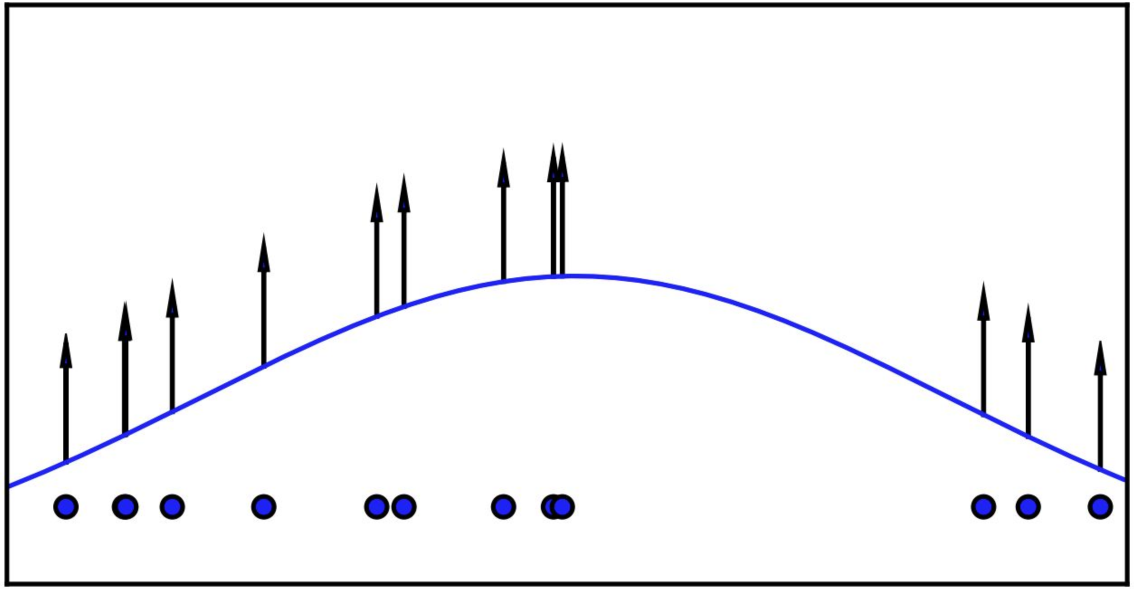
Generative Adversarial Network

Instructor: Seunghoon Hong

Course logistics

- New assignment will be out today
  - **Deadline:** 23:59:59 November 22th
  - **Quiz:** Nov. 27th

Recap: objective of generative models



$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{x \sim p_{\text{data}}} \log p_{\theta}(x)$$

Figure credit: Ian Goodfellow

Recap: Autoregressive models

- Explicit optimization of likelihood based on chain rule

$$\log p_\theta(x) = \sum_{t=1}^{d} \log p_\theta(x^t | x^1, \ldots, x^{t-1})$$

For d-dimensional data $x = (x^1, x^2, \ldots, x^d)$

- Advantages:
    - Optimizing exact likelihood

- Disadvantages:
    - Sequential generation process (O(n))
    - No latent variable to control the generation process

Recap: Variational Autoencoder

- Explicit optimization of variational lower-bound of likelihood

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z))$$

- Advantage:
  - Latent variable model (i.e. we can control generation via z)
  - Feedforward generation
- Disadvantages:
  - Lower-bound optimization (there is a gap between actual likelihood and lower-bound if q≠p).
  - Generally not satisfactory generation qualtiy

Explicit density models

- Explicitly modeling the likelihood of data
- Autoregressive models and VAEs are both explicit density models

Autoregressive models

$$\log p_\theta(x) = \sum_{t=1}^{d} \log p_\theta(x^t | x^1, \ldots, x^{t-1})$$

Variational Autoencoder

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x) || p(z))$$

Challenges in explicit density models

- What if we cannot measure the likelihood?
- Example: conditional generation (e.g. machine translation)
  - In some cases we do not have a paired data
  - Example: unaligned data for machine translation → pair가 없어 likelihood measure 불가능.
  - In this case, we cannot measure the conditional probability

Implicit density model

- Modeling distribution without explicit likelihood estimation

Today’s agenda

- Generative Adversarial Network (GAN)

↱ End: Equilibrium (50%).

Generation task as an adversarial game

Note: money의 quality는 측정하지 않음 (D).

(위조지폐)

- Intuitive example: a game of counterfeiting

Adversarial game between two players

위조지폐 탐지할 정도만 됨.

