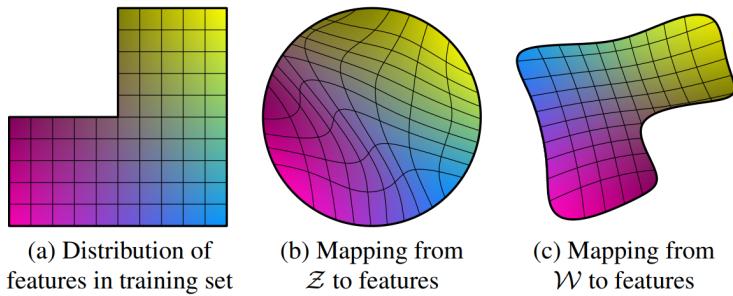
→ 네트워크는 global, local channel을 적절하게 사용하는 법을 배움

4. Disentanglement studies

Disentanglement 되었다는 것은 latent space가 linear한 subspace로 구성되어 있을 때를 뜻하며,

latent space에 존재하는 벡터를 두 개를 뽑고, 두 벡터 사이에 interpolation을 수행했을 때 우리가 의도했던 특정 feature만 변경될 수 있는 확률이 높아지는 것을 의미

특정한 factor만 잘 control할 수만 있다면 여러 개의 factor들이 잘 분류되어 있다고 할 수 있음



- (a) 세로축은 성별이며 가로축은 머리의 길이라고 했을 때, 머리가 긴 남성의 dataset이 없다고 가정
- (b)는 가우시안 distribution으로 mapping 시킨 것이고, 이러한 경우에서는 필연적으로 feature들이 entangle하게 됨
중간 값으로 interpolation하는 과정에서 급격하게 이미지의 semantic한 정보가 많이 변경될 수 있음
- (c) 하지만 W space로 mapping 시키고 나서 한다면 상대적으로 **덜 급격하게** 변경되게 됨
상대적으로 다양한 factor들이 linear하게 subspace를 갖도록 만들 수 있게 되는 것

latent space가 disentangle 하다는 것을 정량화하기 위해 다음과 같은 두 가지 measure를 제안

- **perceptual path length** : 두 벡터를 interpolation할 때 얼마나 급격하게 이미지 특징이 바뀌는지
- **linear separability** : latent space에서 attributes가 얼마나 선형적으로 분류될 수 있는지 평가

4.1. Perceptual path length

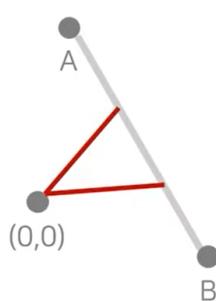
두 vector를 interpolation할 때 얼마나 급격하게 이미지 특징이 바뀌는지에 대한 지표

이것의 성능이 잘 나온다면 서서히 바뀌게 되는 것임을 증명함

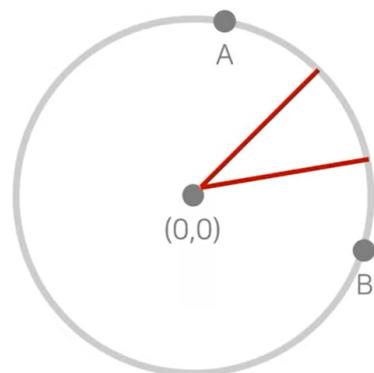
pre-trained VGG16 network를 통해서 perceptual loss를 계산함

▼ Latent Vector Interpolation

두 개의 latent codes를 보간(interpolation)하는 방법 2가지



Linear Interpolation (LERP)



Spherical Linear Interpolation (SLERP)

- **LERP(선형보간)** : A와 B를 잇는 직선을 그린 뒤, 직선 상에 존재하는 벡터들을 이용하는 방식
- **SLERP(구면선형보간)** : A와 B를 잇는 구면에 존재하는 다양한 벡터들을 Sampling
구면을 따라 중간 위치에 잡을 수 있도록 하는 방법임

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

지점 t 와 $t + e$ 사이에서의 VGG 특징(features)의 거리가 얼마나 먼지 계산
latent z 에 대해서 interpolation을 진행할 때는 구면선형보간을 사용

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

또한 w vector끼리의 interpolation에 대해서 진행할 때는 단순하게 선형보간 방법으로 수행
 w space에서 interpolation을 수행했을 때가 훨씬 좋은 것을 아래 표로 확인할 수 있음

Method	Path length		Separability
	full	end	
B Traditional generator \mathcal{Z}	412.0	415.3	10.78
D Style-based generator \mathcal{W}	446.2	376.6	3.61
E + Add noise inputs \mathcal{W}	200.5	160.6	3.54
+ Mixing 50% \mathcal{W}	231.5	182.1	3.51
F + Mixing 90% \mathcal{W}	234.0	195.9	3.79

