Jiin Kim

Progressive Growing of GANs for Improved Quality, Stability, and Variation

Karras. T., Aila. T., Laine. S.,Lehtinen. J., Progressive Growing of GANs for Improved Quality, Stability, and Variation,*arXiv:1710.10196*,2017.

안녕하세요. 이번에 PGGAN 논문 발표를 하게 된 김지인입니다.

논문 제목은 Progressive Growing of GANs for Improved Quality, Stability, and Variation 입니다.

제목에서 알 수 있는 것은 이미지의 품질, training 의 안정성, 그리고 variation 의 증가를 위해 Progressive growing of GANs 를 만들었다는 것입니다.

이제 살펴보겠습니다.

Contents

목차는 다음과 같습니다.

Introduction

Introduction 입니다.

여기서 중점적으로 배우게 될 것은 GAN 의 한계점입니다.

GAN 의 한계점을 잘 살펴보고 PGGAN 이 어떻게 그것들을 극복했는지 비교해보면 좋을 것 같습니다.

Generative Models - overview

- Autoregressive model (e.g. PixelCNN)
  - Sharp images, slow to evaluate. No latent space
- VAEs
  - Fast to train, burry images
- GANs
  - Sharp images, low resolution, limited variation, unstable training

가장 유명한 generative methods는 Autoregressive models, VAEs, GANs가 있습니다.

Autoregressive models는 sharp image를 생성하지만 evaluation이 느리고 latent representation이

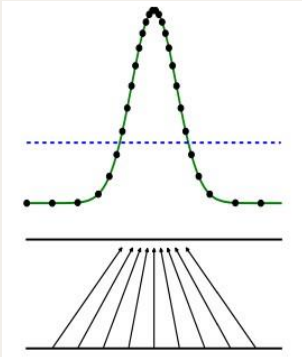
없습니다.

VAEs는 train은 빠르지만 blurry한 결과를 출력합니다.

GANs도 sharp image를 생성하지만, low resolution(저해상도)에서 동작하고 variation이 제한적이며 training이 불안정합니다.

여기서 variation 이라는 단어에 대해 짚고 넘어가자면, face image 를 예로 들었을 때 variation 에는 인물의 포즈, 조명 등이 있습니다.

Generative Models - GANs



Blue, dashed line: discriminative distribution
Black, dotted line: data distribution (P_{data})
Green solid line: generative distribution (P_g)

PGGAN 을 배우기에 앞서, GAN 에 대해서 먼저 한번 살펴보겠습니다. 이 그림은 goodfellow의 GAN 논문에서 가지고 왔습니다.

파란색의 dashed line 은 discriminative distribution 입니다.

검정색의 dotted line 은 data distribution 입니다.

초록색의 solid line 은 generative distribution 입니다.

training을 거친 후에 generator 와 discriminator 가 평형상태에 도달하면, $P_g = P_{data}$ 가 됩니다.

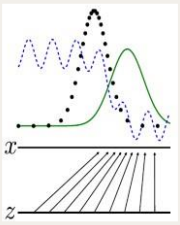
그림을 보시면, 검정색 그래프와 초록색 그래프가 일치하는 것을 볼 수 있습니다.

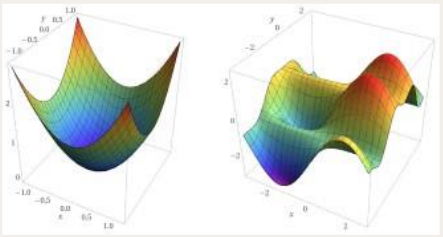
그리고, discriminator는 ½로 수렴합니다.

Generative Models - GANs

- **Challenge 1**
  - If not much overlap between training and generated distributions

    then gradients can point in random directions.





GANs 이 잘 하지 못하는 부분. 4가지의 challenge 에 대해서 알아보겠습니다.

Challenge 1 입니다.

If not much overlap between training and generated distributions then gradients can point in random directions

training distribution 과 generated distribution 이 잘 겹쳐지지(overlap 되지) 않으면, 기울기가 random 한 방향을 가리키게 됩니다.

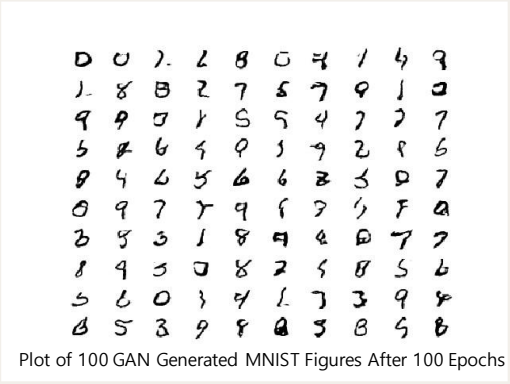
그림을 가지고 한번 더 설명

training distribution 과 generated distribution 이 겹쳐지지 않으면

기울기가 이리저리 뛰어서 local minimum 을 찾기 힘들게 됩니다.

Generative Models - GANs

- **Challenge 2**
  - Little variation in results


Plot of 100 GAN Generated MNIST Figures After 100 Epochs

Challenge 2 입니다.