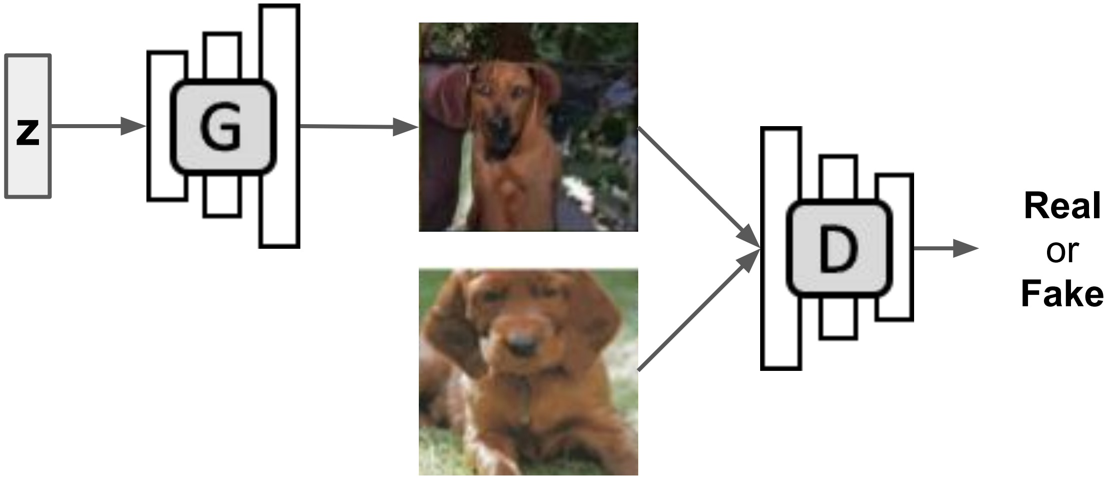
Goal of counterfeiter
Make fake money as realistic as possible
경찰 속일정도로 만들면 충분
Generator

즉, 이 위조지폐의 goodness보다는 경찰 속이기에 집중.

Goal of police officer
Detect counterfeit money from real one
Discriminator

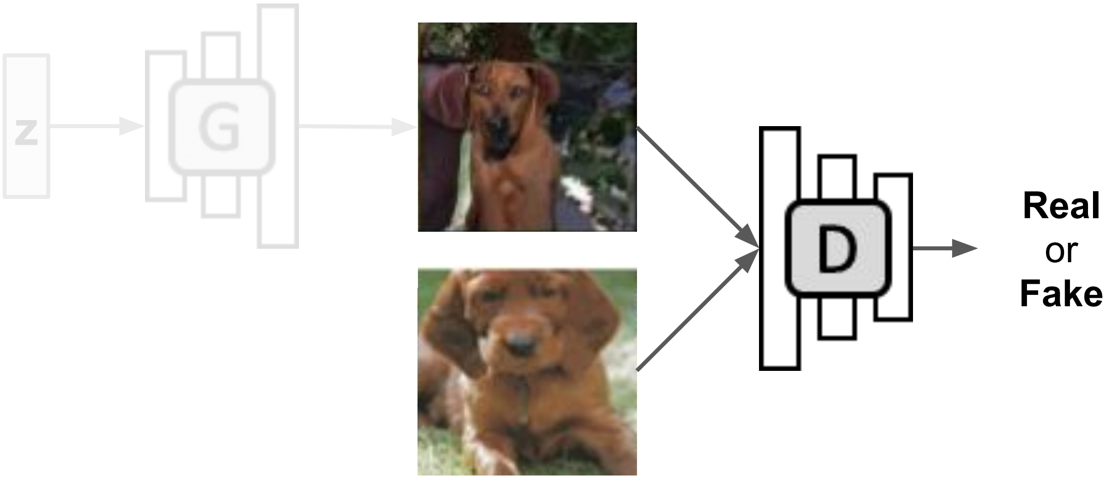
Generative Adversarial Network (GAN)

- Learning to generate via minimax optimization



Generative Adversarial Network (GAN)

- Learning to generate via minimax optimization

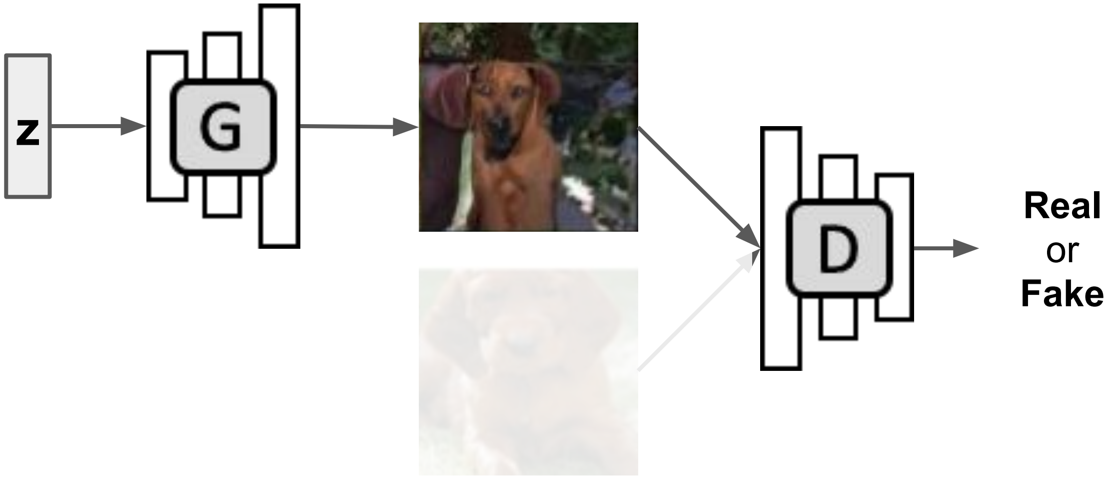


- **Discriminator** (D) tries to tell if its input is real or fake

즉, binary classification.

