Generative Adversarial Networks (GAN)

Presentor: Jinduk park

Contents

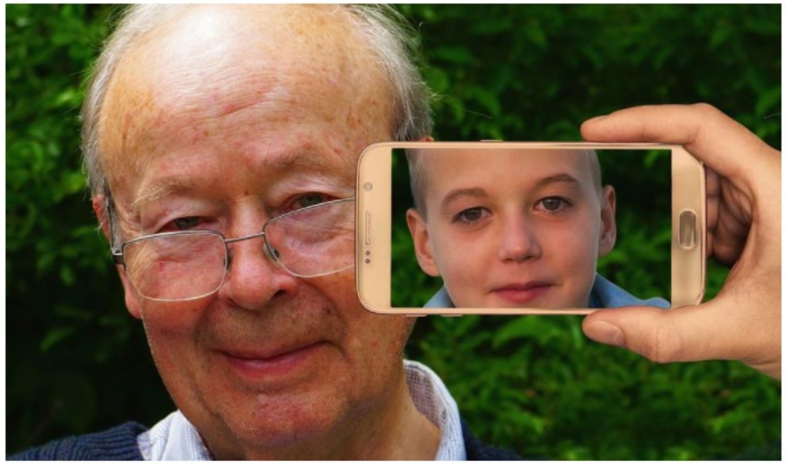
Part1. Introduction to generative model

Generative model: applications 1

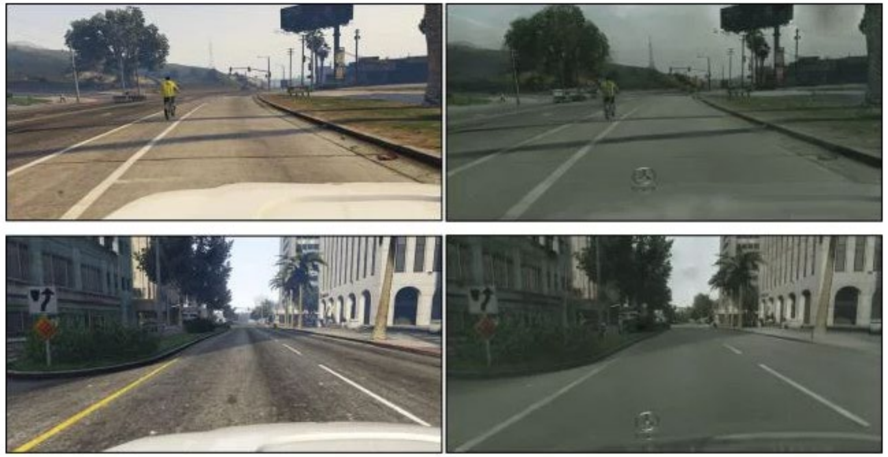
Let's watch some short *Deepfake* video
made by generative model :

https://www.youtube.com/watch?v=A8TmqvTVQFQ

Generative model: applications 2



De-aging



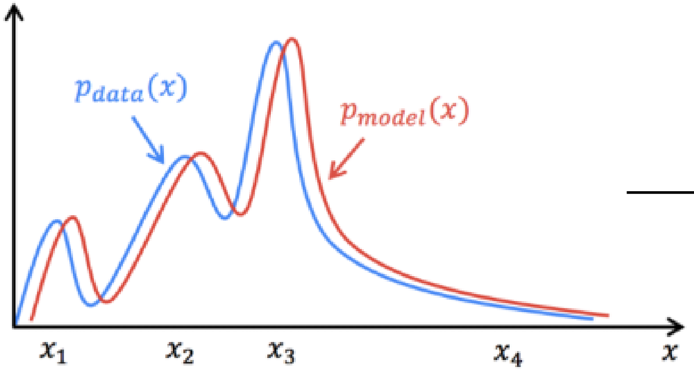
Generate realistic data from simulation
(GTA game image)

Anime avatar generation

images from: https://blog.eduonix.com/artificial-intelligence/grand-finale-applications-gans/

What is generative model?

data



Model learns
$p_{data}(x)$

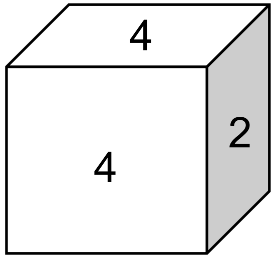
Sampled data
from learnt distribution
$p_{model}(x)$

Generative model try to generate new samples from the same distribution as original data

What exactly data distribution p(x) is ?

Simple example with a bit weird dice that has two '4' planes: {1,2,3,4,4,5}.