Little variation in results

GAN 의 결과에는 variation 이 적습니다.

MNIST 로 예를 들어보겠습니다.

MNIST 의 경우, 1을 생성할 때 정말로 딱 반듯한, 우리가 1을 생각하면 바로 튀어나오는 그 이미지를 생성하는 것을 볼 수 있습니다.

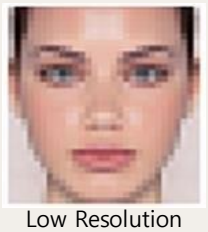
Face image도 마찬가지입니다.

이미지의 색이 파란색이거나, 방향이 90도 회전해 있거나, 이런 variation 은 GAN 에서 잘 일어나지 않습니다.

즉, GAN 은 색상, 질감, 방향 등에서 변이가 많이 없습니다.

Generative Models - GANs

- **Challenge 3**
  - High resolution harder because easier to tell apart


Low Resolution


High Resolution

Challenge 3 입니다.

High resolution harder because easier to tell apart

High resolution 이미지에서 generated image와 training image는 너무 쉽게 구별이 갑니다.

화질이 높기 때문에 아주 작은 차이도 잡아낼 수 있기 때문입니다.

low resolution 은 눈이 잘 못 그려져도 잘못 그려진 것인지 잘 구분하지 못합니다.

그러나, high resolution 은 눈썹이 조금만 이상해 져도 바로 티가 납니다.

그래서 high resolution 일 때 이미지 생성은 매우 어렵습니다.

참고로 초기 gan 은 매우 low resolution 인 28x28 mnist dataset 을 사용하여 실험했습니다.

Generative Models - GANs

- **Challenge 4**
  - High resolution requires smaller minibatches so training less stable

High Resolution

마지막, Challenge 4 입니다.

High resolution 은 더 작은 minibatch 크기를 요구하고 그것은 곧 training 이 불안정해지는 것을 이끕니다.

당연히 high resolution 이미지는 low resolution 이미지보다 메모리를 많이 잡아먹습니다.

그래서 어쩔 수 없이 minibatch 크기를 작게 잡아야 합니다.

그렇게 되면 training 은 느려지게 됩니다.

PGGAN - overview

- Key insight
  - we can **grow both the generator and discriminator progressively**,
  - starting from easier **low-resolution images**,
  - and **add new layers** that introduce higher-resolution details as the training progress

PGGAN 의 key insight 에 대해 알아보겠습니다.

generator 와 discriminator 두개를 함께 점진적으로 키웁니다.

입력 이미지는 low-resolution 이미지부터 시작합니다.

그리고 훈련이 진행됨에 따라 higher-resolution 세부정보를 도입하는 새로운 레이어를 추가합니다.

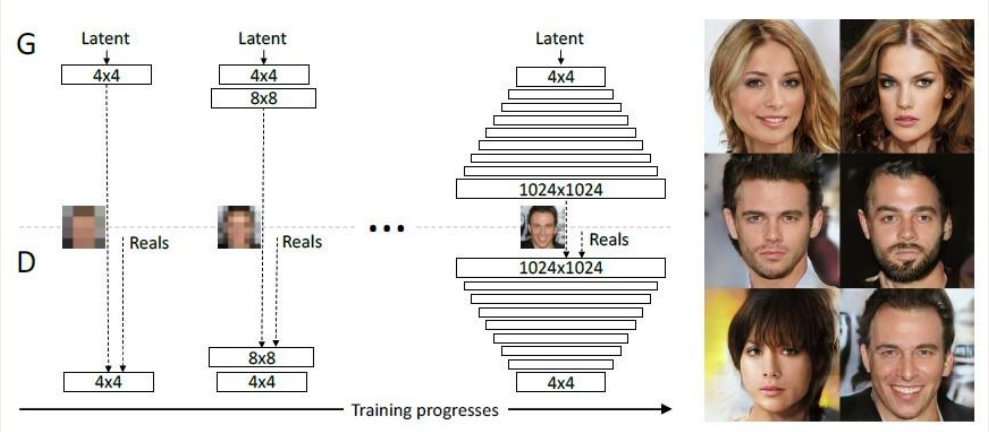
더 구체적인 설명은 바로 다음 섹션에서 하겠습니다.

이제 본격적으로 PGGAN 에 대해서 알아 보겠습니다.

Progressive Growing of GANs 을 줄여서 PGGAN 이라고 합니다. PGGAN 은 말 그대로, GAN 이긴 GAN 인데, 점진적으로 무럭무럭 자라는 GAN 입니다.

어떻게 무럭무럭 자라는지 한번 살펴봅시다.

PGGAN - growing the GAN



어떻게 training 을 하는지 살펴보겠습니다.

Training은 저해상도 4 x 4 이미지와 함께 generator(G)와 discriminator(D)에서 시작합니다.

전체 네트워크에서 위쪽이 generator, 아래쪽이 discriminator 입니다.

