CoE202
Fundamentals of Artificial intelligence
<Big Data Analysis and Machine Learning>

Generative Adversarial Network

Prof. Young-Gyu Yoon
School of EE, KAIST

Contents

- Recap
  - Regularization methods
  - Optimization methods
  - NN architectures
- Understanding classifier
- Joint probability distribution
- Generative model
- Generative Adversarial Network (GAN)

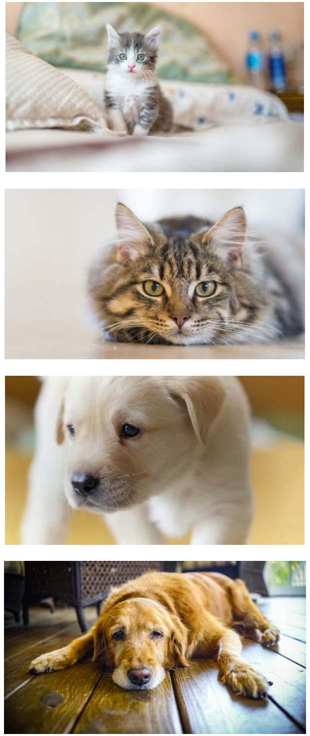
Generative Adversarial Network?

T. Karras et al, A Style-Based Generator Architecture for Generative Adversarial Networks (2018)

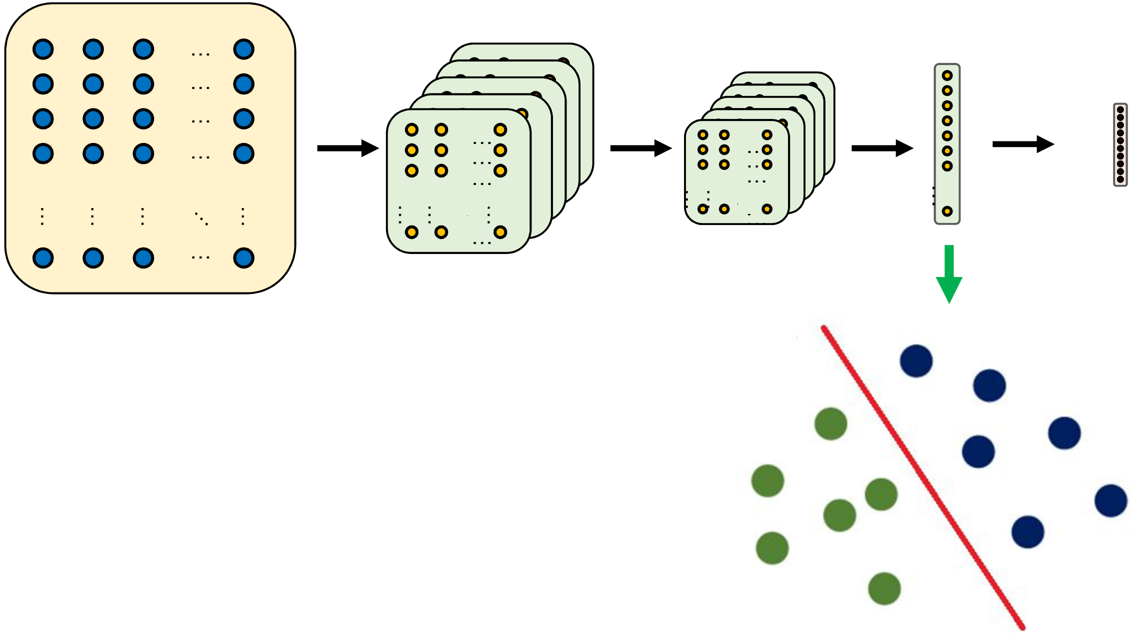
Generative Adversarial Network?

J.-Y Zhu et al, Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks (2017)