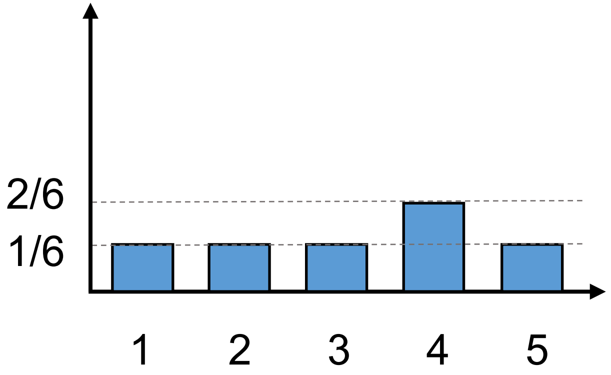
Define event :



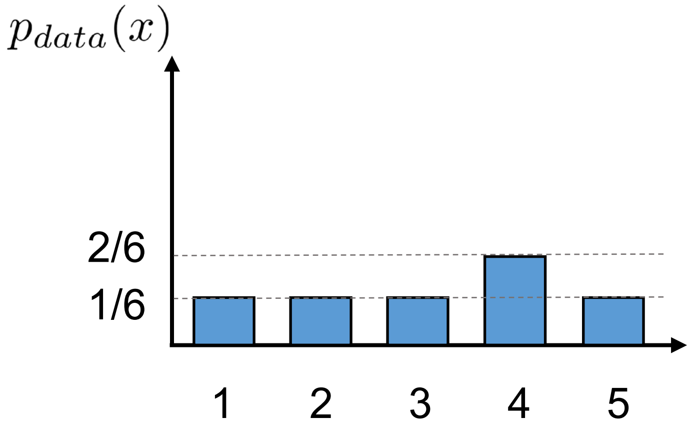
Roll the dice ! $\longrightarrow$ $\boxed{4}$ One discrete value will appear.

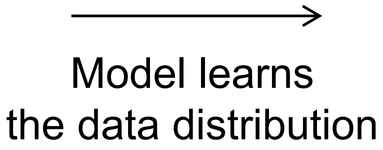
In this case,

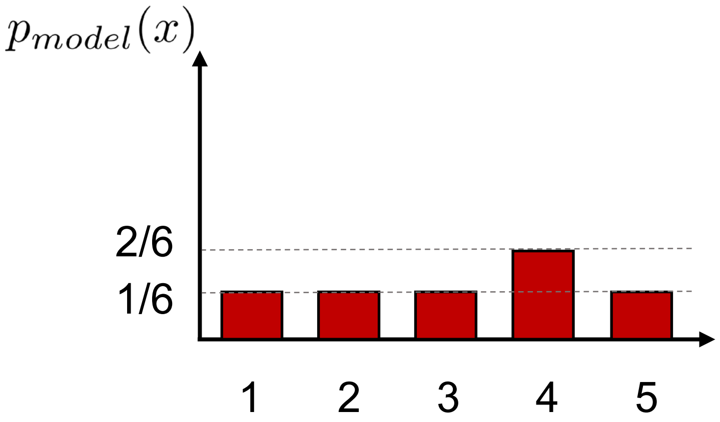
$p_{data}(x)$



What exactly data distribution p(x) is ?







If the model perfectly learns the data distribution,
samples from the probability distribution would be like:

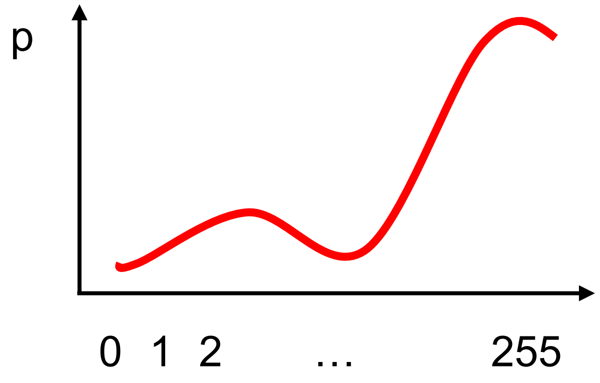
10000 sampling,

1: 1664
2: 1668
3: 1662
4: 3332
5: 1670

Data distribution for multi-dimensional (image) data

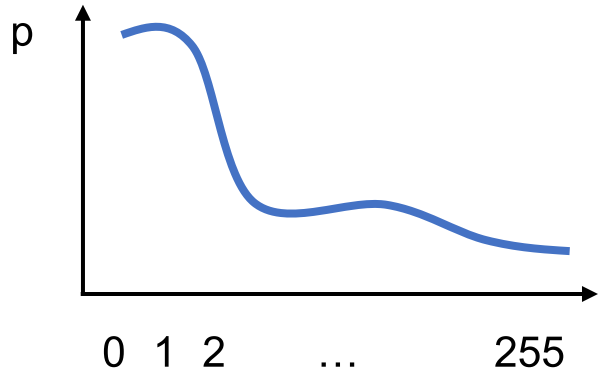
6 data instances to learn distribution

$p_{data}(x)$ at ☐ 1-pixel



The pixel mostly contains bright value (high pixel value)