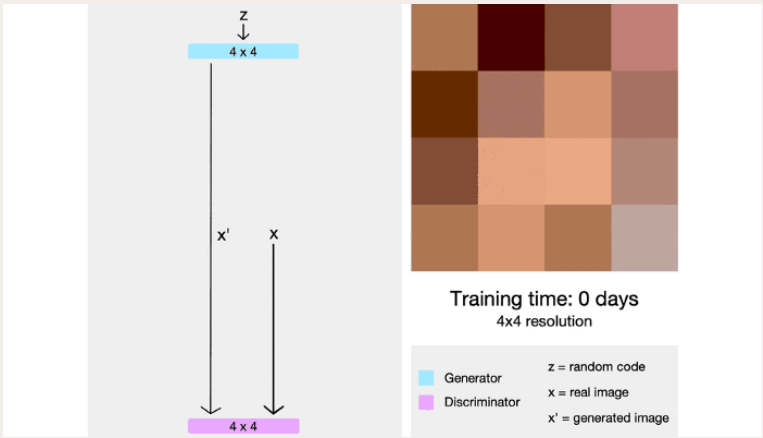
학습이 진행되면, 점진적으로 G와 D에 레이어를 추가해서 생성 이미지의 spatial resolution을 증가시킵니다.

그림을 보면, 훈련이 진행됨에 따라 G와 D에 레이어가 하나씩 추가되고 인물의 face image 도 점점 higher resolution 을 얻는 것을 볼 수 있습니다.

이 방법은 안정적으로 high resolution 이미지를 만들고, 학습 속도도 상당히 빠르게 만듭니다.

이렇게 학습을 할 때 네트워크를 점진적으로 키우면, 이미지에서 전체적인 구조부터 찾고 점차 더 세밀한 곳으로 집중하게 됩니다.

PGGAN - growing the GAN

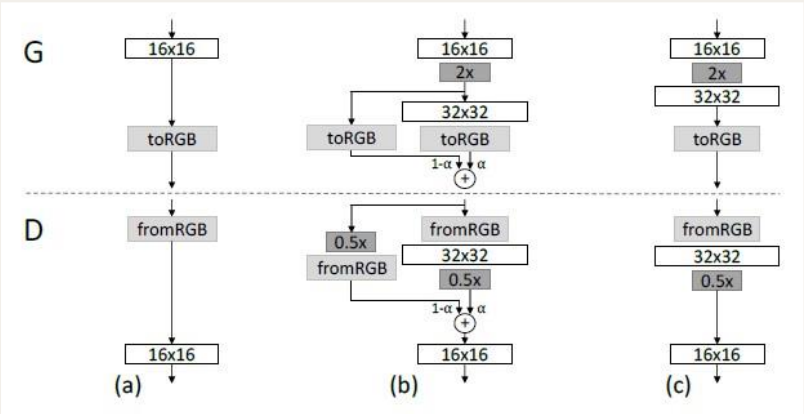


Generator와 discriminator의 구조는 서로 대칭적이고 항상 동시에 커집니다.

학습을 하는 동안, 레이어의 추가와 관계없이 두 네트워크의 모든 레이어들은 계속 trainable 합니다.

PGGAN - fading in higher resolution layers

Fade in smoothly



네트워크에 레이어를 추가하는 방법에 대해 알아보겠습니다.

PGGAN 은 네트워크에 새로운 레이어가 추가될 때, fade in smoothly 라는 방법을 사용합니다.

(a)에서 (c)로 가기 위해 (b) 과정을 거칩니다.

2x는 nearest neighbor filtering을 사용해서 image resolution을 두 배로 만들고, 0.5x는 average pooling을 사용해서 image resolution을 반으로 줄입니다.

toRGB는 feature vectors를 RGB colors로 바꾸는 과정이고, fromRGB는 그 반대의 과정입니다.

16 x 16 해상도 이미지를 만든 후, 바로 32 x 32 해상도를 학습시키면 아직 학습이 안된 32 x 32 레이어가 잘 학습된 16 x 16 레이어에 안 좋은 영향을 끼칠 수 있습니다.

그래서 "fade in smoothly"가 필요합니다.

구체적인 동작을 살펴보겠습니다.

G에서의 동작은 다음과 같습니다.

(b)에서 16 x 16 해상도 이미지 크기를 2배로 늘려 크기만 32 x 32인 이미지를 만듭니다.

그리고 32 x 32 레이어를 통과하면 고해상도 결과 이미지가 나옵니다.

고해상도 이미지의 각 픽셀 값에 0~1 사이의 값인 비율을 나타내는 α를 곱하고,

크기만 32 x 32 인 저 해상도 픽셀에는 그 나머지 비율인 $1-\alpha$를 곱합니다.

두 개의 결과를 서로 더하면 저해상도의 큰 그림과 앞으로 학습시킬 고해상도의 디테일이 합쳐집니다.

D에서도 G와 같은 방식으로 진행합니다.

"fade in"을 사용하면, 기존의 저해상도 레이어에서 학습한 것을 해치지 않고, 새로 추가된 레이어를 학습시킬 수 있습니다.

Loss Function

- The author's say their work is **independent of loss function**
- Do experiments with both WGAN-GP and LSGAN

PGGAN 은 loss function 에 독립적입니다.

그래서 실험을 할 때, Wasserstein GAN gradient penalty 와 Least squares GAN 모두 사용하여 실험했습니다.