Generative Adversarial Network (GAN)

- Learning to generate via minimax optimization



- **Discriminator** (D) tries to tell if its input is real or fake

- **Generator** (G) tries to fool the discriminator

Learning objective

- Minimax objective function

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \underbrace{\log D_{\theta_d}(x)}_{\text{Discriminator output for real input x}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for the generator output G(z)}}) \right]$$

Given a generator $G(z; \theta_g)$

- Discriminator tries to maximize the objective such that
  D(x) is close to 1 (real) and D(G(z)) is close to 0 (fake) → solves binary classification problem

Given a discriminator $D(x; \theta_d)$

- Generator tries to minimize objective such that D(G(z)) is close to 1
  (discriminator is fooled into thinking generated G(z) is real)

Learning objective

- Minimax objective function

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Gradient descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

Optimization challenge

Alternate between:

1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

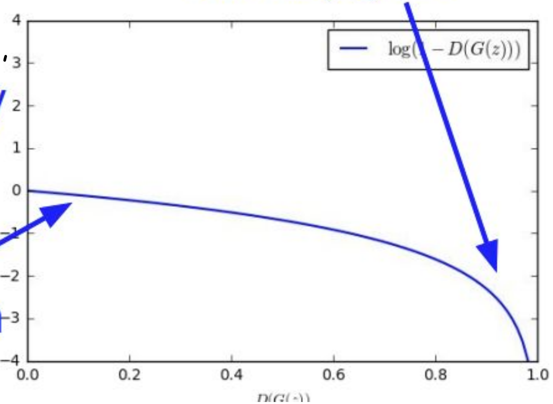
2. **Gradient descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

G가 0에 가까우면 더
학습해야하는데
gradient가 작아버린다.

When sample is likely fake, want to learn from it to improve generator. But gradient in this region is relatively flat!

Gradient signal dominated by region where sample is already good



Optimization challenge

Alternate between:

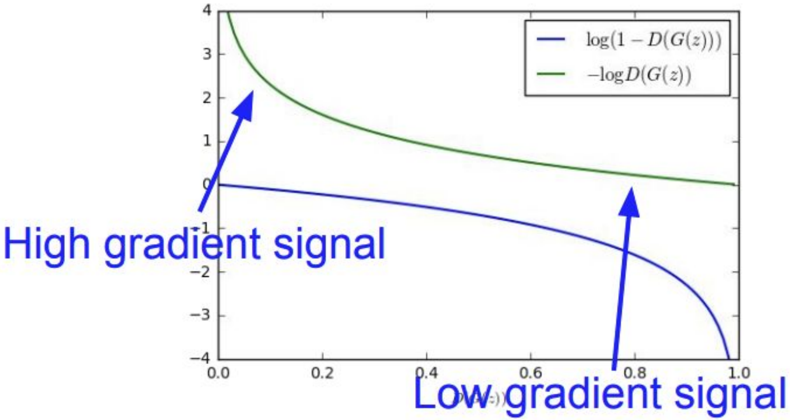
1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Gradient** ascent on generator but different using different objective

$$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$

Also known as non-saturating loss



nition

GAN training algorithm

for number of training iterations **do**

what it means?

for k steps **do**

즉 batch size 2m.

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by ascending its stochastic gradient (improved objective):

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

end for

The very first GAN results

Get sharp image.

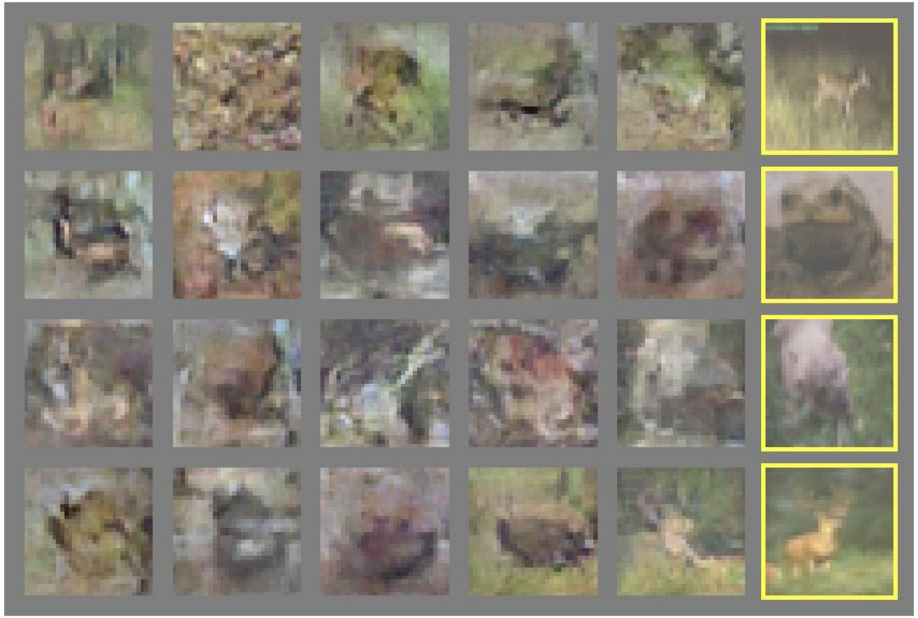
Goodfellow et al., Generative adversarial networks, In NIPS, 2014

Synthesized images by the generator

Nearest neighbors in a training set
(of the rightmost generated example)