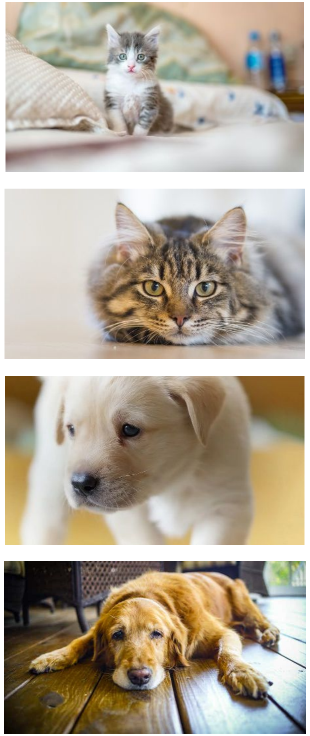
Understanding classifier



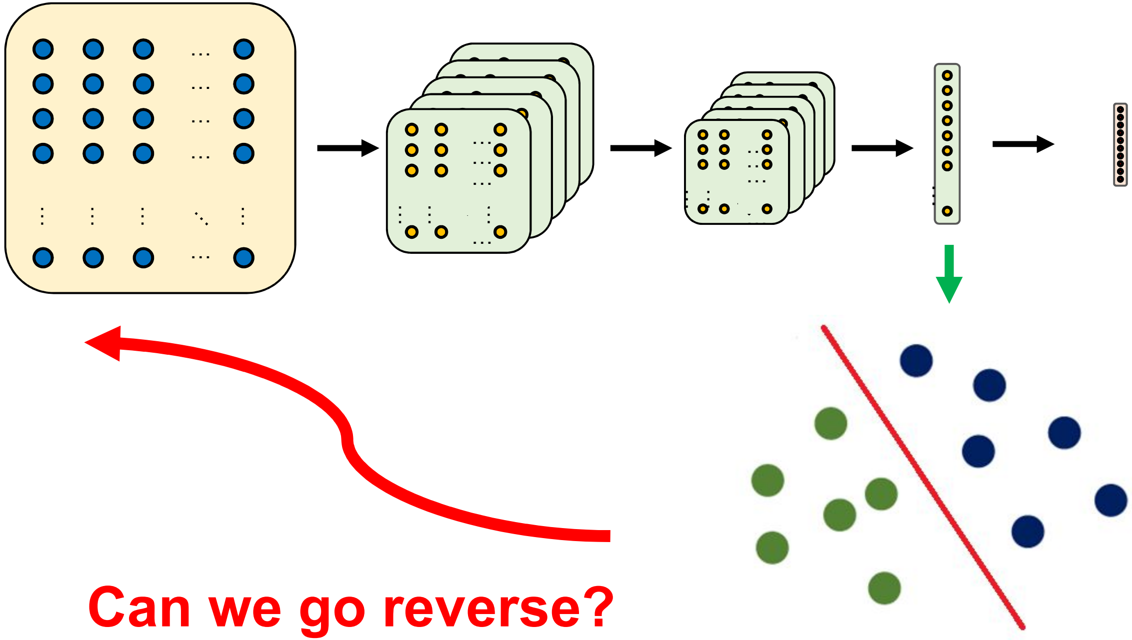
Images ($\mathrm{e.g.}, \mathbb{R}^{640\mathrm{x}480\mathrm{x}3}$) are mapped to a vector space ($\mathrm{e.g.}, \mathbb{R}^{128\mathrm{x}1}$) where cats and dogs are linearly separable.
Then, the last layer does linear classification.



Understanding classifier

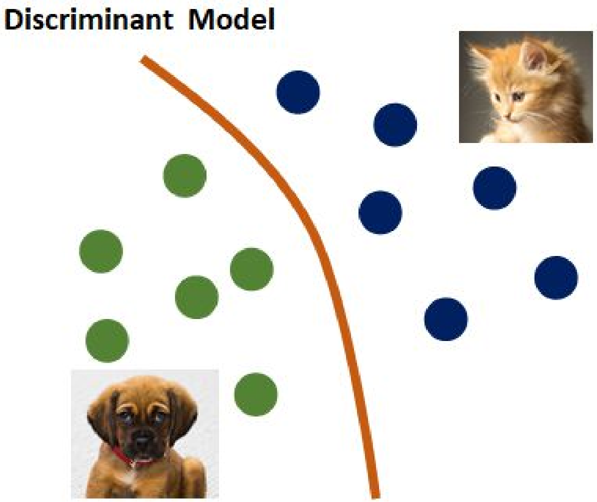


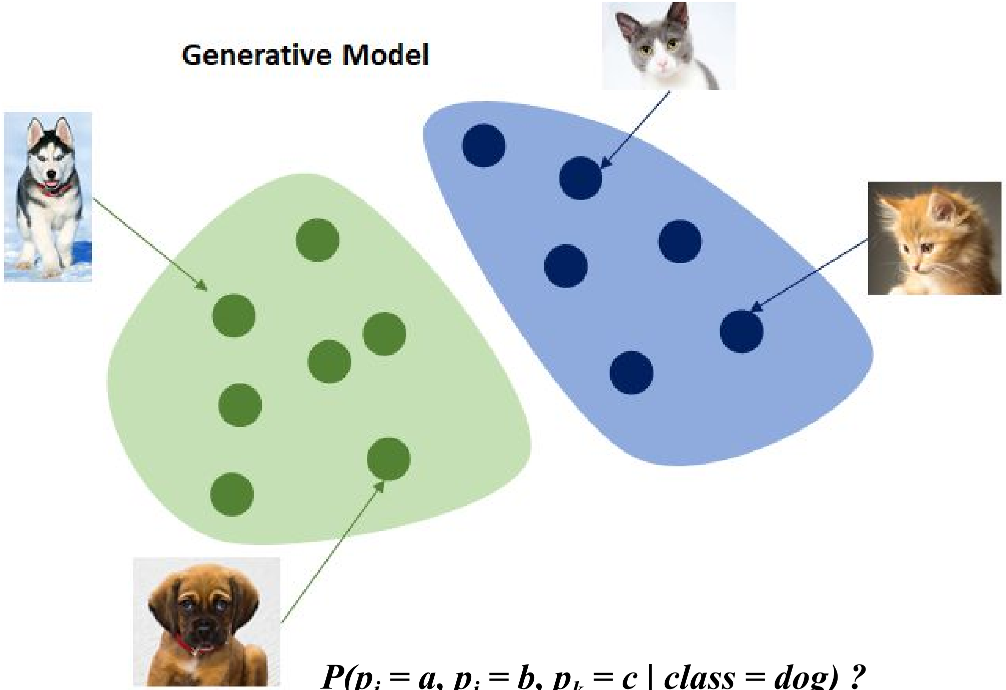
Images (e.g., $\mathbb{R}^{640x480x3}$) are mapped to a vector space (e.g., $\mathbb{R}^{20x1}$)
where cats and dogs are linearly separable.
Then, the last layer does linear classification.



