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

빅데이터학과
20195168
박서정

Table

논문 선정 이유

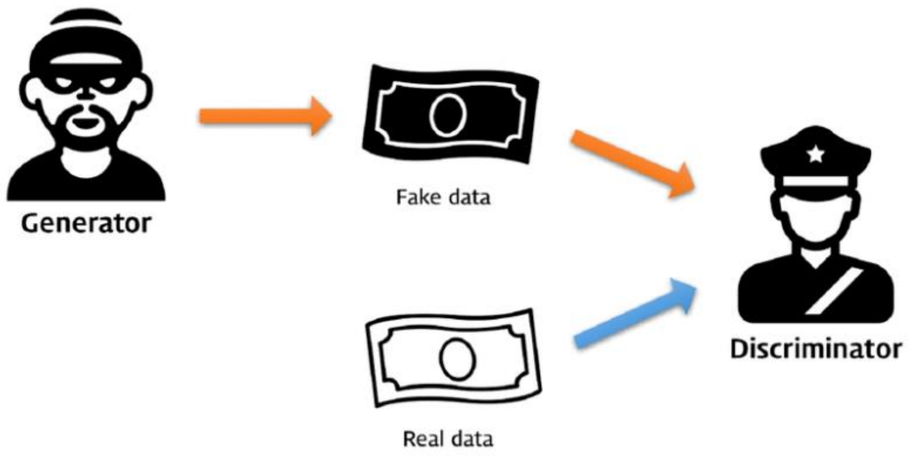
1. 인공지능의 최적화 알고리즘이 수학적으로 완벽하다는 점

2. 두 네트워크를 대치시켜 최고의 성능을 뽑아내는 GAN의 구조에 매료

3. GAN의 구조적인 문제점을 찾다가 다양한 방법론이 제시된 해당 논문 선택

I. GAN (Generative Adversarial Network)

- What is GAN ?
  - Generative Adversarial Networks (적대적 생성 네트워크) designed by Ian. J. Goodfellow
  - The training procedure for Generator is to maximize the probability of Discriminator making a mistake.



$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf

I. Previous study의 문제점

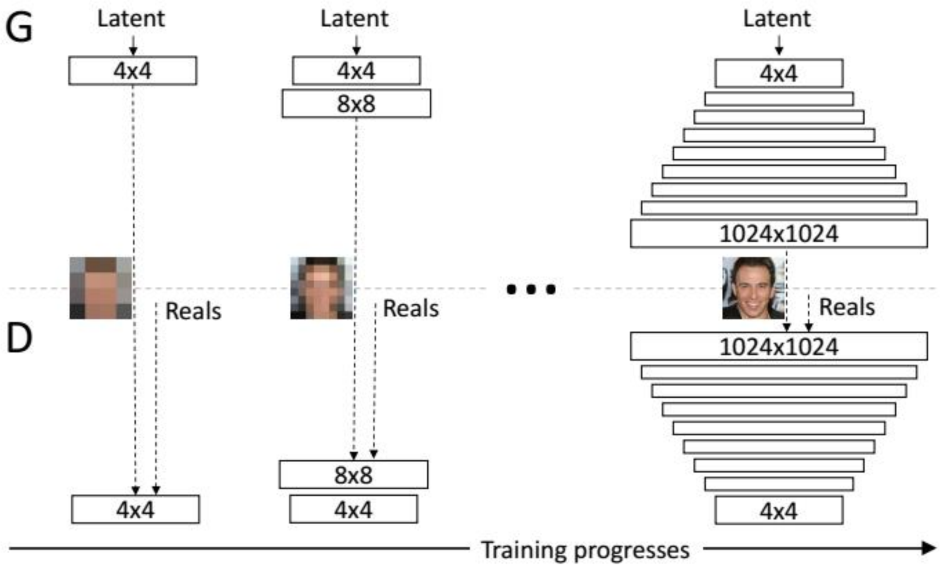
고해상도 이미지 생성 시

1. 큰 규모로 동시에 처리하게 되면 작은 영역의 특징을 부드럽게 검출해내지 못함
2. 잠재 벡터에서 고해상도로 한꺼번에 처리해 버리기 때문에 학습이 불안정함 (sudden shock)
3. 고해상도 이미지의 경우 D가 너무 쉽게 판별해 버림 (D의 성능이 G를 압도함)
4. 해상도가 크기 때문에 용량이 커져 미니배치의 크기를 작게 설정할 수 밖에 없었음