4.2. Linear separability

latent space에서 attribute가 얼마나 선형적으로 분류될 수 있는지 판단하는 것

이를 위해 간단한 선형 분류기를 학습한 뒤 엔트로피를 계산해서 latent vector가 얼마나 linear한 subspace에 존재하는지를 확인하는 것

latent space point를 분류할 수 있는 하나의 선형 분류기(SVM)을 학습 후,

latent vector가 선형 분류기에서 얼마나 떨어져 있는지 알 수 있고 conditional entropy 값을 구함

entropy는 하나의 입력 벡터 X 가 주어졌을 때 그때의 true class에 대한 entropy 값을 측정할 수 있음

→ 하나의 이미지가 특정 클래스로 정확히 분류되기 위해 해당하는 feature가 얼마나 부족한 지에 대한 정보를 알 수 있으며 이는 값이 낮을수록 이상적인 image임을 나타냄

Method	FID	Path length		Separability
		full	end	
B Traditional 0 \mathcal{Z}	5.25	412.0	415.3	10.78
Traditional 8 \mathcal{Z}	4.87	896.2	902.0	170.29
Traditional 8 \mathcal{W}	4.87	324.5	212.2	6.52
Style-based 0 \mathcal{Z}	5.06	283.5	285.5	9.88
Style-based 1 \mathcal{W}	4.60	219.9	209.4	6.81
Style-based 2 \mathcal{W}	4.43	217.8	199.9	6.25
F Style-based 8 \mathcal{W}	4.40	234.0	195.9	3.79

5. Conclusion

StyleGAN은 Traditional GAN보다 월등히 좋은 result를 낼 수 있음

high level attribute가 잘 분리되며 stochastic effect도 잘 control 할 수 있음

latent space가 disentangle하게 되기 위해서 **Mapping Network**를 사용한 점이 인상 깊음

AdaIN을 통해서 원하는 style로 변형을 시켜줄 수 있는 모델이며 특히 높은 성능 향상을 보여줌

생성된 이미지들을 살펴 보니 물방울 형태의 작은 방울(blob)들이 관측된다는 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.

StyleGAN은 물방울 모양이 있는 상태로 생성됨

Reference

- [Blog](#)

A Style-Based Generator Architecture for Generative Adversarial Networks review

본 리뷰는 <https://arxiv.org/pdf/1812.04948v3.pdf> 논문을 참고 했습니다. [StyleGAN] 본 논문에서는 GAN에서 style transfer의 아이디어를 활용하여 새로운 generator architecture를 제안합니다. 이 구조를 활용한다면, high-level attribute 분류를 잘 할 수록 도와주고 (= 이를 잘 분리할 수 있게 된다

 <https://velog.io/@ghgh5317/A-Style-Based-Generator-Architecture-for-Generative-Adversarial-Networks-review>



StyleGAN: A Style-Based Generator Architecture for GANs

본 포스트는 최근 NVIDIA에서 발표하여 놀라운 성능으로 화제가 된 StyleGAN에 대해 소개 합니다. Introduction 최근 Generative Adversarial Network (GAN)를 기반으로 한 이미지 합성 기술은 BEGAN, PGGAN 등을 거치며 눈부신 발전을 거듭하고 있습니다. 그러나

 <https://blog.lunit.io/2019/02/25/a-style-based-generator-architecture-for-generative-adversarial-networks/>



• Youtube

StyleGAN: 고해상도 가상 얼굴 이미지 생성 기술 (꼼꼼한 딥러닝 논문 리뷰와 코드 실습)

딥러닝을 시작하시는 많은 분이 StyleGAN 모델에게 매력을 느껴 시작하곤 합니다. StyleGAN을 이용하면 실제로는 존재하지 않지만, 굉장히 그럴싸한 1024 x 1024의 고해상도 얼굴 이미지를 만들 수 있습니다. 많은 분이 알고 있을 <https://thispersondoe...>

 <https://www.youtube.com/watch?v=HXgfw3Z5zRo>

논문 소개: StyleGAN (CVPR 2019)

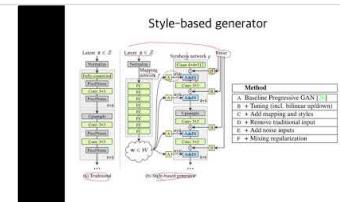


PR-131: A Style-Based Generator Architecture for Generative Adversarial Networks

paper : <https://arxiv.org/abs/1812.04948>

 <https://www.youtube.com/watch?v=TWzEbMrH59o>

Style-based generator



• GitHub

[GAN] Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

CycleGAN 논문 구현 및 생각 과정과 정리 현재 하고 있는 task에서 cycleGAN을 활용한 방법이 있어 CycleGAN 부터 제대로 알고 구현하고자 좀 딥하게 읽고 생각해봤습니다. CycleGAN 논문과 공식 구현 코드입니다. CycleGAN 이해에 도움이 될만한 자료입니다. 논문의 공동 저자인 박태성님의 한글 발표 자료입니다.

<https://subinium.github.io/CycleGAN/>



PyTorch-CycleGAN/train at master · aitorzip/PyTorch-CycleGAN

You can't perform that action at this time. You signed in with another tab or window. You signed out in another tab or window. Reload to refresh your session. Reload to refresh your session.

 <https://github.com/aitorzip/PyTorch-CycleGAN/blob/master/train>

aitorzip/PyTorch-CycleGAN



A clean and readable Pytorch implementation of CycleGAN

1 Contributor 29 Issues 843 Stars 237 Forks