The very first GAN results

Goodfellow et al., Generative adversarial networks, In NIPS, 2014



Case study: DCGAN

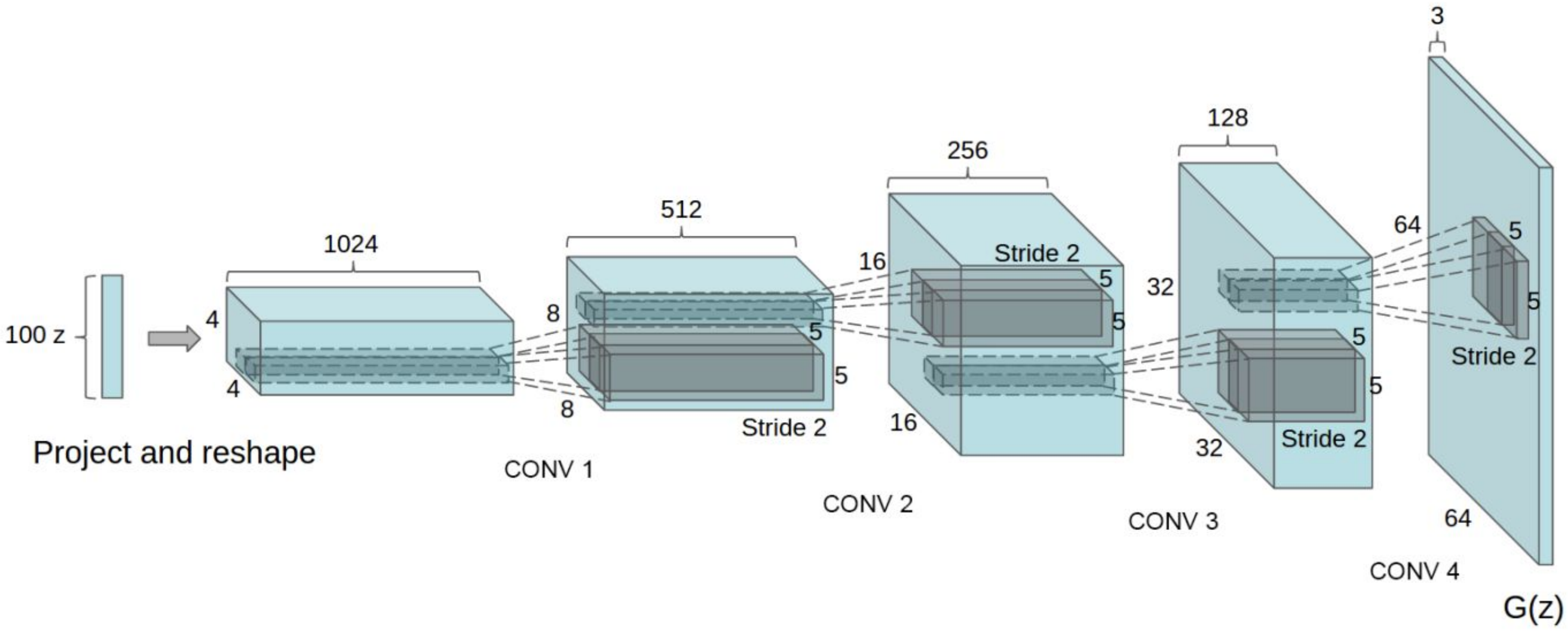
- Techniques to improve GAN training

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

Case study: DCGAN

- Generator architecture



Radford et al., unsupervised representation learning with deep convolutional generative adversarial networks, In ICLR, 2016

Generated images - LSUN bedroom dataset

Radford et al., unsupervised representation learning with deep convolutional generative adversarial networks, In ICLR, 2016