Can we go reverse?

Discriminative model vs. Generative model

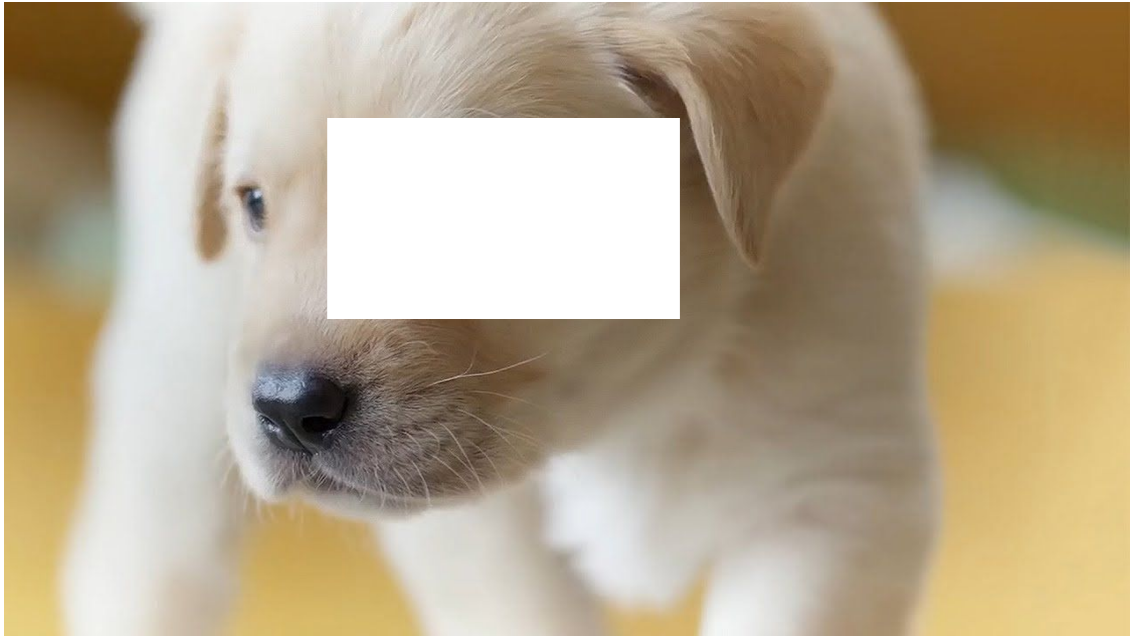




$P(class = dog \mid p_i = a, p_j = b, p_k = c)$ $P(p_i = a, p_j = b, p_k = c \mid class = dog)$?

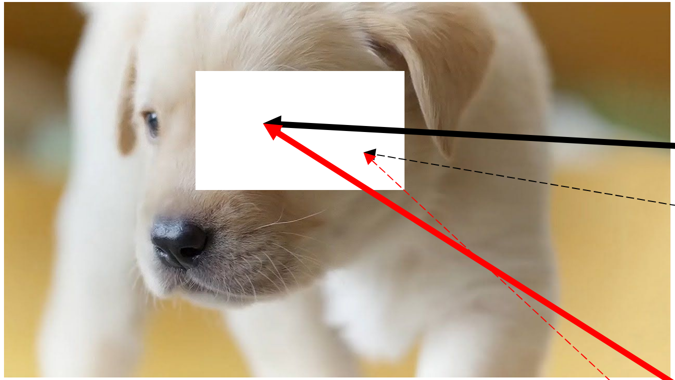
Just drawing a decision boundary (discriminative model)
vs. knowing possible distribution of data (generative model)

Human-way of thinking “distribution”



- Let’s occlude a part of the image of a puppy
- Without even seeing the occluded part, we kind of know how that part is supposed to look like
  - We know that there should be an eye, the fur color would be similar to other parts, etc

Let’s translate that thought



- We need to translate this intuition into a mathematical form
- First translation attempt (still human-like)

We expect an eye here

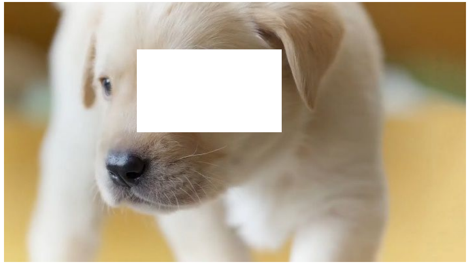
We expect light brown color fur in this area