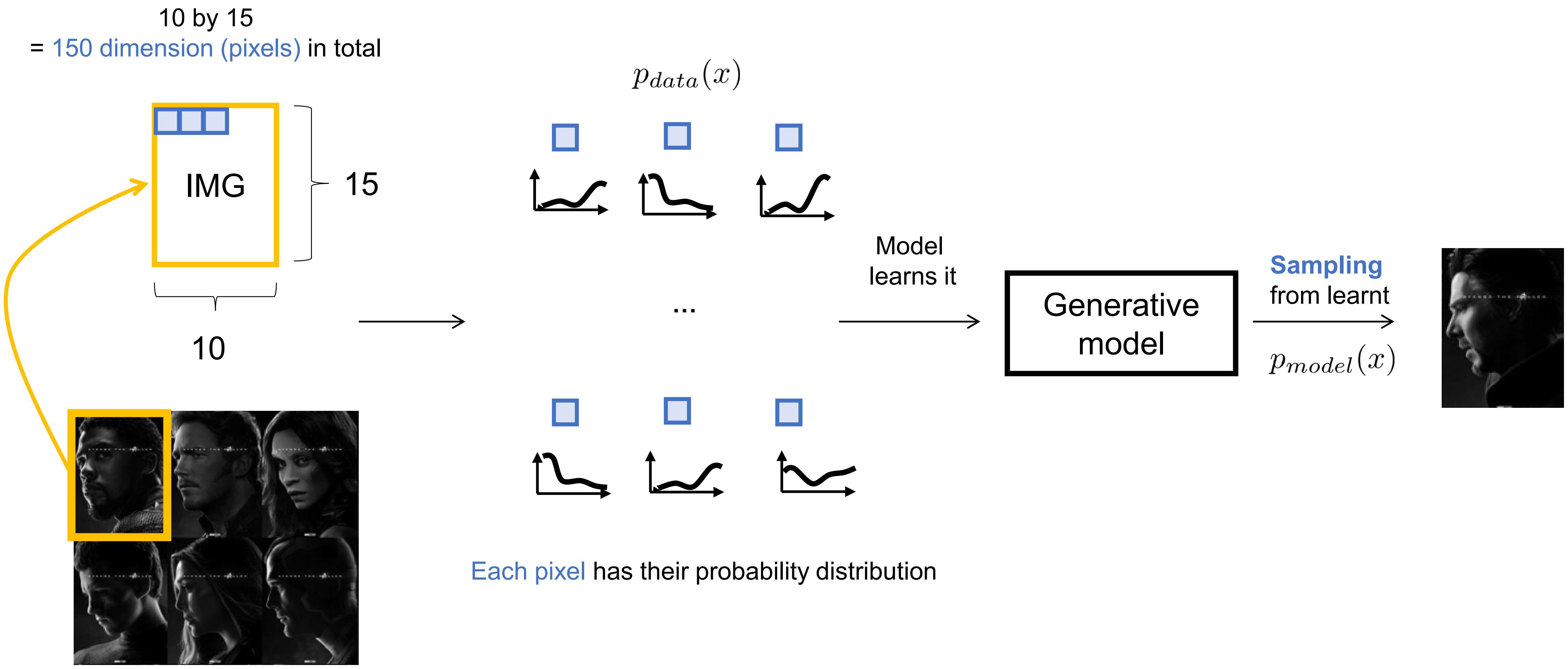
$p_{data}(x)$ at ☐ 1-pixel



The pixel mostly contains dark value (low pixel value)

images from: https://www.goldderby.com/feature/avengers-endgame-32-character-posters-photo-gallery-1202830589/

Data distribution for multi-dimensional (image) data

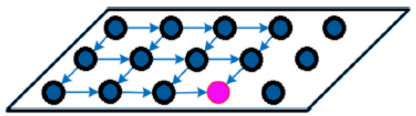


Taxonomy of generative model

Based on tractability of probability density function..

- Explicit generative model defines and solves for $p_{model}(x)$

  e.g.) pixelRNN *



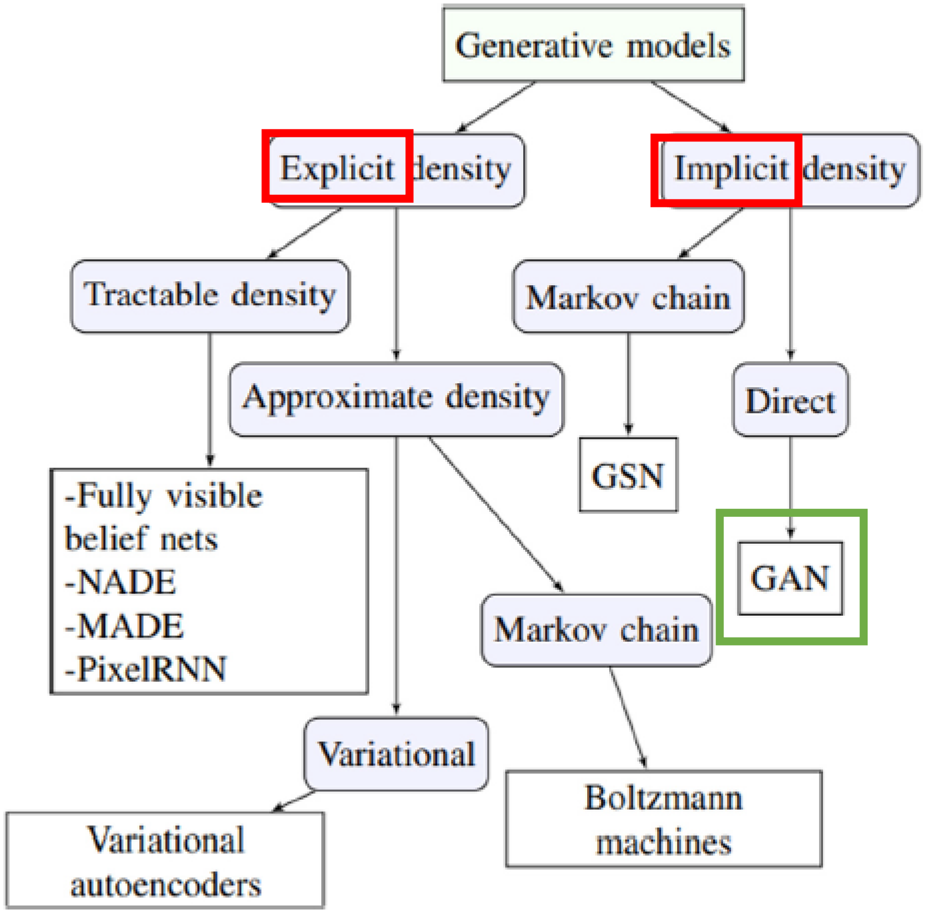
$$p(x) = \prod_{i=1}^{n} p(x_i | x_1, ..., x_{i-1})$$

Likelihood of image x

Probability of i'th pixel value given all previous pixels

- Implicit generative model samples from $p_{model}(x)$ without explicitly defining it.