II. PGGAN (Progressive Growing GAN) Key Idea

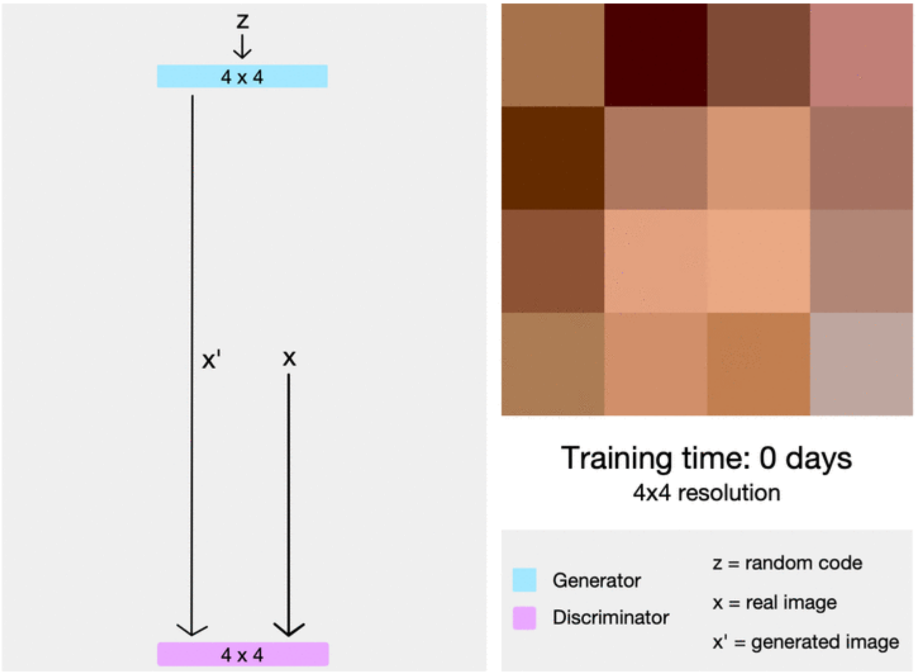
이를 해결하기 위해 해당 논문에서는

저해상도부터 시작하여 레이어를 더해감으로써 고해상도의 학습을 진행하는 구조



II. PGGAN (Progressive Growing GAN) Key Idea

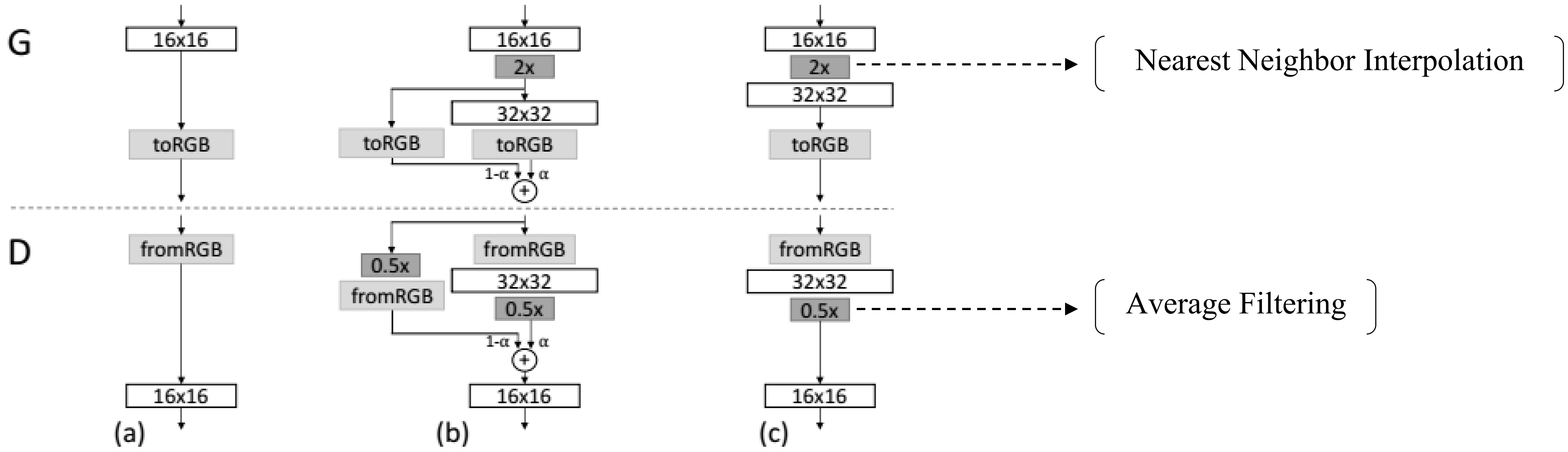
이를 해결하기 위해 해당 논문에서는 **G와 D 동시에 점진적으로 증가하는 구조**를 제시