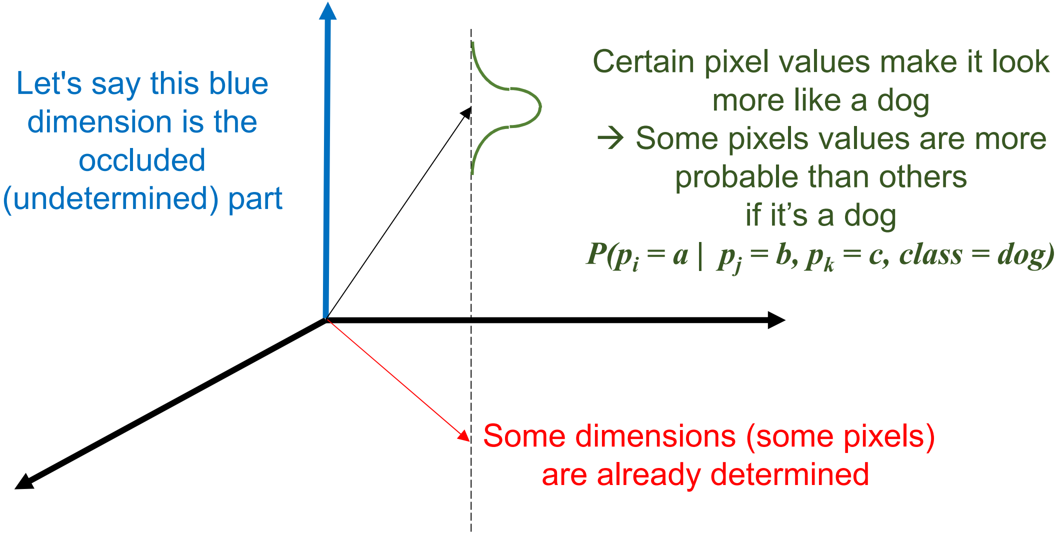
- Second translation attempt

The pixels values here have relatively high probability to be black

The pixels values here have high probability to be light brown color

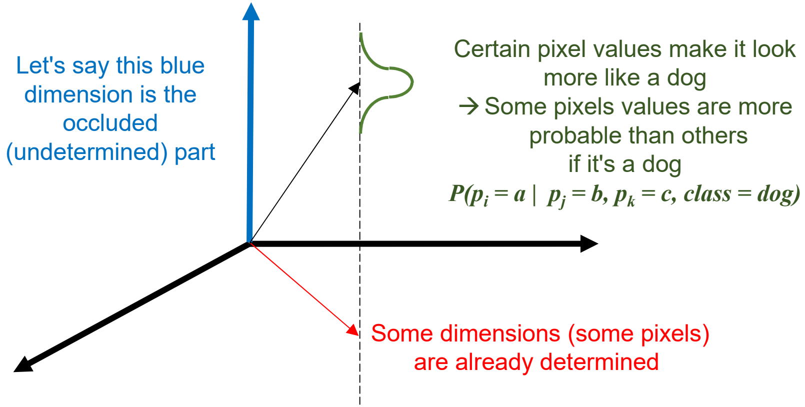
Let's translate that thought

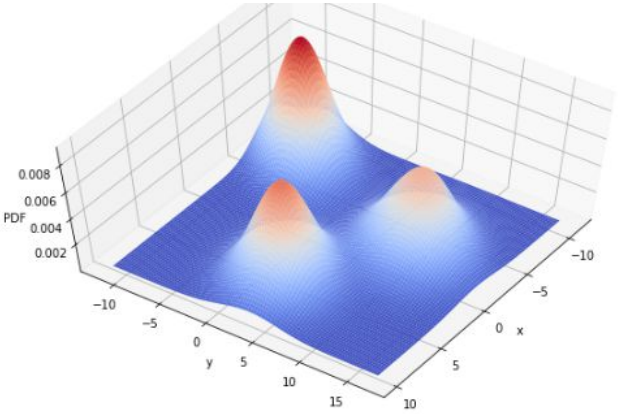




- Assuming that this is a dog image,
  - (Given the non-occluded part of the image) we know that those occluded pixels are more likely to have certain values than other values

Let’s extend

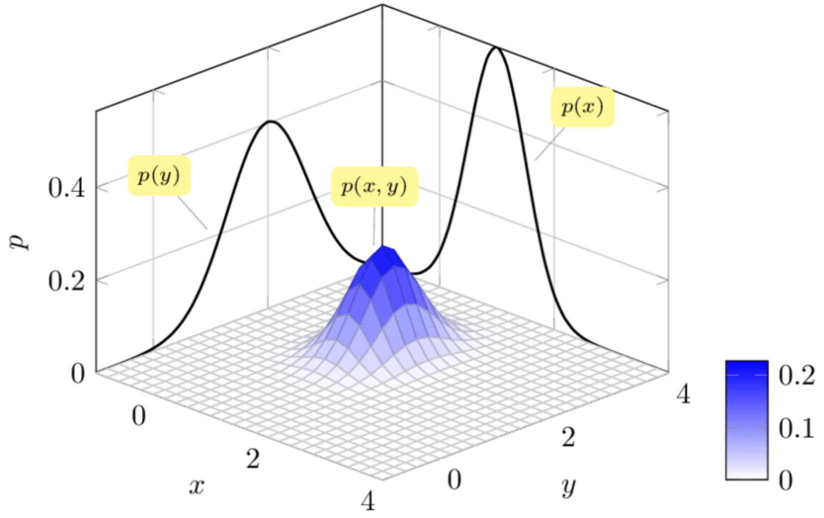




$P(p_i = a, p_j = b, p_k = c \mid class = dog)$

- The earlier logic was based on the assumption that certain pixel values were fixed
  → Let’s drop this assumption
- What we know from the earlier logic is that the “probability” of pixel values are entangled with one another
- And the way they are entangled partly determined by the class “dog“
  → **joint** probability distribution
- “Images of a dog” has their own joint PDF and the images of different classes have their own joint PDFs