  : GAN is implicit generative model !



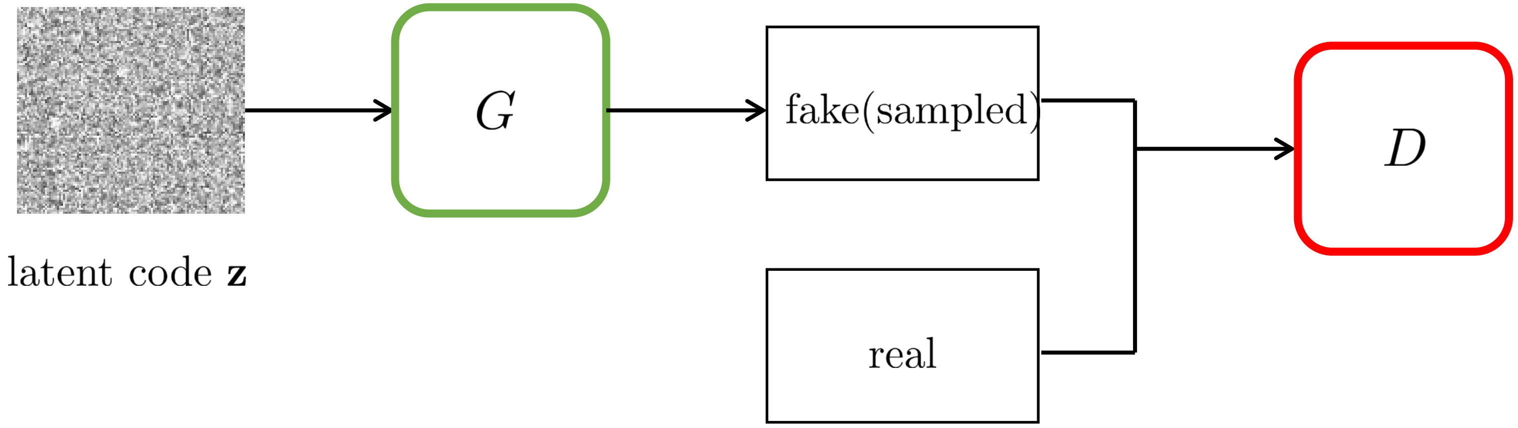
Goodfellow et al.

* Van Oord, Aaron, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel recurrent neural networks." *International Conference on Machine Learning*. PMLR, 2016.

Part 2. Generative Adversarial Networks (GAN)

Referenced this link to some extent:
Youtube “1시간만에 GAN(Generative Adversarial Network) 완전 정복하기”, naver d2,
https://www.youtube.com/watch?v=odpjk7_tGY0