https://hackmd.io/@_XGVS6ZYTL2p6MEHmqMvsA/HJ1BBDtP4?type=view

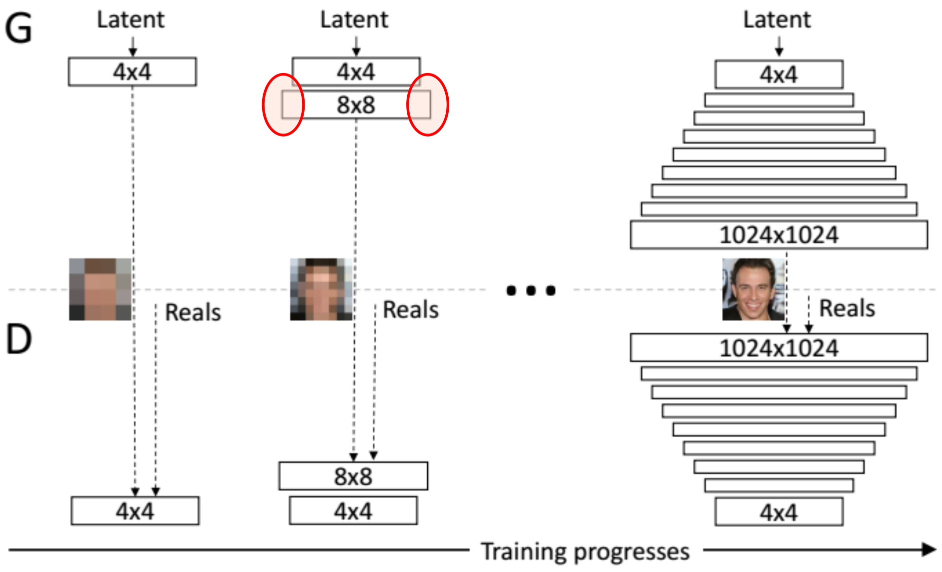
II. PGGAN (Progressive Growing GAN)

- Adding Layer



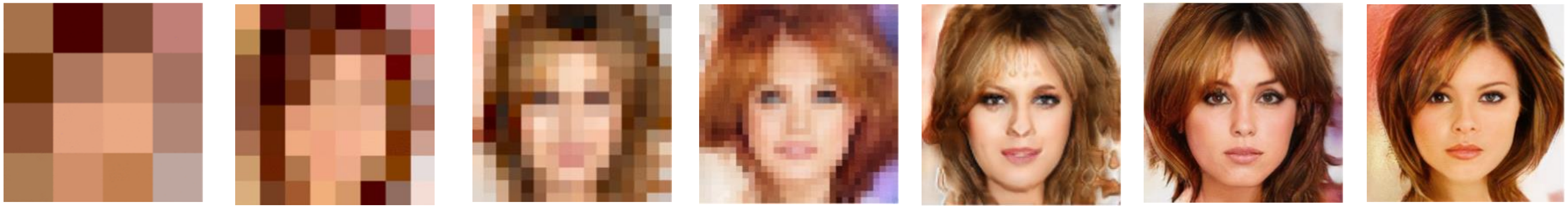
II. PGGAN Advantage

- ✓ 안정적이고 빠른 학습 가능
  - ✓ 초반에 작은 이미지에서 시작하여 적은 정보량을 추가해가며 학습을 하기 때문에 학습이 안정화 됨
  - ✓ 어려운 질문이 아닌 간단한 질문을 반복해 던지는 것과 같음