PGGAN - benefits

- Training **avoids high resolution problem** of too much divergence early on
- **Faster** training, 2-6 x faster
- Only use a **single GAN** instead of a hierarchy of GANs
- More **stable training** - more steps done at lower resolution with larger minibatches

PGGAN 의 이점에 대해 알아보겠습니다.

PGGAN 의 training 은 초기에 발산하는 high resolution 문제를 해결합니다.

training 이 빠릅니다. 보통 2-6 배 빠릅니다.

하나의 네트워크로만 많은 문제를 해결할 수 있습니다.

training 이 더 안정적입니다. 더 큰 미니배치로 더 낮은 해상도에서 더 많은 단계를 수행합니다.

Increasing Variation using Minibatch Standard deviation

지금까지 PGGAN 의 네트워크 구조에 대해 알아보았습니다.

이번에는 minibatch standard deviation 기법을 사용하여 variation 을 증가시키는 방법에 대해 알아보겠습니다.

Minibatch standard deviation

- **Minibatch discrimination** (Salimans et al. 2016)
  - Compute feature statistics **across the minibatch**
  - Encourage the minibatches of generated and training images to show similar statistics
  - Add a **minibatch layer** towards the end of the discriminator

minibatch standard deviation 을 알아보기에 앞서, 먼저 Salimans et al. 이 2016년에 발표한 Minibatch discrimination 에 대해 알아보겠습니다.

minibatch discrimination 은 이미지 한 장 한 장이 feature statistics 를 계산하기도 하지만, minibatch 단위로도 feature statistics 를 계산합니다.

여기서 feature statistic 은 mean 과 standard deviation 이라고 생각하면 됩니다.

그래서 generated images 와 training images 가 유사한 statistics 를 갖도록 만듭니다.

이렇게 하기 위해, discriminator 의 끝에 minibatch layer 를 추가합니다.

minibatch layer 에서 minibatch 단위로 statistics, 즉 평균과 표준편차를 구하고, 구한 값들을 layer 의 output 에 concatenate 합니다.

이 과정을 수행하면, discriminator 는 내부적으로 statistics(평균과 표준편차)를 사용할 수 있습니다.

Minibatch standard deviation

- **Minibatch standard deviation**
  - **Simplifies** the minibatch discrimination and **improves variation**
  - How to compute ?
    - Compute standard deviation for each feature in each spatial location
    - Then average over all features and spatial locations to get a single value
    - Replicate the value and concatenate it to all spatial locations and over the minibatch, yields one additional feature map

이제 minibatch standard deviation 에 대해 살펴보겠습니다.

이것은 전 페이지에서 언급한 minibatch discrimination 을 단순화 시키고, variation 을 향상시킨 것입니다.

계산과정은 다음과 같습니다.

각 spatial location 에 있는 각 feature 에 대해 표준편차를 계산합니다.

모든 features 와 spatial locations 를 평균을 냅니다.

그 값을 복사해서 모든 spatial location 과 minibatch 에 연결시키면, 하나의 추가적인 feature map 이 만들어집니다.

Normalization in Generator and Discriminator

지금까지 variation 을 증가시키기 위한 기법으로 minibatch standard deviation 을 설명했습니다.

이번에는 generator 와 discriminator 를 normalize 하는 방법에 대해 설명하겠습니다.

이 섹션은 PGGAN 의 training 을 더 효율적으로 하기 위한 3가지 기법들이 설명되어 있는데, 좀 복잡한 설명이라 이해하기 어려우실 수도 있습니다.

저도 완벽하게 이해했는지는 잘 모르겠지만, 그래도 할 수 있는 만큼 한번 설명해보겠습니다.

Equalized Learning Rate

- Use trivial **N(0, 1)** weight initialization and **scale weights** at runtime
- $\hat{w}_i = w_i / c$
  - w_i: weights, c: per-layer normalization constant from Hés initializer
- Adaptive SGD methods normalize a gradient update by its estimated standard deviation
- If some parameters have a **larger dynamic range** than others, they will take **longer to adjust**
- Our approach ensures that the **dynamic range**, and thus the **learning speed**, is the **same for all weights**

먼저 learning rate 를 equalized 하는 방법에 대해 설명하겠습니다.

표준 정규분포를 사용하여 가중치를 초기화한 후, runtime 에서 가중치들을 scale 합니다. 즉, 조정합니다.

가중치를 조정하는 방식은 다음과 같습니다.

$\hat{w}_i = w_i / c$

w_i: weights, c: per-layer normalization constant from He's initialize

RMSProp 나 Adam 과 같은 adaptive SGD 방식은 가중치의 scale(크기) 에 관계없이 측정된 표준편차를 이용하여 gradient 를 update 합니다.

그 결과, 상대적으로 더 큰 dynamic range 를 가지는 일부 파라미터들에 대해서 적응하는데 시간이 더 오래 걸립니다.

우리의 접근방식은 모든 가중치들이 동일한 dynamic range 을 가짐으로써 동일한 learning speed 를 가질 수 있도록 보장합니다.