Basic concept of GAN

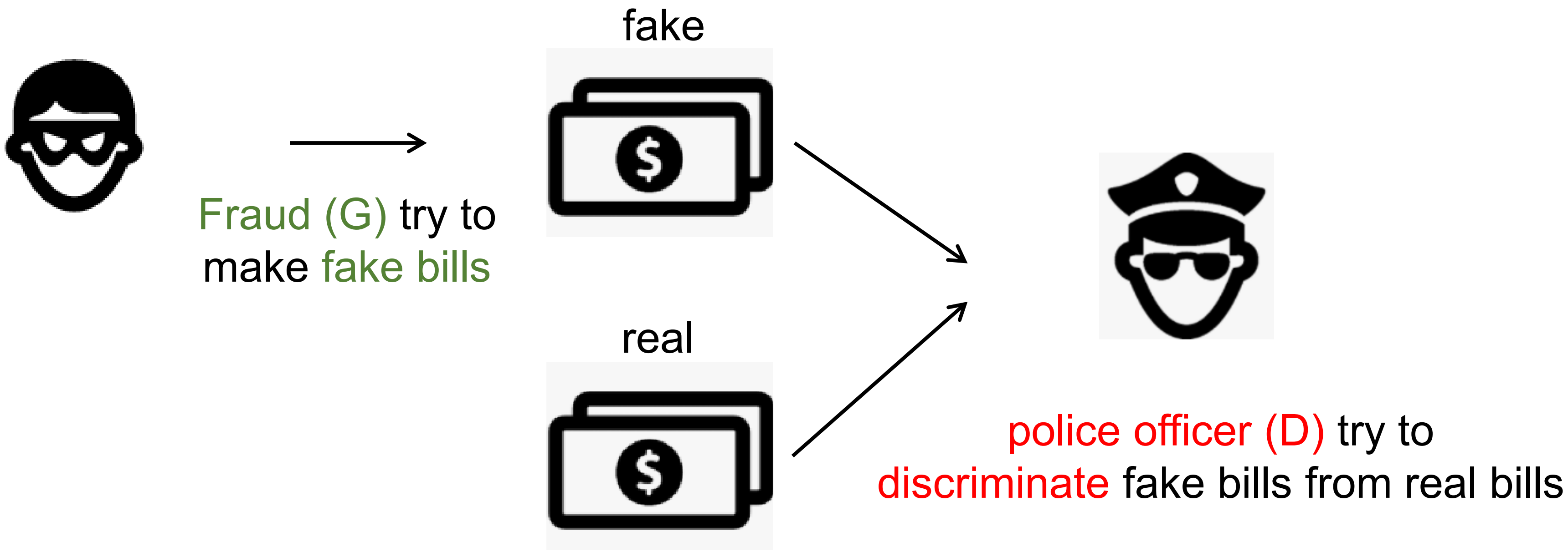


- Adversarial concept of GAN

  Two different networks compete each other,
  one (generator) tries to generate fake example that almost same as real one and
  the other (discriminator) tries to distinguish fake from real one.

A famous metaphor for GAN

- A famous metaphor for GAN : fraud and police officer



They both (~ G, D) train (~ optimize) to improve their ability (~ objective).

Objective function for GAN

Two player minimax game

$$L = min_G max_D \mathbb{E}_{x \sim p_{real}} log D(x) + \mathbb{E}_{z \sim p_z(z)} log(1 - D(G(z)))$$

Some explanation of the model output !

$G(z)$	Generated sample (output of generator) from latent code **z** In a standard task, usually "*noise*" sampled from uniform or Gaussian
$D(x)$	Decision of discriminator with given input x, 1 means real 0 means fake

Objective function for GAN

For Generator's perspective,

$$L = \boxed{min_G} max_D \mathbb{E}_{x \sim p_{real}} logD(x) + \boxed{\mathbb{E}_{z \sim p_z(z)} log(1 - D(G(z))}$$

Independent of G

⟶ Maximize $D(G(z))$

⟶ Want to make $D(fake)$ to 1

⟶ Want to *fool* discriminator

Objective function for GAN

For Discriminator's perspective,

$$L = min_G \boxed{max_D} \boxed{\mathbb{E}_{x \sim p_{real}} log D(x)} + \boxed{\mathbb{E}_{z \sim p_z(z)} log(1 - D(G(z))}$$

$\longrightarrow$ Maximize $D(x)$

$\longrightarrow$ Distinguish real as real

D(.) ~ 1

$\longrightarrow$ Minimize $D(G(z))$

$\longrightarrow$ Distinguish fake as fake

D(.) ~ 0

" Want to well discriminate real and fake (sampled) one "

Objective function for GAN

Now we understood the concept of adversarial learning, but…

" how do we know the process works in a way that
$p_{model}(x)$ learns $p_{data}(x)$
? "

Global optimality in GAN (1/3)

First, Find optimal discriminator D*

$$L = min_G max_D \underline{\mathbb{E}_{x\sim p_{real}} log D(x) + \mathbb{E}_{z\sim p_z(z)} log(1 - D(G(z)))}$$

$$= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x}))dx + \int_{z} p_{\boldsymbol{z}}(\boldsymbol{z}) \log(1 - D(g(\boldsymbol{z})))dz$$

$$= \int_{\boldsymbol{x}} \underline{p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x})) + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x}))dx}$$

form of $a \log x + b \log(1-x)$

acheive maximum in [0,1] : $\frac{a}{(a+b)}$

by differential

optimal discriminator

for fixed G, $\boxed{D^*_G(x) = \frac{p_{data}}{p_{data}+p_g}}$

Global optimality in GAN (2/3)

with optimal discriminator, $D^*_G(x) = \frac{p_{data}}{p_{data}+p_g}$

Now the objective converted to

$$
\begin{aligned}
=&\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D^*_G(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D^*_G(\boldsymbol{x}))] \\
=&\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x}\sim p_g}\left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right]
\end{aligned}
$$

By the way, $\boxed{JSD(p||q)} = \frac{1}{2}\mathbb{E}_{x\sim p}\ln\left(\frac{p}{\frac{(p+q)}{2}}\right) + \frac{1}{2}\mathbb{E}_{x\sim q}\ln\left(\frac{q}{\frac{(p+q)}{2}}\right)$

Using some tricks, $\boxed{\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{\frac{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}{2}}\right] + \mathbb{E}_{\boldsymbol{x}\sim p_g}\left[\log \frac{p_g(\boldsymbol{x})}{\frac{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}{2}}\right]} - 2\log 2$

$$\boxed{= 2JSD(p_{data}(x)||p_g(x)) - 2\log 2}$$

Global optimality in GAN (3/3)

$$\boxed{2JSD(p_{data}(x)||p_g(x)) - 2\log 2}$$

1. Uniqueness of global optima only when

$$p_{data} = p_g$$

This point is called as "Nash equilibrium"

2. This proves how GAN model with the object function works as generative model (implicitly learns p_{data})
(Convergence of the algorithm)