II. PGGAN Advantage

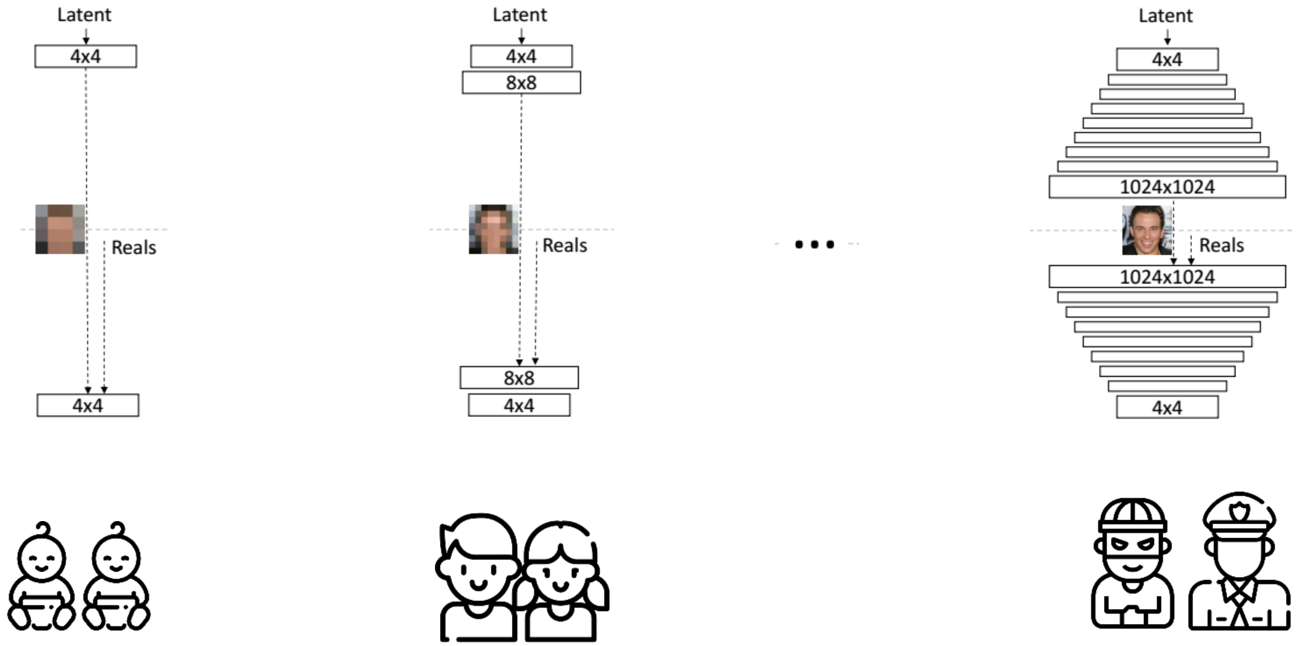
- ✓ 넓은 영역의 특징 (==작은 해상도)에서 점차 작은 영역의 특징(==고해상도)에 집중하게 됨.
  - ✓ 보다 부드러운 특징 검출이 가능함
  - ✓ 이미지의 커다란 구조를 먼저 파악하고 후에 미세한 구조를 파악



II. PGGAN Advantage

✓ 생성자와 판별자를 비슷한 정도로 함께 학습을 하기 때문에

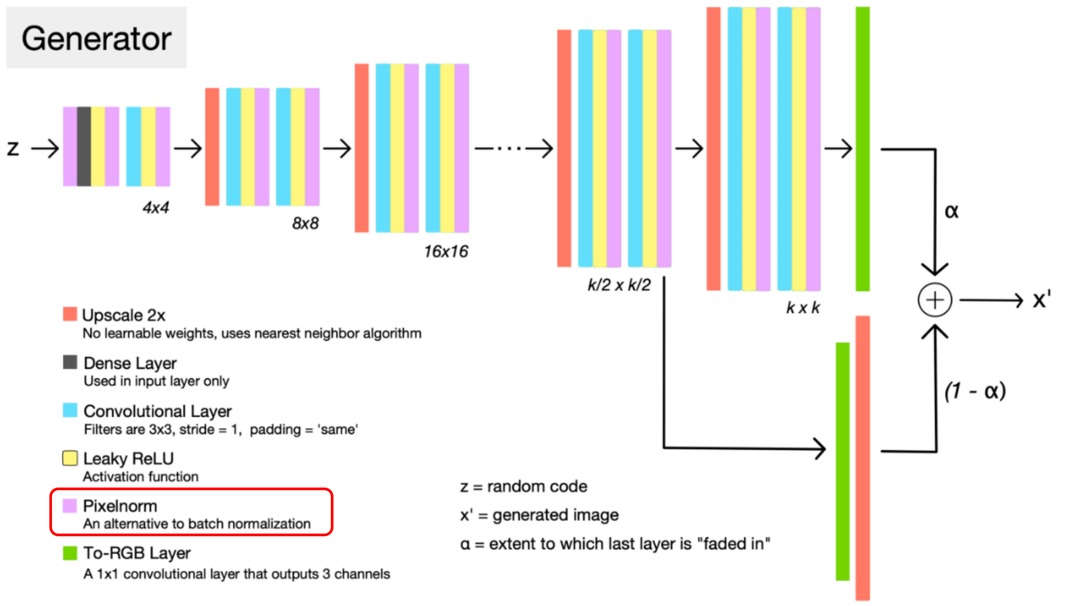
고해상도 이미지에서 판별자의 성능이 생성자의 성능을 과하게 뛰어넘는 것을 방지