Pixelwise Feature Vector Normalization in Generator

- Normalize the feature vector in each pixel to unit length in the generator after each convolutional layer
- Prevent feature map magnitudes from getting too large

$$b_{x,y} = a_{x,y} / \sqrt{\frac{1}{N}\sum_{j=0}^{N-1}(a_{x,y}^{j})^2 + \epsilon}, \text{where } \epsilon = 10^{-8}$$

Generator 에서 수행하는 pixelwise feature vector normalization 에 대해 알아보겠습니다.

각 합성곱 연산 후에 generator에서 각 픽셀의 feature vector를 단위 길이로 정규화 합니다.

식은 다음과 같이 생겼습니다.

여기서,

N : feature map의 수

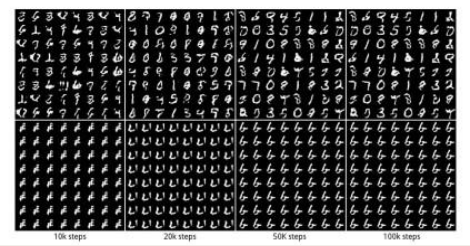
$a_{x,y}$: pixel (x, y)에서의 original feature vector

$b_{x,y}$: pixel (x, y)에서의 normalized feature vector

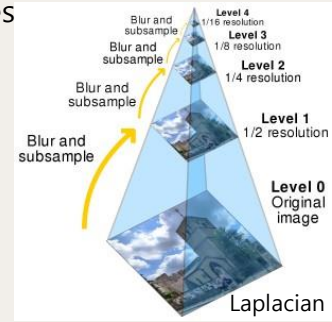
이 방법은 최종적인 결과를 바꾸지 못하지만, training 중에 signal 의 크기가 갑자기 커지는 현상을 막아줍니다.

Multi-Scale Statistical Similarity for Assessing GAN Results

- **MS-SSIM**: Good at identifying global mode collapse, **not good for local mode collapse** like on colors and textures
- Do MS-SSIM on local patches drawn from **Laplacian pyramid** representations of generated and target images



Mode collapsing



Laplacian pyramid

이번에 배울 것은 Multi-Scale statistical similarity 입니다. 이것은 줄여서 MS-SSIM 이라고 합니다.

MS-SSIM 에 대해 알기 전에, mode collapse 와 laplacian pyramid 부터 알고 넘어가겠습니다.

먼저 Mode collapse 는 대표적인 GAN 의 문제점으로 흔히 알려져 있습니다.

mode collapsing 는 학습시키려는 모델이 실제 데이터의 분포를 모두 커버하지 못하고 다양성을 잃어버리는 현상을 가리킵니다.

그저 손실(loss)만을 줄이려고 학습을 하기 때문에 G가 전체 데이터 분포를 찾지 못하고, 한번에 하나의 mode 에만 강하게 몰리게 됩니다.

예컨대 MNIST를 학습한 G가 특정 숫자만 생성하는 사례가 여기에 속합니다.

그림을 보게 되면, 윗줄은 GAN 의 mode collapsing 을 해결한 unrolled GAN 의 결과이고, 아랫줄은 standard GAN 의 결과입니다. 언제나 다양한 숫자들을 골고루 생성하는 윗줄과 달리, 아랫줄은 학습이 진행될 때 각 step 마다 한가지 모양만 종류를 바꿔가며 보여줍니다. 아랫줄은 학습이 잘 안되기도 했습니다. 숫자가 아닌 이상한 문양입니다.

여기까지 mode collapse 에 대한 설명이었습니다.

==========

이번에는 Laplacian pyramid 에 대해 알아보겠습니다.

Laplacian pyramid 는 이미지 피라미드의 한 종류로, 이미지 피라미드를 활용해 이미지의 크기를 원하는 크기까지 샘플링 합니다. 이미지의 크기를 확대하거나 축소했을 때 이미지들의 형태가 피라미드와 같이 표현됩니다.

==========

MS-SSIM과 같은 기존 방법은 global **mode collapses** 를 잘 찾습니다. MNIST 를 예로 들면 다양한 숫자를 생성하는지 잘 찾습니다.

그러나, 색상 변화나 질감 변화 같은 local mode collapse 는 잘 찾지 못합니다. 또한 generated image set 과 training set 의 유사성을 평가하지 않습니다.

Generated images(low resolution 16 x 16)와 target images(low resolution 16 x 16)를 **Laplacian pyramid** 에 넣고, 출력으로 나온 local image patch 들의 분포 사이에서 multi-scale statistical similarity(MS-SSIM)를 사용하는 방법을 제안합니다.

Multi-Scale Statistical Similarity for Assessing GAN Results

- Sample 16,384 images and extract 128 descriptors from each level of the Laplacian pyramid. Each descriptor is a 7x7 pixel neighborhood with 3 color channels
- Compute **sliced Wasserstein distance** between samples. **Smaller distance** means that at that level of resolution training images and generator samples have **similar variation**

multi-scale 에서 statistical similarity 를 하는 방법은 다음과 같습니다.

16.384 개의 이미지를 샘플링하고 Laplacian pyramid 의 각 레벨에서 128 개의 descriptor 를 추출합니다.