Clear understanding of GAN : Pytorch implementation

```
class Generator(nn.Module):
    def __init__(self, args, img_shape):
        super(XAIGAN.Generator, self).__init__()
        self.img_shape = img_shape

        def block(in_feat, out_feat, normalize=True):
            layers = [nn.Linear(in_feat, out_feat)]
            if normalize:
                layers.append(nn.BatchNorm1d(out_feat, 0.8))
            layers.append(nn.LeakyReLU(0.2, inplace=True))
            return layers

        self.model = nn.Sequential(
            *block(args.latent_dim, 128, normalize=False),
            *block(128, 256),
            *block(256, 512),
            *block(512, 1024),
            nn.Linear(1024, int(np.prod(img_shape))),
            nn.Tanh()
        )

    def forward(self, z):
        img = self.model(z)
        img = img.view(img.size(0), *self.img_shape)
        return img
```

Because input pixels are nomalized into [-1,1]

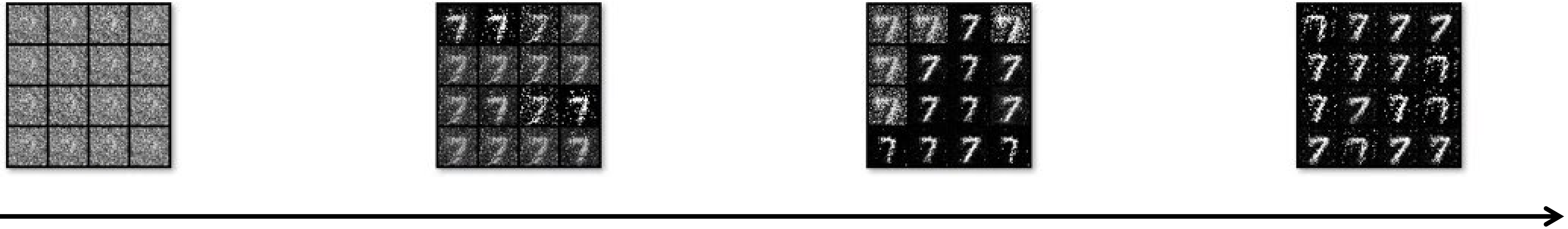
```
class Discriminator(nn.Module):
    def __init__(self, args, img_shape):
        super(XAIGAN.Discriminator, self).__init__()
        self.img_shape = img_shape

        self.model = nn.Sequential(
            nn.Linear(int(np.prod(img_shape)), 512),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Linear(512, 256),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Linear(256, 1),
            nn.Sigmoid(),
        )

    def forward(self, img):
        img_flat = img.view(img.size(0), -1)
        validity = self.model(img_flat)

        return validity
```

G, D are just any types of ANN that has capacity to extract feature of given data !

How sample looks over training

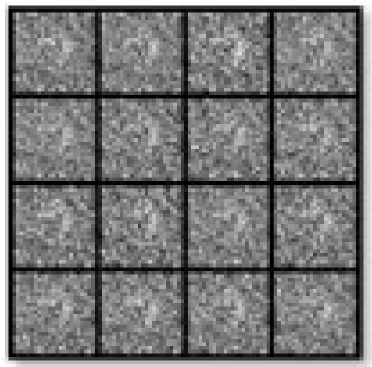
GAN training with MNIST label 7

Noisy / unstable samples
at the very first epochs



Training goes on

Faster convergence to Nash-equilibrium

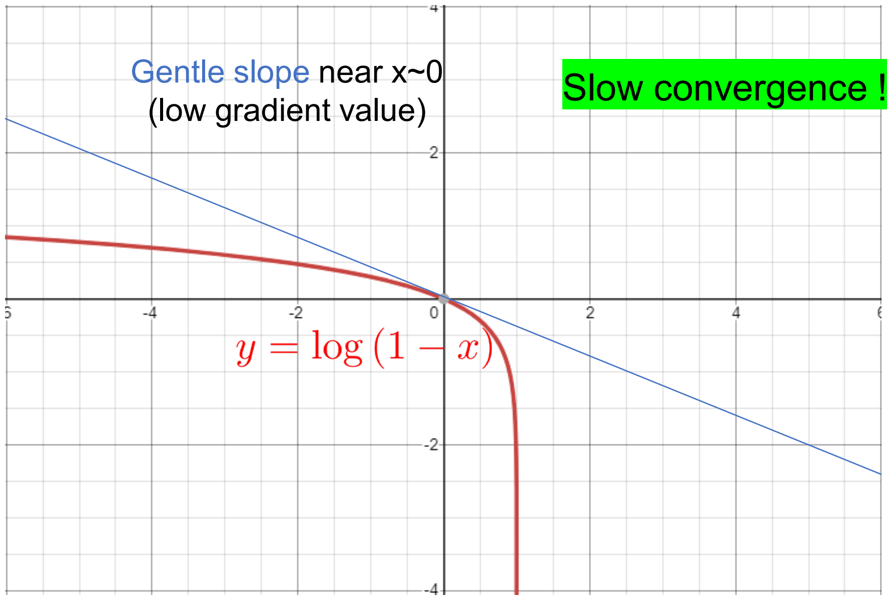


At the beginning of GAN training, discriminator easily classify fake example as 0