III. Network Consist

- Generator

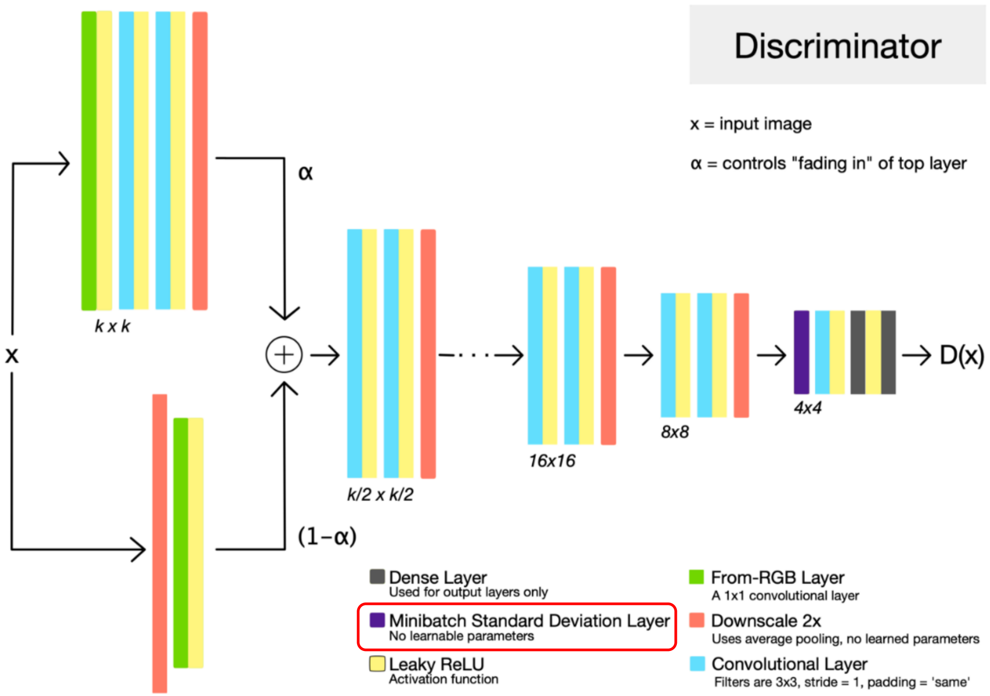
Generator	Act.	Output shape	Params
Latent vector	–	512 × 1 × 1	–
Conv 4 × 4	LReLU	512 × 4 × 4	4.2M
Conv 3 × 3	LReLU	512 × 4 × 4	2.4M
Upsample	–	512 × 8 × 8	–
Conv 3 × 3	LReLU	512 × 8 × 8	2.4M
Conv 3 × 3	LReLU	512 × 8 × 8	2.4M
Upsample	–	512 × 16 × 16	–
Conv 3 × 3	LReLU	512 × 16 × 16	2.4M
Conv 3 × 3	LReLU	512 × 16 × 16	2.4M
Upsample	–	512 × 32 × 32	–
Conv 3 × 3	LReLU	512 × 32 × 32	2.4M
Conv 3 × 3	LReLU	512 × 32 × 32	2.4M
Upsample	–	512 × 64 × 64	–
Conv 3 × 3	LReLU	256 × 64 × 64	1.2M
Conv 3 × 3	LReLU	256 × 64 × 64	590k
Upsample	–	256 × 128 × 128	–
Conv 3 × 3	LReLU	128 × 128 × 128	295k
Conv 3 × 3	LReLU	128 × 128 × 128	148k
Upsample	–	128 × 256 × 256	–
Conv 3 × 3	LReLU	64 × 256 × 256	74k
Conv 3 × 3	LReLU	64 × 256 × 256	37k
Upsample	–	64 × 512 × 512	–
Conv 3 × 3	LReLU	32 × 512 × 512	18k
Conv 3 × 3	LReLU	32 × 512 × 512	9.2k
Upsample	–	32 × 1024 × 1024	–
Conv 3 × 3	LReLU	16 × 1024 × 1024	4.6k
Conv 3 × 3	LReLU	16 × 1024 × 1024	2.3k
Conv 1 × 1	linear	3 × 1024 × 1024	51
Total trainable parameters			**23.1M**