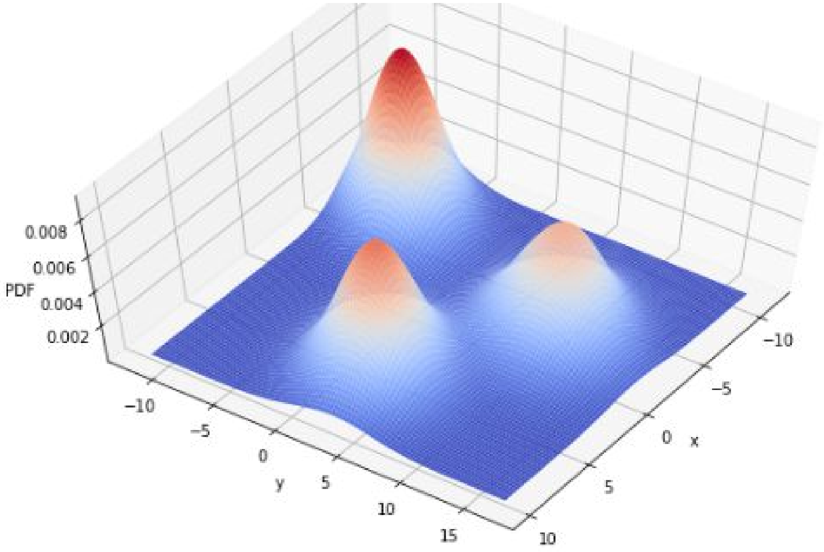
Joint probability distribution



$$p(x, y) = p(x)p(y)$$

x and y are independent (not entangled at all)

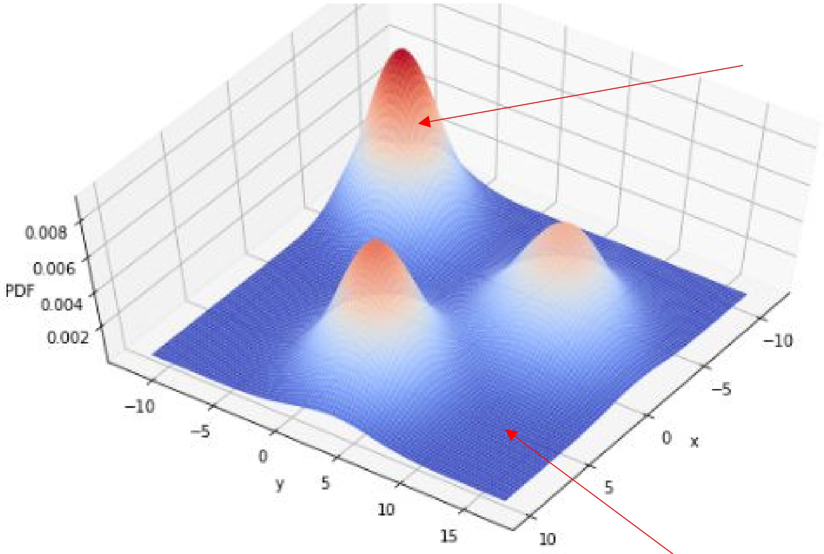
Joint probability distribution



$$p(x, y) \neq p(x)p(y)$$

x and y are not independent (entangled)

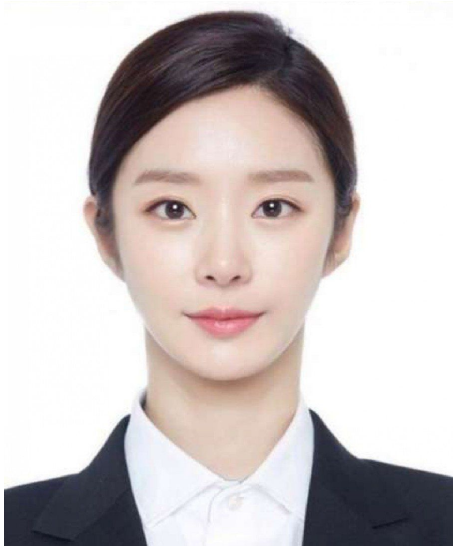
Joint probability distribution



It is probable that this data points is drawn from this PDF

It is not probable that this data points is drawn from this PDF

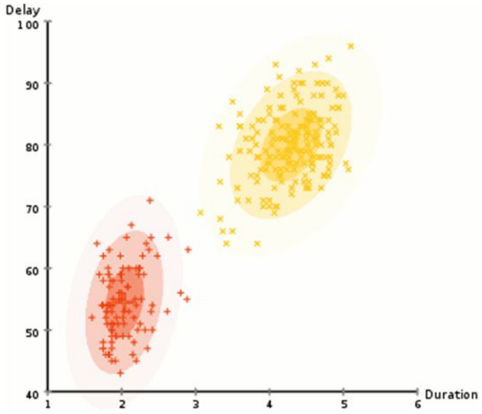
Simple image to think about: ID picture



- What can we say about the joint PDFs of passport photo?