$$D(G(z)) \sim 0$$

Objective for generator,

$$min_G \mathbb{E}_{z \sim p_z(z)} log(1 - D(G(z)))$$



* Plotted using *desmos*

Faster convergence to Nash-equilibrium

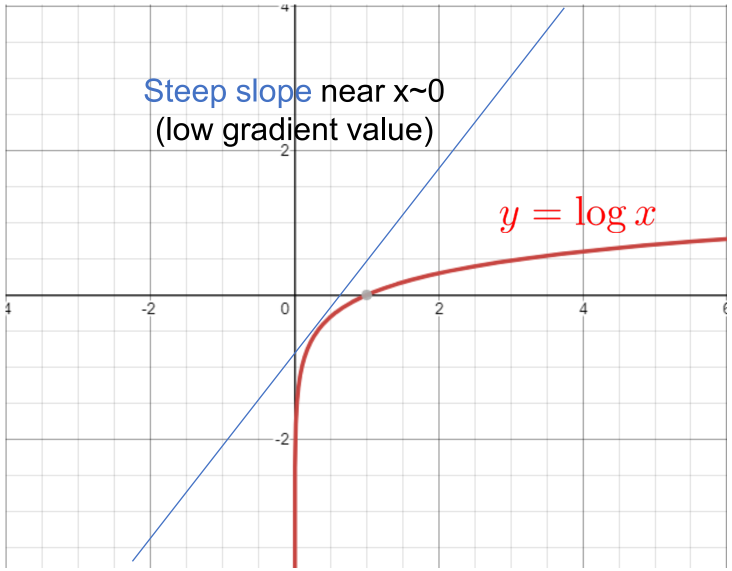
For the practical usage,

$$min_G \mathbb{E}_{z \sim p_z(z)} log(1 - D(G(z)))$$

$$\downarrow$$

$$max_G \mathbb{E}_{z \sim p_z(z)} log D(G(z))$$

Steep slope at the beginning of the training : Fast convergence !



* Plotted using *desmos*

Clear understanding of GAN : Pytorch implementation

- Training generator

We can implement GAN objective with BCE loss !

$$max_G \mathbb{E}_{z \sim p_z(z)} logD(G(z))$$

1. reformulate for gradient descent

$$\boxed{min_G \mathbb{E}_{z \sim p_z}(-logD(G(z)))}$$

Binary Cross Entropy Loss(x, y)
$$-y \log x - (1-y) \log(1-y)$$

$$\text{y} = 1$$
$$x = D(G(z))$$

2.

```
# Label for real (1) valid = Variable(Tensor(imgs.size(0), 1).fill_(1.0), requires_grad=False)
# Label for fake (0) fake = Variable(Tensor(imgs.size(0), 1).fill_(0.0), requires_grad=False)
```

3.

```
# Loss function
adversarial_loss = torch.nn.BCELoss()

for epoch in range(self.args.n_epochs):
    for i, (imgs, _) in enumerate(self.dataloader):

        # Sample noise as generator input
        z = Variable(Tensor(np.random.normal(0, 1, (imgs.shape[0], self.args.latent_dim))))

        # Generate a batch of images
        gen_imgs = generator(z)

        g_loss = adversarial_loss(discriminator(gen_imgs), valid)
        g_loss.backward()
        optimizer_G.step()
```

4. Fool discriminator

Clear understanding of GAN : Pytorch implementation

- Training discriminator

```
for epoch in range(self.args.n_epochs):
    for i, (imgs, _) in enumerate(self.dataloader):
        optimizer_D.zero_grad()

        # Measure discriminator's ability to classify real from generated samples
        real_loss = adversarial_loss(discriminator(real_imgs), valid)
        fake_loss = adversarial_loss(discriminator(gen_imgs.detach()), fake)
        d_loss = (real_loss + fake_loss) / 2

        d_loss.backward()
        optimizer_D.step()
```

Discriminator is trained in a way that well distinguish
real as real and *fake as fake*

Clear understanding of GAN : Pytorch implementation

- Compete each other until reaching at the convergence !

```
for _ in range(epochs):
        optimizer_G.step()
        optimizer_D.step()
```

Part3. Evaluation of generative model

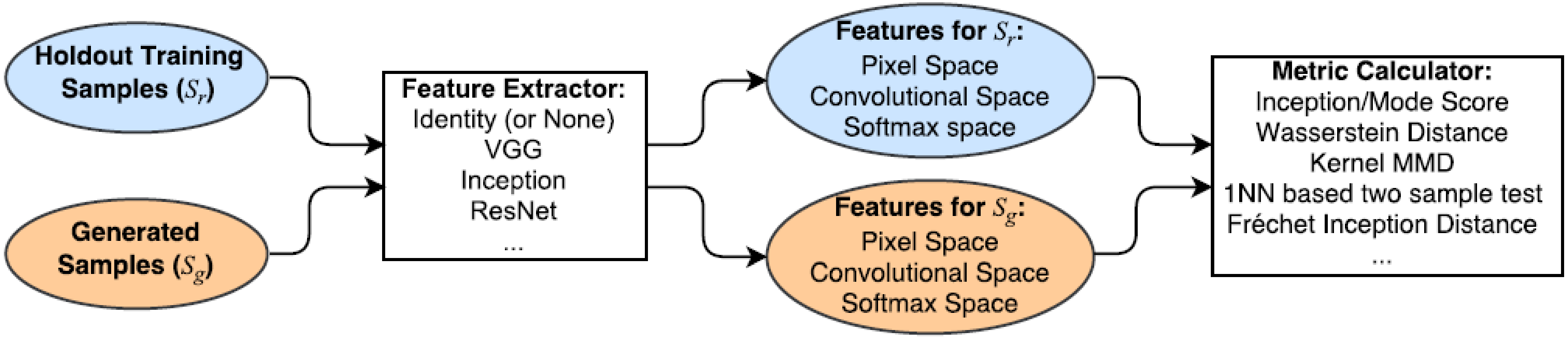
Qualitative evaluation

Examples of Photorealistic GAN-Generated Faces.Taken from Progressive Growing of GANs for Improved Quality, Stability, and Variation, 2017.

https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/

Subjective,
But easy way.