III. Network Consist

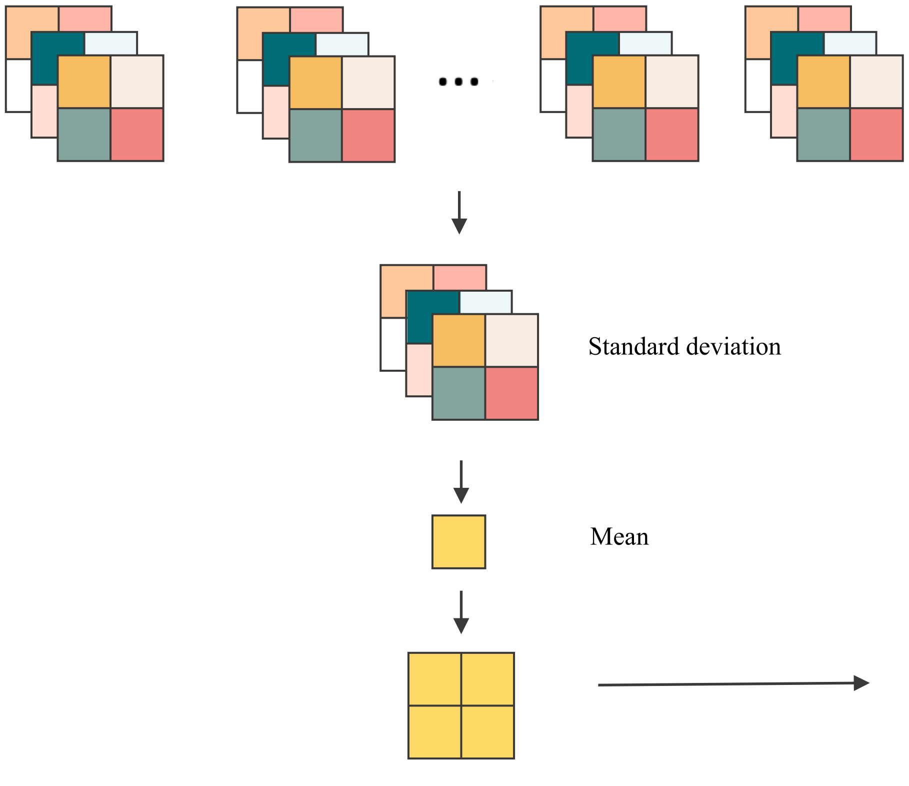
- Discriminator

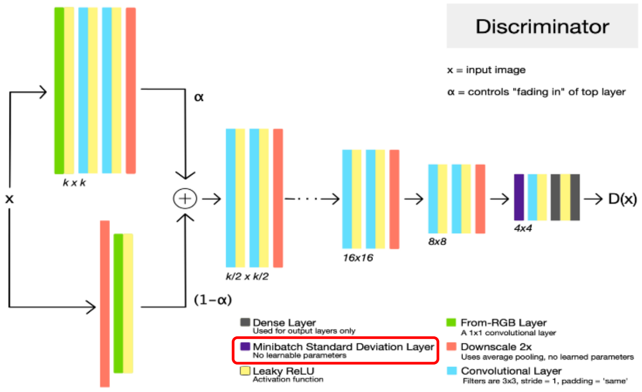
Discriminator	Act.	Output shape	Params
Input image	–	3 × 1024 × 1024	–
Conv 1 × 1	LReLU	16 × 1024 × 1024	64
Conv 3 × 3	LReLU	16 × 1024 × 1024	2.3k
Conv 3 × 3	LReLU	32 × 1024 × 1024	4.6k
Downsample	–	32 × 512 × 512	–
Conv 3 × 3	LReLU	32 × 512 × 512	9.2k
Conv 3 × 3	LReLU	64 × 512 × 512	18k
Downsample	–	64 × 256 × 256	–
Conv 3 × 3	LReLU	64 × 256 × 256	37k
Conv 3 × 3	LReLU	128 × 256 × 256	74k
Downsample	–	128 × 128 × 128	–
Conv 3 × 3	LReLU	128 × 128 × 128	148k
Conv 3 × 3	LReLU	256 × 128 × 128	295k
Downsample	–	256 × 64 × 64	–
Conv 3 × 3	LReLU	256 × 64 × 64	590k
Conv 3 × 3	LReLU	512 × 64 × 64	1.2M
Downsample	–	512 × 32 × 32	–
Conv 3 × 3	LReLU	512 × 32 × 32	2.4M
Conv 3 × 3	LReLU	512 × 32 × 32	2.4M
Downsample	–	512 × 16 × 16	–
Conv 3 × 3	LReLU	512 × 16 × 16	2.4M
Conv 3 × 3	LReLU	512 × 16 × 16	2.4M
Downsample	–	512 × 8 × 8	–
Conv 3 × 3	LReLU	512 × 8 × 8	2.4M
Conv 3 × 3	LReLU	512 × 8 × 8	2.4M
Downsample	–	512 × 4 × 4	–
Minibatch stddev	–	513 × 4 × 4	–
Conv 3 × 3	LReLU	512 × 4 × 4	2.4M
Conv 4 × 4	LReLU	512 × 1 × 1	4.2M
Fully-connected	linear	1 × 1 × 1	513
Total trainable parameters			**23.1M**