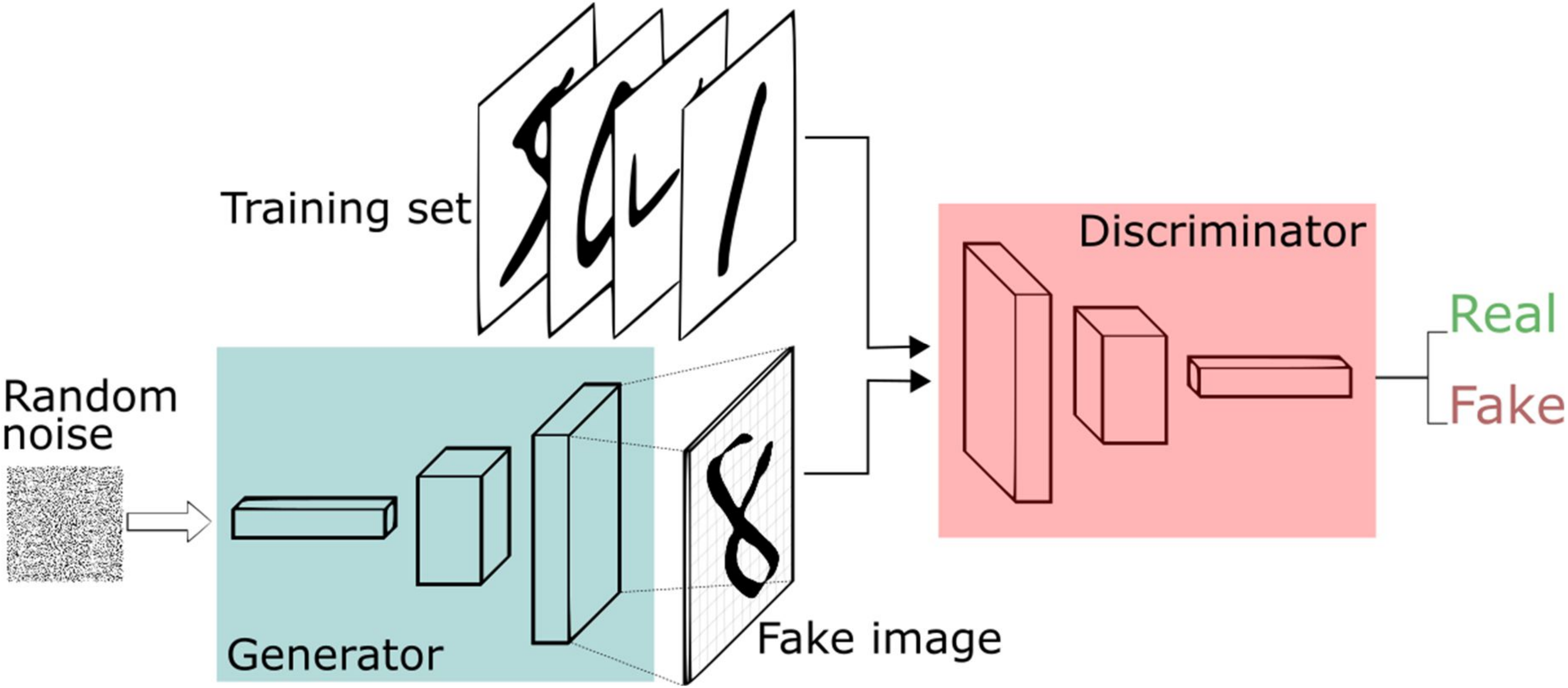
For a generative model

- Once we know (or have a model of) the joint probability distribution function of data, we can generate the data
- For that, we want our generative model to "learn" the joint PDF from the data
    - Data typically has very high dimension and we want our model to learn the "key features"



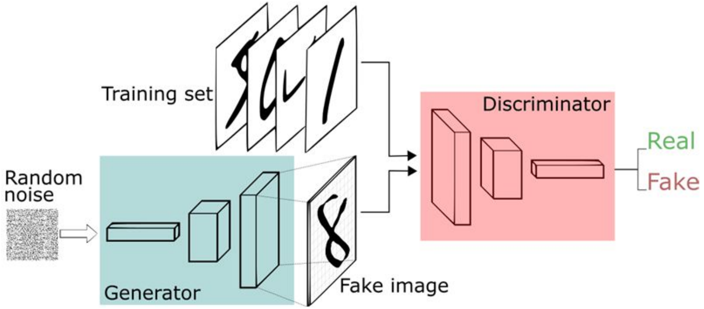
- Multiple data points are now represented by a PDF that has only several parameters (through learning)
- Once we have the PDF we can generate as many data points as we want