Quantitaive evaluation



Huang, Gao, et al. "An empirical study on evaluation metrics of generative adversarial networks." (2018).

Quantitaive evaluation : FID metrics

The Fréchet Inception Distance (FID)

$$\mathrm{FID}(\mathbb{P}_r, \mathbb{P}_g) = \|\mu_r - \mu_g\| + \mathbf{Tr}(\mathbf{C}_r + \mathbf{C}_g - 2(\mathbf{C}_r\mathbf{C}_g)^{1/2}),$$

μ: mean $\qquad$ $\mathbf{C}$: covariance

Step 1. extract hidden representation of real and samples using Inception Network*
(pretrained model using ImageNet dataset)

Step 2. compute mean and covariance of the each representation

Step 3. compute FID

lower FID means $\mathbb{P}_r \sim \mathbb{P}_g$

*Inception Network: Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. arXiv preprint arXiv:1409.4842, 2014

Heusel, Martin, et al. "Gans trained by a two time-scale update rule converge to a local nash equilibrium." *Advances in neural information processing systems* 30 (2017).

Quantitaive evaluation : Comparison overall with additional metrics

lower value means $\mathbb{P}_r \sim \mathbb{P}_g$

Kernel MMD
(Maximum Mean Discrepancy)

$$\mathrm{MMD}(\mathbb{P}_r, \mathbb{P}_g) = \left(\mathbb{E}_{\substack{\mathbf{x}_r, \mathbf{x}'_r \sim \mathbb{P}_r, \\ \mathbf{x}_g, \mathbf{x}'_g \sim \mathbb{P}_g}} \left[k(\mathbf{x}_r, \mathbf{x}'_r) - 2k(\mathbf{x}_r, \mathbf{x}_g) + k(\mathbf{x}_g, \mathbf{x}'_g) \right] \right)^{\frac{1}{2}}$$

k(.): kernel function

- Low sampling & computation complexity
- Recommended to use practically

WD
(Wassertein Distance)

$$\mathrm{WD}(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Gamma(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(\mathbf{x}^r, \mathbf{x}^g) \sim \gamma} \left[d(\mathbf{x}^r, \mathbf{x}^g) \right]$$

joint distribution

- Informally, minimum energy cost of moving and transforming a pile of dirt in one probability distribution to the other.
- $O(n^3)$ computation complexity, not recommended practically

---

FID
(Fretchet Inception Distance)

$$\mathrm{FID}(\mathbb{P}_r, \mathbb{P}_g) = \| \mu_r - \mu_g \| + \mathrm{Tr}(\mathbf{C}_r + \mathbf{C}_g - 2(\mathbf{C}_r \mathbf{C}_g)^{1/2}),$$

μ: mean $\quad$ $\mathbf{C}$: covariance

- Inception model is used for extract embedding of samples / almost standard metric for GAN
- Known to perform well in terms of discriminability, robustness and efficiency

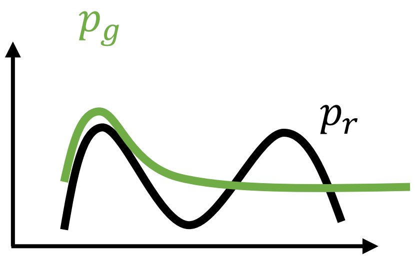
Huang, Gao, et al. "An empirical study on evaluation metrics of generative adversarial networks." (2018).

Wrap things up !

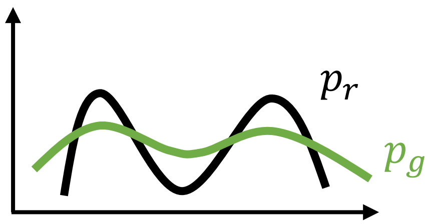
Consideration of limitations

Finding *Nash equilibrium* is difficult due to instability and possible collapse.

* *mode collapsing* problem in GAN



The model learns like this.



Instead of like this.

The model just try to minimize loss "*value*" given D, distributions are clustered in one mode.

There are **several follow-up research** for the problem :
unrolled-GAN [1], VEEGAN [2], …

[1] Metz, Luke, Poole, Ben, Pfau, David, and Sohl-Dickstein, Jascha. Unrolled generative adversarial networks. Corr, abs/1611.02163, 2016

[2] Srivastava, Akash, et al. "Veegan: Reducing mode collapse in gans using implicit variational learning." *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017.

Summary and conclusion

- GAN made a great breakthrough in AI-based generative model area and paved the way for many follow-up studies.

- However,

1. p_g is intractable, which lacks interpretability and makes it difficult for modeling & evaluating.
2. standard GAN has instability issue regard to mode collapsing

(Next talk)

- Introduction to many variants of GAN (follow-up studies.)

Thanks you for the listening.