Sample interpolation

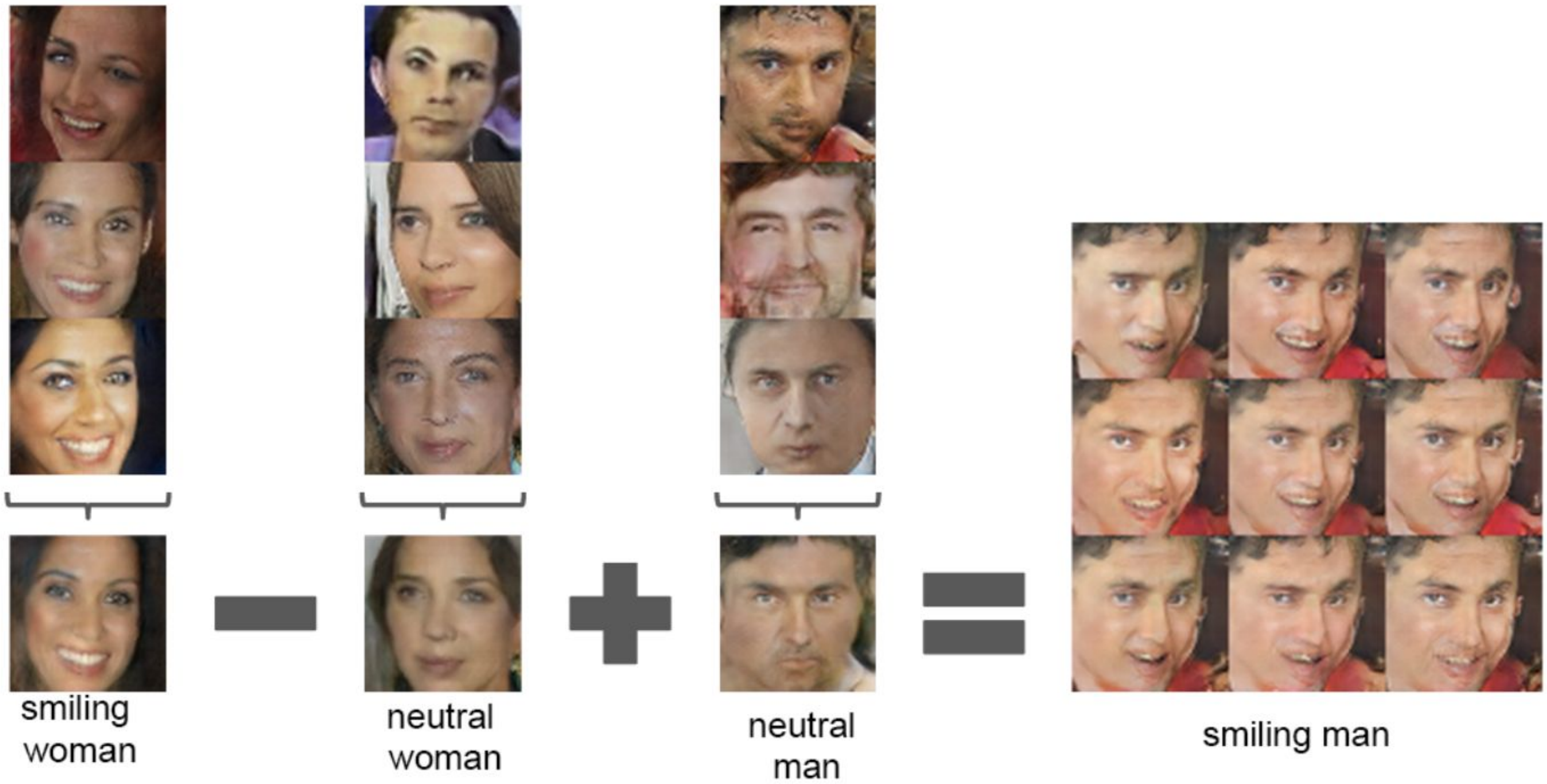
z_1 interpolate z_2

Radford et al., unsupervised representation learning with deep convolutional generative adversarial networks, In ICLR, 2016

Sample interpolation

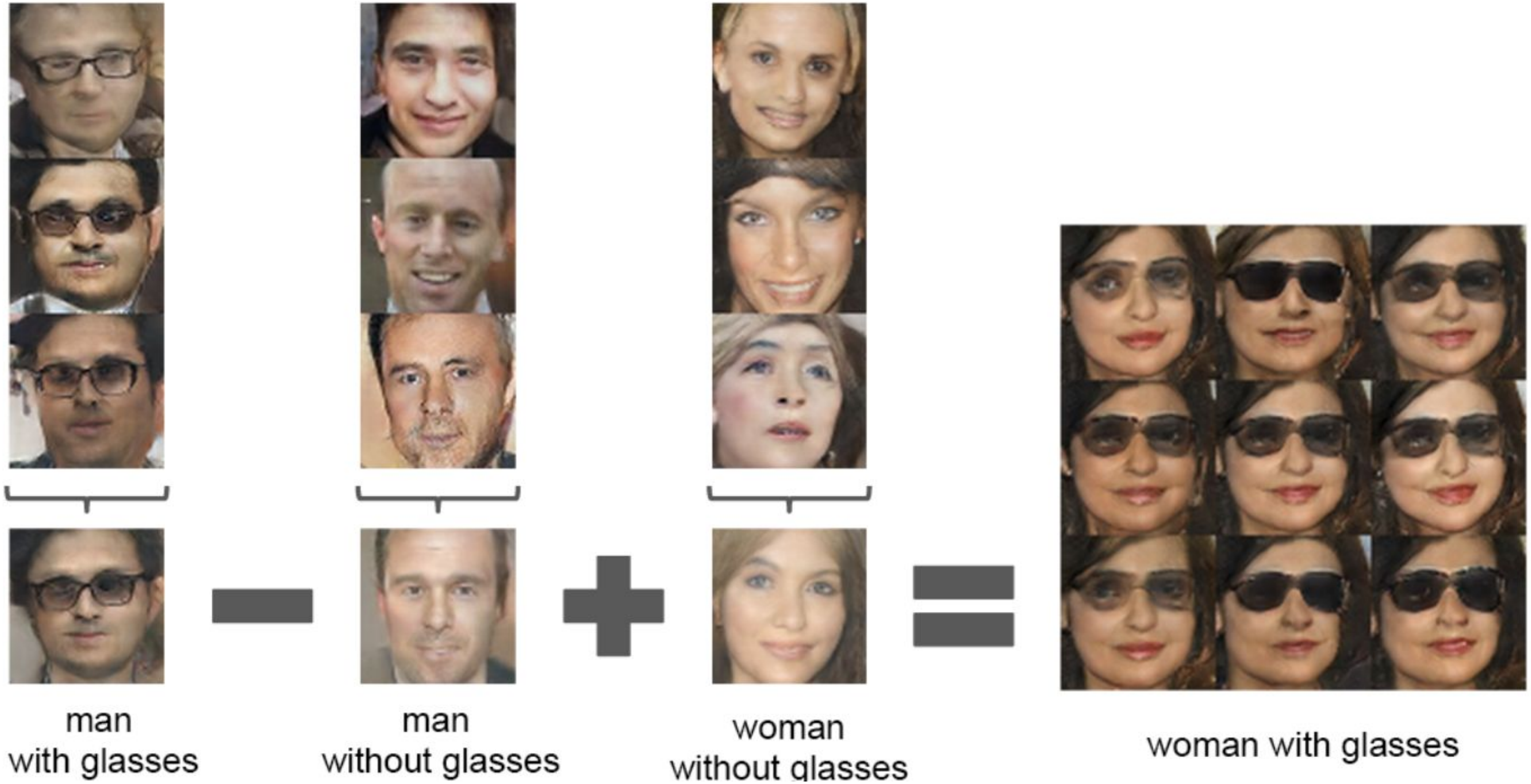
Radford et al., unsupervised representation learning with deep convolutional generative adversarial networks, In ICLR, 2016