각 descriptor 는 3개의 color channel 들이 있는 7x7 pixel neighborhood 입니다. 즉, 7x7x3 입니다. 이 작업을 training set 과 generated set 에 대해 수행합니다.

그럼 training set 의 descriptor 들의 집합 (patch 들의 집합), generated set 의 descriptor 들의 집

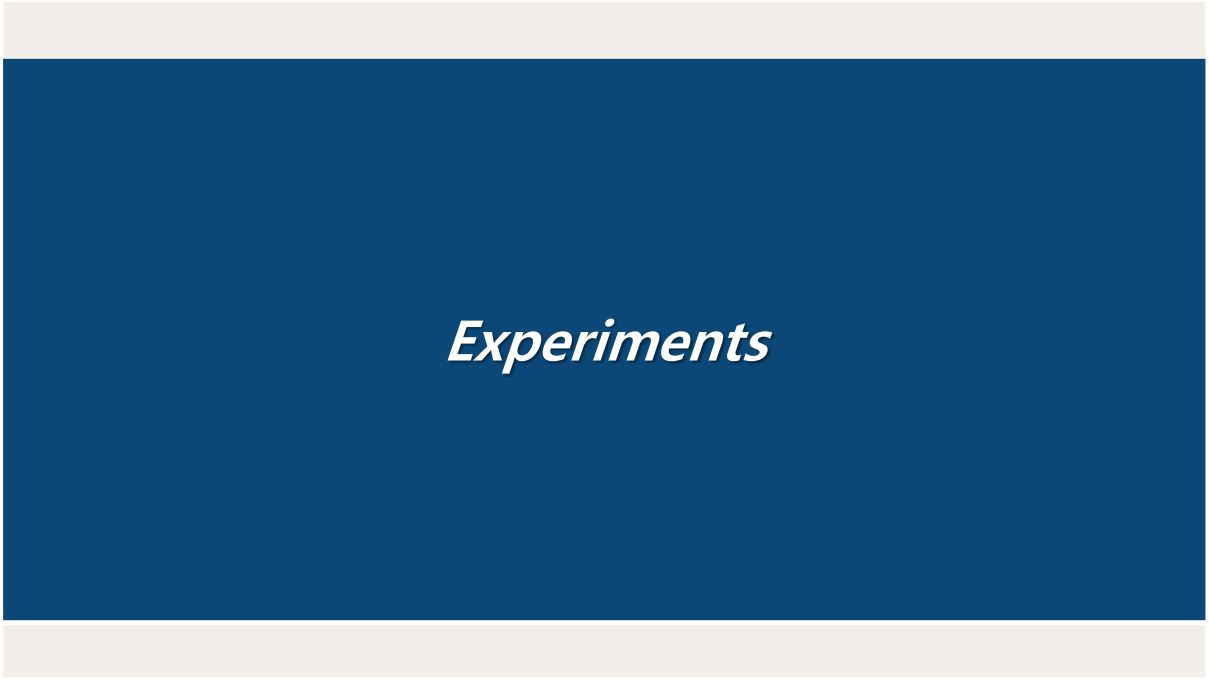
합 (patch 들의 집합)

training set 의 patch 들과 generated set 의 patch 들 사이에서 sliced Wasserstein distance 를 계산합니다.

더 작은 distance 는 해당 해상도 수준에서 training 이미지와 generator 이미지가 유사한 variation 을 가지는 것을 의미합니다.

여기까지 제가 이해한 바로는, Multi-Scale statistical similarity (MS-SSIM)은 LP 에 넣기 전에 유사성을 계산한 것이고

새로 제안한 방법은 LP 에 넣어서 나온 각기 다른 크기의 patch 들로 sliced Wasserstein distance 를 구하는 것입니다.



마지막 섹션인 Experiments 에 대해 설명하겠습니다.

Importance of Individual Contributions in Teams of Statistical Similarity

Training configuration	CELEBA						LSUN BEDROOM					
	Sliced Wasserstein distance $\times 10^3$					MS-SSIM	Sliced Wasserstein distance $\times 10^3$					MS-SSIM
	128	64	32	16	Avg		128	64	32	16	Avg	
(a) Gulrajani et al. (2017)	12.99	7.79	7.62	8.73	9.28	0.2854	11.97	10.51	8.03	14.48	11.25	**0.0587**
(b) + Progressive growing	4.62	**2.64**	3.78	6.06	4.28	**0.2838**	7.09	6.27	7.40	9.64	7.60	0.0615
(c) + Small minibatch	75.42	41.33	41.62	26.57	46.23	0.4065	72.73	40.16	42.75	42.46	49.52	0.1061
(d) + Revised training parameters	9.20	6.53	4.71	11.84	8.07	0.3027	7.39	5.51	3.65	9.63	6.54	0.0662
(e*) + Minibatch discrimination	10.76	6.28	6.04	16.29	9.84	0.3057	10.29	6.22	5.32	11.88	8.43	0.0648
(e) Minibatch stddev	13.94	5.67	2.82	5.71	7.04	0.2950	7.77	5.23	3.27	9.64	6.48	0.0671
(f) + Equalized learning rate	4.42	3.28	2.32	7.52	4.39	0.2902	**3.61**	3.32	**2.71**	6.44	4.02	0.0668
(g) + Pixelwise normalization	**4.06**	3.04	**2.02**	**5.13**	**3.56**	0.2845	3.89	**3.05**	3.24	**5.87**	**4.01**	0.0640
(h) Converged	2.42	2.17	2.24	4.99	2.96	0.2828	3.47	2.60	2.30	4.87	3.31	0.0636

Contribution을 평가하기 위해 sliced Wasserstein distance (SWD)와 multi-scale structural similarity (MS-SSIM)를 사용합니다.