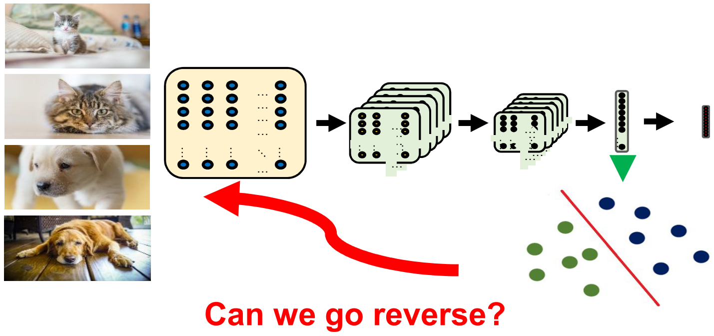
Generative Adversarial Network



I. J. Goodfellow et al, Generative Adversarial Nets (2014)

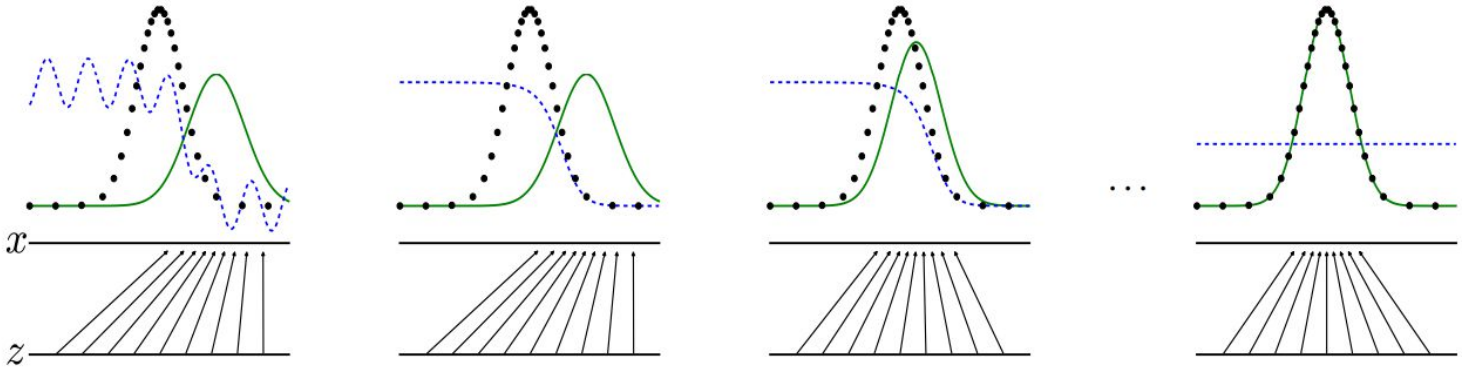
Generative Adversarial Network





- There are two networks
  - **Discriminator**: a network that takes real and fake inputs and attempts to discriminate them (real/fake classification)
  - **Generator**: a network that takes a random vector (low dimensional input) and generate output (high dimensional output) as an attempts to fool the discriminator network
- We train them simultaneously
- Both networks will get better and better over time
  - Discriminator will become better at classifying
  - Generator will become better at generating real-like output

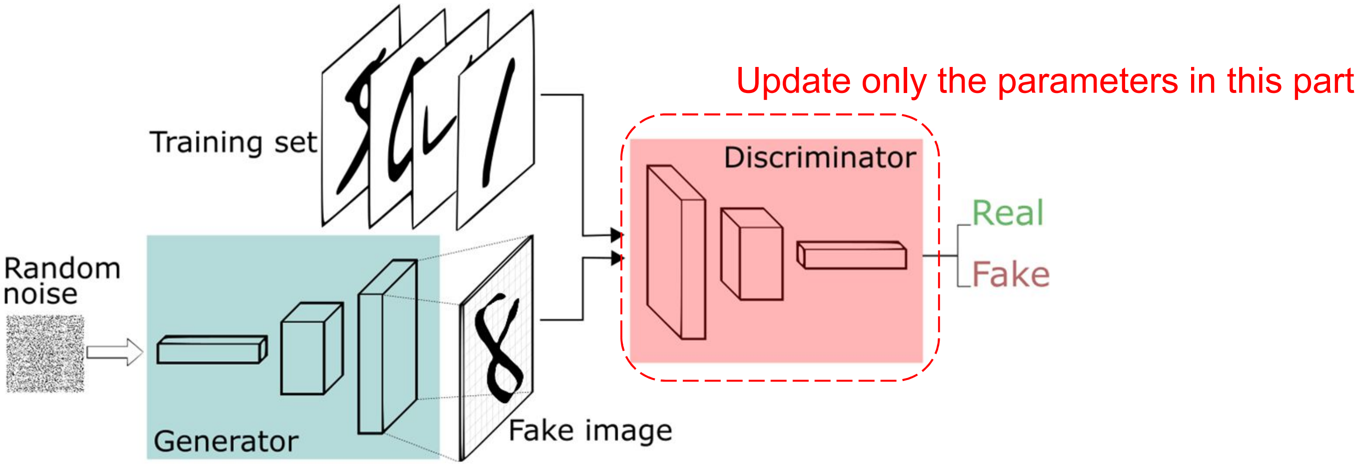
OK, how is this related to PDF…?



Blue: discriminator
Black: data PDF
Green: generator PDF

- Our generator does not explicitly learn the PDF
- Conceptually speaking, the discriminator attempts to see if the generated data "fits" in the PDF (of real data)
- Conversely, the generator must generate data that "fits" in the PDF of real data to fool the discriminator

How to train (discriminator)