Arithmetic on latent variable



Radford et al., unsupervised representation learning with deep convolutional generative adversarial networks, In ICLR, 2016

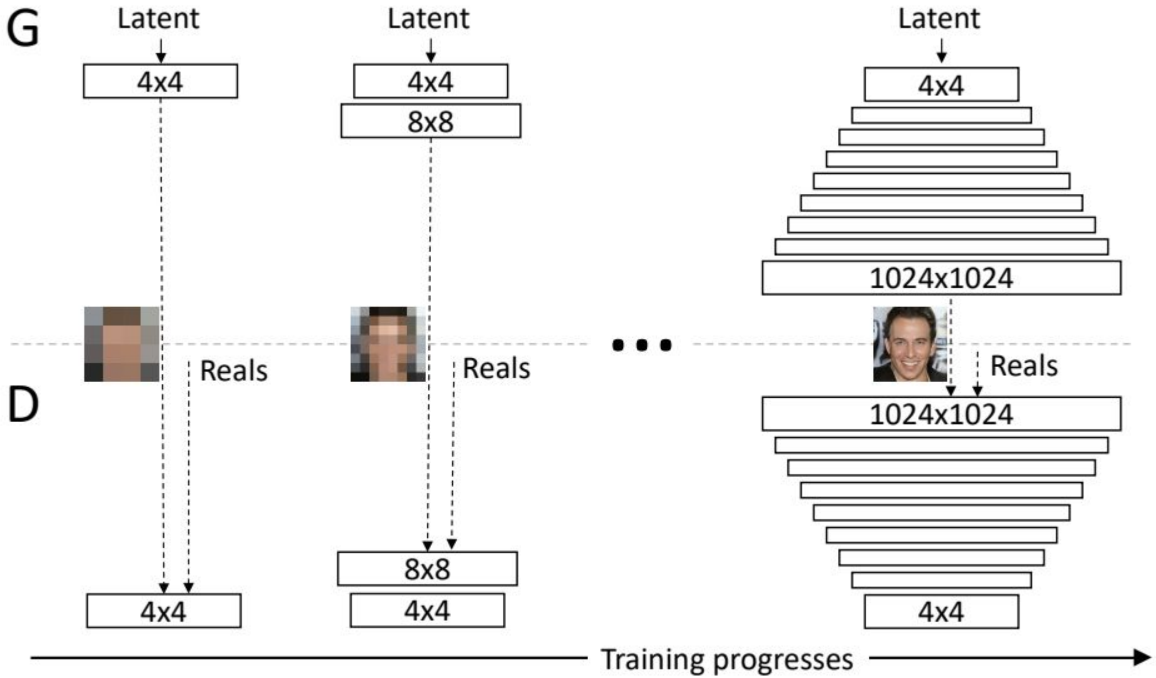
Arithmetic on latent variable



Radford et al., unsupervised representation learning with deep convolutional generative adversarial networks, In ICLR, 2016