Loss는 WGAN-GP를, 데이터셋은 CelebA와 LUSN bedroom을 사용합니다.

SWD와 MS-SSIM을 계산한 결과입니다.

SWD에서, 각 column 은 Laplacian pyramid의 level을 나타내고, 마지막 column 은 네 개 distance의 평균값입니다.

Importance of Individual Contributions in Teams of Statistical Similarity

Training configuration	CELEBA						LSUN BEDROOM					
	Sliced Wasserstein distance $\times 10^3$					MS-SSIM	Sliced Wasserstein distance $\times 10^3$					MS-SSIM
	128	64	32	16	Avg		128	64	32	16	Avg	
(a) Gulrajani et al. (2017)	12.99	7.79	7.62	8.73	9.28	0.2854	11.97	10.51	8.03	14.48	11.25	**0.0587**
(b) + Progressive growing	4.62	**2.64**	3.78	6.06	4.28	**0.2838**	7.09	6.27	7.40	9.64	7.60	0.0615
(c) + Small minibatch	75.42	41.33	41.62	26.57	46.23	0.4065	72.73	40.16	42.75	42.46	49.52	0.1061
(d) + Revised training parameters	9.20	6.53	4.71	11.84	8.07	0.3027	7.39	5.51	3.65	9.63	6.54	0.0662
(e*) + Minibatch discrimination	10.76	6.28	6.04	16.29	9.84	0.3057	10.29	6.22	5.32	11.88	8.43	0.0648
(e) Minibatch stddev	13.94	5.67	2.82	5.71	7.04	0.2950	7.77	5.23	3.27	9.64	6.48	0.0671
(f) + Equalized learning rate	4.42	3.28	2.32	7.52	4.39	0.2902	**3.61**	3.32	**2.71**	6.44	4.02	0.0668
(g) + Pixelwise normalization	**4.06**	3.04	**2.02**	**5.13**	**3.56**	0.2845	3.89	**3.05**	3.24	**5.87**	**4.01**	0.0640
(h) Converged	2.42	2.17	2.24	4.99	2.96	0.2828	3.47	2.60	2.30	4.87	3.31	0.0636

(a) (b) (c) (d) (e∗) (e) (f) (g) (h) Converged

(a) - (g) 는 Table 1의 행에 해당하는 CelebA 예제들입니다.

(h)는 PGGAN 결과입니다.

좋은 평가 지표는 색상, 질감, 방향에서 많은 variation이 있는 그럴듯한 이미지인지 판단할 수 있어야 합니다.

MS-SSIM은 outputs 사이에서의 variation만을 측정하기 때문에 generated images와 training set의 유사성을 판단할 수 없습니다.

그러나, SWD는 generated images의 분포가 training set 와 유사하다는 것을 올바르게 찾습니다.

==========

첫 번째 configuration (a)는 Gulrajani et al. (2017)입니다.

이 논문은 "Improved training of wasserstein gans" 입니다.

이 네트워크는 Generator에서 batch normalization 을 사용하고 Discriminator 에서 layer normalization 을 사용하며, minibatch 크기는 64입니다.

(b)는 네트워크에 progressive growing 을 추가했고, 더 좋은 결과를 냅니다.

주요한 목표 중 하나는 고해상도 이미지를 만드는 것입니다. 고해상도 이미지를 만들려면 메모리 제한으로 인해 mini-batches 의 크기를 줄여야 합니다.

© 에서 minibatch 크기를 64에서 16으로 줄여보았습니다. 그 때 생성된 이미지는 상당히 부자연스러운 것을 볼 수 있습니다. MS-SSIM 과 SWD 에서도 결과가 좋지 않은 것을 확인할 수 있습니다.

(d)는 hyperparameters 를 조정해서 training process 를 안정화 시키고, generator 에서 batch normalization 을 discriminator 에서 layer normalization 을 제거했습니다.

(e*)는 minibatch discrimination 을 사용했지만, 그다지 SWD 와 MS-SSIM 을 개선하지 못합니다.

대조적으로 우리가 제안한 minibatch standard deviation 을 사용한 € 는 average SWD 와 이미지들을 개선합니다.

그런 후, (f)와 (g)에서 우리의 contribution 을 추가해서, (f는 equalized learning rate, g는 pixelwise normalization) 전반적인 SWD 를 향상시키고 이미지의 질을 높입니다.

마지막으로, (h)에서 제대로 된 네트워크(PGGAN)를 사용하고 더 오랫동안 training 합니다. - SWD, MS-SSIM, 이미지, 모든 방면에서 결과가 좋은 것을 볼 수 있습니다.