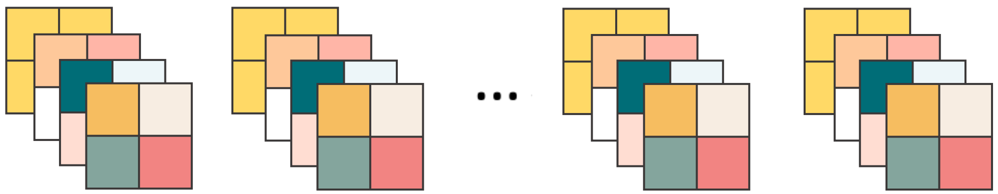


III. Network Consist

- Minibatch Standard Deviation (in D) → Increasing Variation







III. Network Consist

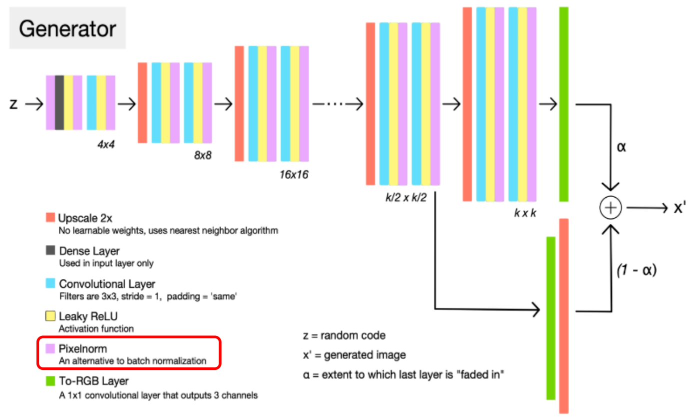
- Pixel Normalization (in G)