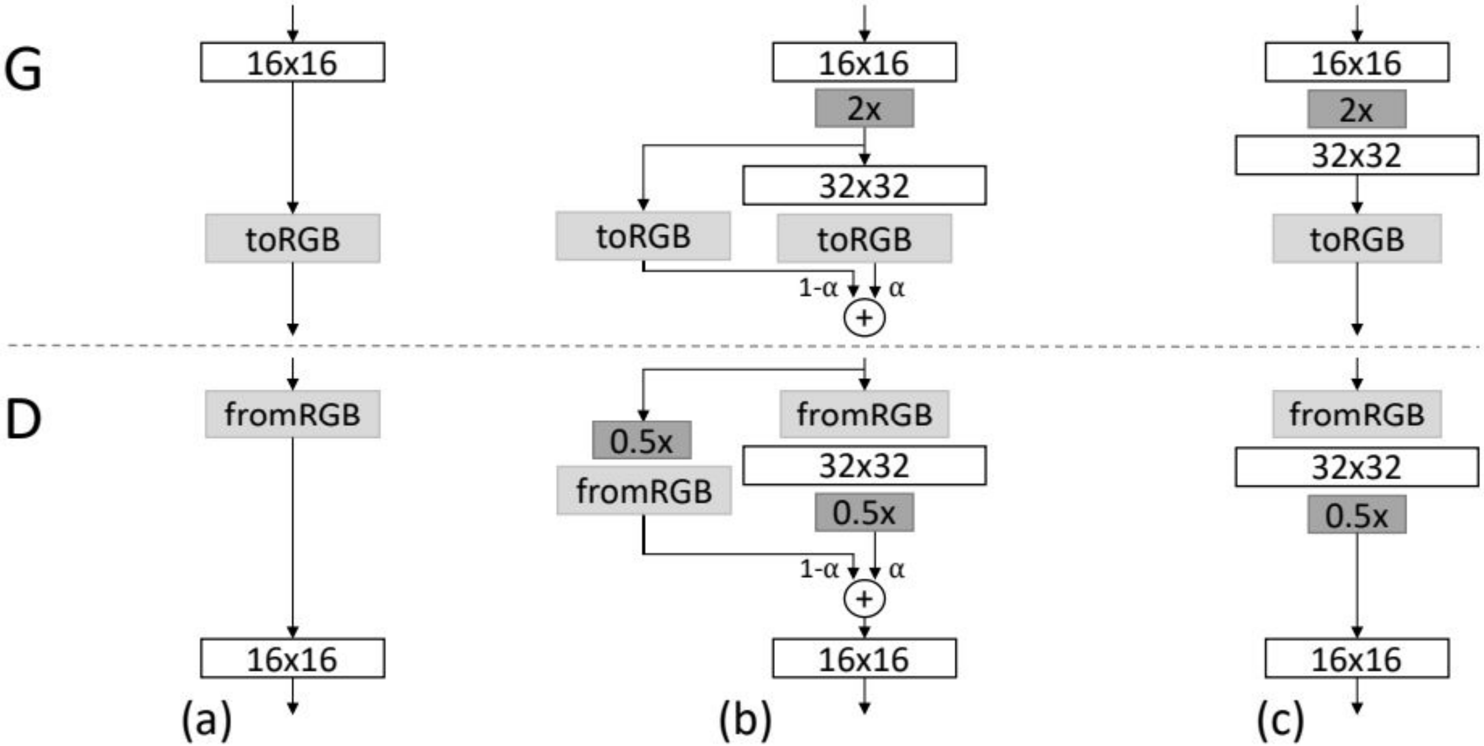
Progressive growing of GAN

- Improve the generation quality through hierarchical generation



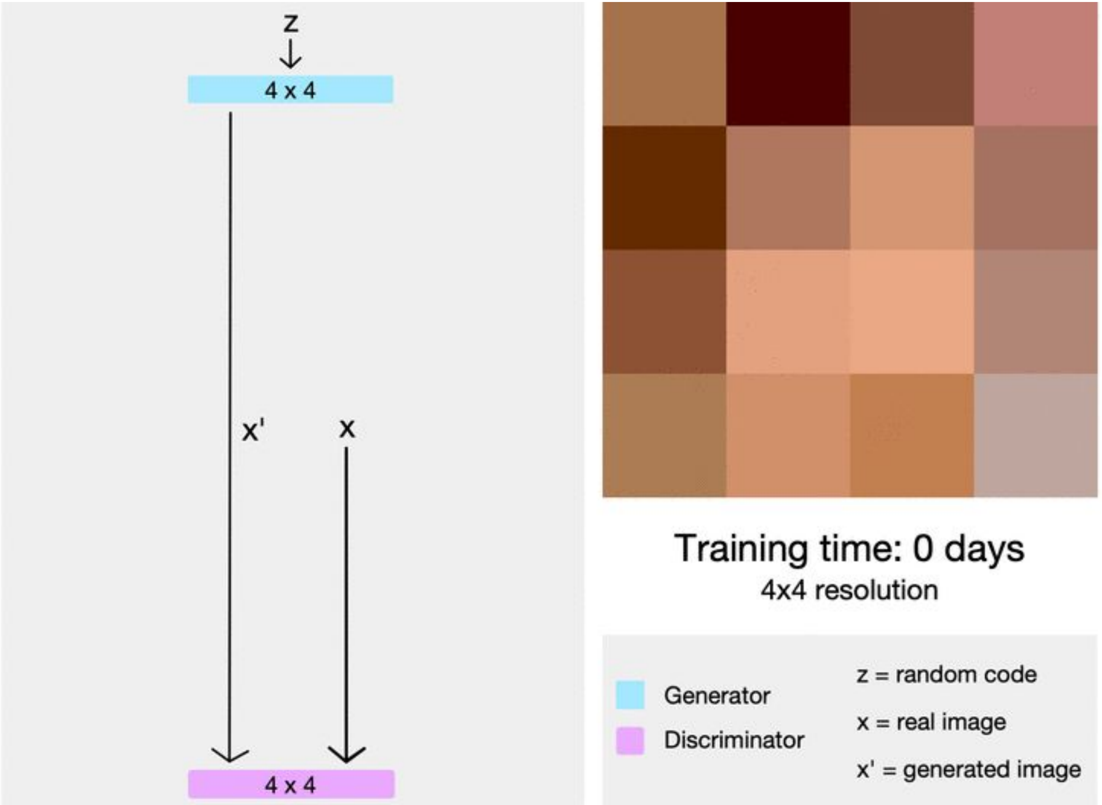
Kerras et al., Progressive Growing Of Gans For Improved Quality, Stability, And Variation, In ICLR, 2018

Progressive growing of GAN



Kerras et al., Progressive Growing Of Gans For Improved Quality, Stability, And Variation, In ICLR, 2018