Convergence and Training Speed

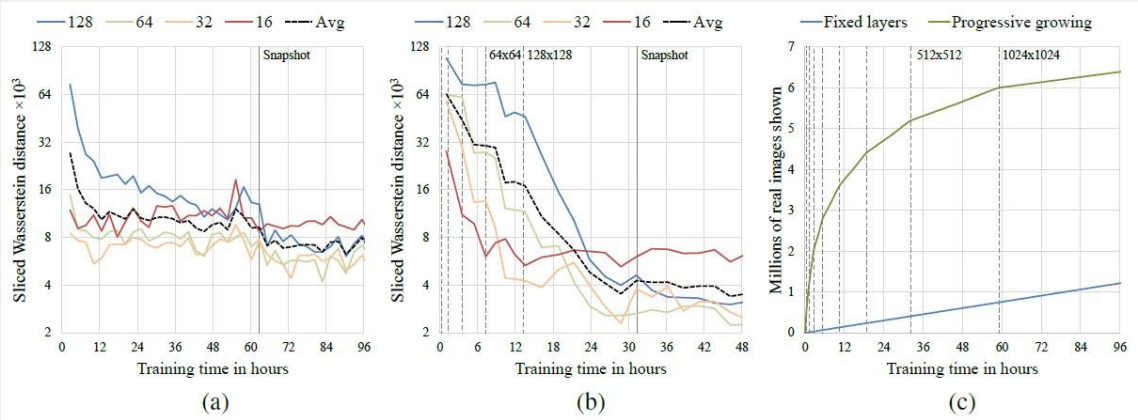


Figure 4는 SWD metric과 raw image throughput의 관점에서 progressive growing의 효과를 나타냅니다.

NVIDIA Tesla P100을 사용한 single-GPU setup에서 측정했습니다.

(a) 128 x 128 해상도를 가진 CelebA를 Gulrajani et al. (2017) 네트워크에 넣고 training을 한 것입니다.

각 그래프는 Laplacian pyramid의 한 레벨에서 sliced Wasserstein distance를 나타내고, 수직선은 Table 1에서 training을 중단한 지점입니다.

(b) Gulrajani et al. (2017)에 progressive growing을 추가한 그래프입니다. 점선으로 된 수직선은 G와 D에서 해상도를 두 배로 만들 지점을 나타냅니다.

(b)는 better optimum에 상당히 잘 수렴하고, 총 훈련시간을 약 2배 단축합니다.

Progressive growing이 없으면, generator와 discriminator의 모든 레이어들은 large-scale variation과 small-scale detail을 위해 간결한 intermediate representations를 동시에 찾습니다.

그러나 progressive growing이 있으면, 기존의 low-resolution layers는 이미 일찍 수렴될 가능성이 있어서 네트워크는 새로운 레이어가 도입됨에 따라 점점 더 작은 스케일의 효과로 representations를 구체화하는 작업만 담당합니다.

실제로, Figure 4(b)에서 largest-scale statistical similarity curve, 즉 level 이 16인 경우, optimal value 에 매우 빠르게 도달합니다.

Smaller-scale curves (32, 64, 128)는 동등하게 일관성 있게 수렴합니다.

Non-progressive training인 Figure 4(a)에서 SWD metric의 각 scale은 예상대로 일정하게 수렴됩니다.

(c) 1024 x 1024 해상도에서 raw training speed 에서 progressive growing의 효과를 나타냅니다.

Figure 4 (c) 에서 progressive growing은 해상도가 증가하면 속도가 증가합니다.

High-Resolution Image Generator using CelebA-HQ Dataset

High output resolutions에서 결과를 의미 있게 설명하기 위해서, 충분히 다양한 고품질 데이터가 필요합니다.

그러나, 이전에 GAN 논문에서 사용된 거의 모든 공개 데이터 세트는 상대적으로 low resolutions입니다.

이를 위해, 1024 x 1024 해상도의 30000개의 이미지들로 구성된 CelebA의 고품질 버전을 만들었습니다.

이 이미지는 CelebA-HQ 데이터셋을 사용해서 만든 1024 x 1024 이미지입니다.

8개의 Tesla V100 GPUs를 가지고 4일동안 train 했습니다.

해당 output resolution에 따른 adaptive minibatch size를 사용해서 memory 를 효율적으로 사용했습니다.

High-Resolution Image Generator using CelebA-HQ Dataset

PGGAN 이 loss function 에 독립적이라는 것을 설명하기 위해, WGAN-GP 대신 LSGAN loss를 사용해서 같은 네트워크를 train 했습니다.

이 이미지는 LSGAN을 사용하여 만든 이미지입니다.

WGAN-GP 를 사용하든, LSGAN 을 사용하든, 결과는 둘 다 비슷하게 나옵니다.

LUSN Results

이 이미지는 LSUN BEDROOM에서 결과를 비교한 것입니다.

LUSN Results

이것은 256 x 256 해상도의 다양한 LSUN categories에서의 결과를 보여줍니다.

Q & A

PGGAN 의 더 추가적인 정보를 알고 싶다면, 부록을 확인하면 됩니다.

질문 있나요?