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

$a_{x,y}$: Origin pixel

$b_{x,y}$: pixel after Norm

N : Number of feature map



→ Prevent the escalation of Signal magnitudes between G and D

IV. Result

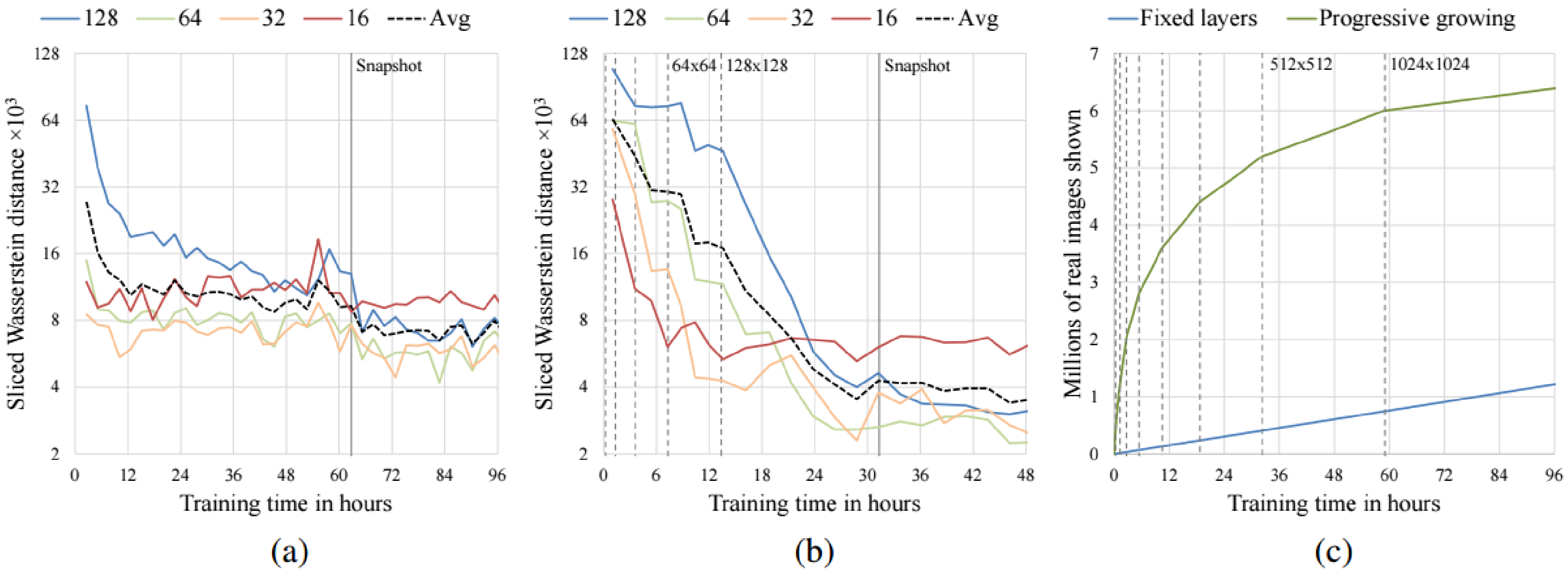
Training configuration	CELEBA Sliced Wasserstein distance $\times 10^3$ 128	64	32	16	Avg	MS-SSIM	LSUN BEDROOM Sliced Wasserstein distance $\times 10^3$ 128	64	32	16	Avg	MS-SSIM
(a) Gulrajani et al. (2017)	12.99	7.79	7.62	8.73	9.28	0.2854	11.97	10.51	8.03	14.48	11.25	**0.0587**
(b) + Progressive growing	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

* MS $-$ $SSIM$: (SSIM + scale) 다중 스케일 정보를 결합한 SSIM (인간의 눈으로 보았을 때와 비슷한 관점으로 평가)

* $SSIM(x,y) = l(x,y)c(x,y)s(x,y)$: 두 이미지의 구조적 유사도를 평가

IV. Result



Not using
PG−layer

Using
PG−layer

IV. Result

Unsupervised		
Method		**Inception score**
ALI	(Dumoulin et al., 2016)	5.34 ± 0.05
GMAN	(Durugkar et al., 2016)	6.00 ± 0.19
Improved GAN	(Salimans et al., 2016)	6.86 ± 0.06
CEGAN-Ent-VI	(Dai et al., 2017)	7.07 ± 0.07
LR-AGN	(Yang et al., 2017)	7.17 ± 0.17
DFM	(Warde-Farley & Bengio, 2017)	7.72 ± 0.13
WGAN-GP	(Gulrajani et al., 2017)	7.86 ± 0.07
Splitting GAN	(Grinblat et al., 2017)	7.90 ± 0.09
Our (best run)		$\mathbf{8.80 \pm 0.05}$
Our (computed from 10 runs)		$\mathbf{8.56 \pm 0.06}$

Label conditioned		
Method		**Inception score**
DCGAN	(Radford et al., 2015)	6.58
Improved GAN	(Salimans et al., 2016)	8.09 ± 0.07
AC-GAN	(Odena et al., 2017)	8.25 ± 0.07
SGAN	(Huang et al., 2016)	8.59 ± 0.12
WGAN-GP	(Gulrajani et al., 2017)	8.67 ± 0.14
Splitting GAN	(Grinblat et al., 2017)	8.87 ± 0.09

Table 3: CIFAR10 inception scores, higher is better.

* *Inception Score* : 생성된 이미지의 품질과 다양성을 기준으로 평가
두 기준이 모두 우수하다면 큰 값

Any Question?

Thank you

Tel : 010-6725-2073

e-mail : diditjwjd@gmail.com

Kakaotalk : tjwjd0909