Progressive growing of GAN



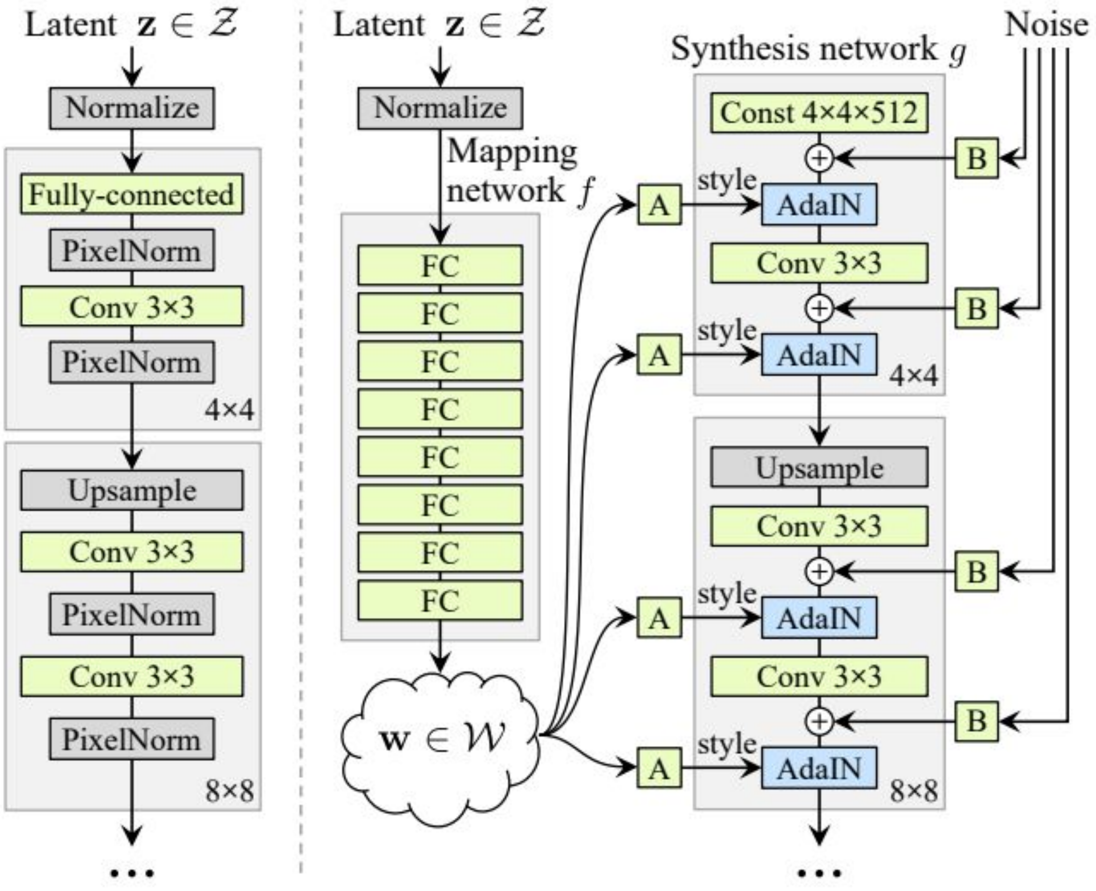
lamge source: https://hackmd.io/@_XGVS6ZYTL2p6MEHmqMvsA/HJ1BBDtP4?type=view

BigGAN

Brock et al., Large Scale GAN Training for High Fidelity Natural Image Synthesis, In ICLR, 2019

StyleGAN

- Improved version of progressive GAN
- Deep embedding layers of latent variable
- Injecting latent variable via modulation



(a) Traditional (b) Style-based generator

Kerras et al., A Style-Based Generator Architecture for Generative Adversarial Networks, In CVPR, 2019

StyleGAN: results

Kerras et al., A Style-Based Generator Architecture for Generative Adversarial Networks, In CVPR, 2019

StyleGAN: sample interpolation

Kerras et al., A Style-Based Generator Architecture for Generative Adversarial Networks, In CVPR, 2019

StyleGAN: try yourself

https://thispersondoesnotexist.com/

Challenges in GAN

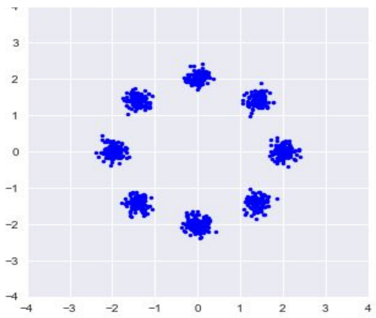
- Is adversarial game stable?

Challenges in GAN

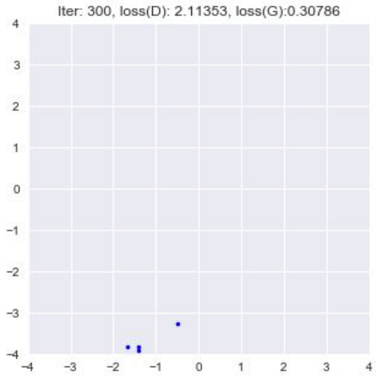
combine modes. blurred images

- Is adversarial game stable? → it is turned out to be not
- One representative problem: **mode-collapse** (It does not happen in autoencoder)

True Data



GAN output



When mode-collapse happens,
the generator models **only part of** the true data distribution (e.g. one data mode)

Why does it happens?

→ intuitively, a single mode is also indistinguishable from the real data