$$\mathcal{L}_{BCE} = -(y\,log(f(x)) + (1-y)log(1-f(x)))$$

$y = 1$ for real

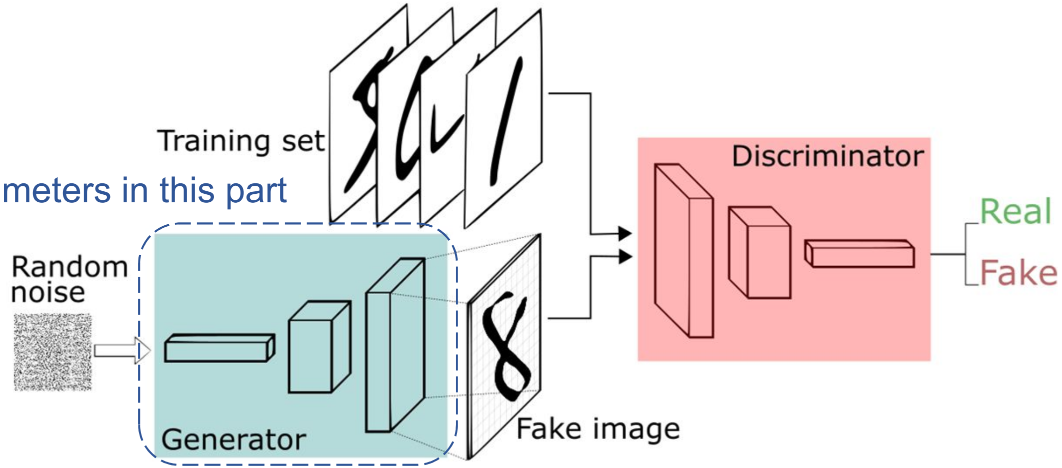
$y = 0$ for fake

$$\mathcal{L}_D = -(log(D(\underbrace{x}_{\text{real image}})) + log(1 - D(\underbrace{G(z)}_{\text{generated image}})))$$

Then, we do gradient descent update to train the discriminator

How to train (generator)

Update only the parameters in this part



Let's remove the negative sign here, because we want the classifier to be "wrong"

$$\mathcal{L}_{BCE} = \boxed{-}\boxed{(y\,log(f(x))} + (1-y)log(1-f(x)))$$

$y = 1$ for real

$y = 0$ for fake

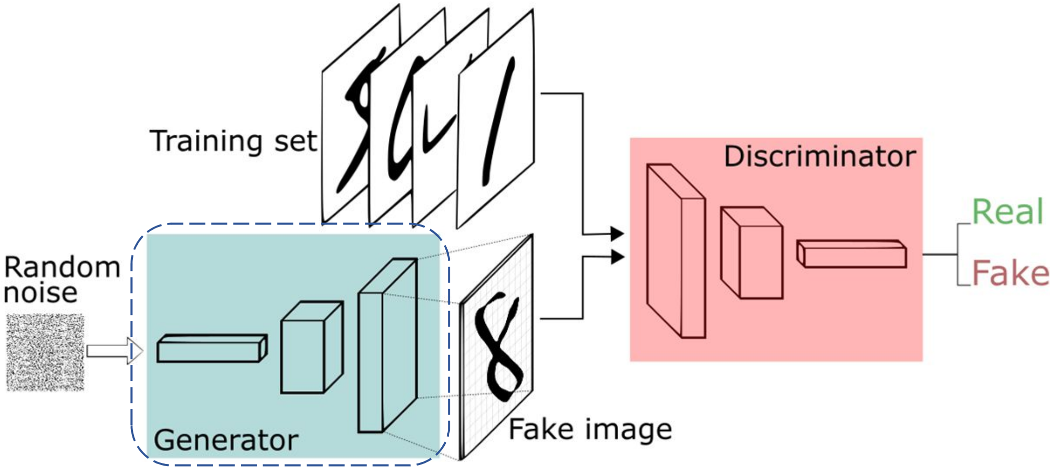
Let's remove this term, as we don't care about what happens to real image for training generator

$$\mathcal{L}'_G = log(1 - D(\boxed{G(z)}))$$

generated image

Then, we do gradient descent update to train the generator

How to train (generator)



$$\mathcal{L}'_G = log(1 - D(G(z)))$$

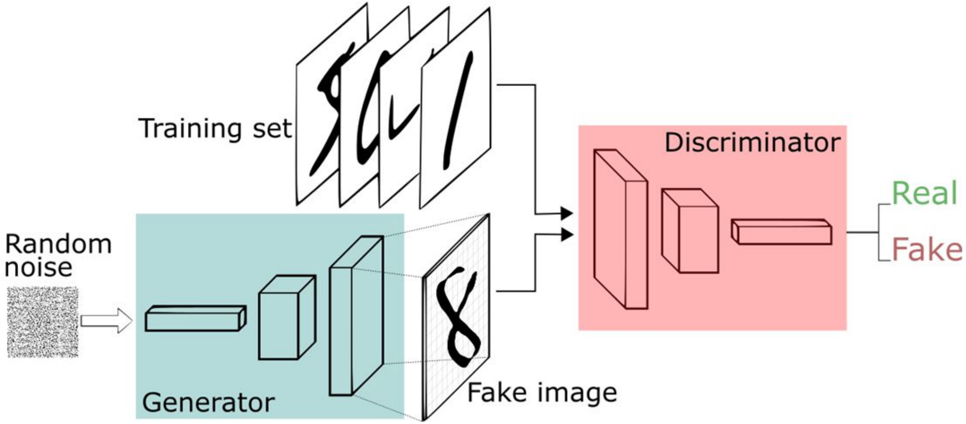
$$\Downarrow$$

$$\mathcal{L}_G = -log(D(G(z)))$$

analogy

$$\arg\min_x (x-2)^2 = \arg\min_x (x-2)^4$$

How to train GAN



$$\mathcal{L}_D = -(log(D(x)) + log(1 - D(G(z))))$$

$$\mathcal{L}_G = -log(D(G(z))$$

```
for I in range(num_iteration):

    for j in range (num_batch):

        sample minibatch of m random noise
        sample minibatch of m real data

        update discriminator with gradient descent applied to L_D
        update generator with gradient descent applied to L_G
```

How to train GAN

- Why

$$\mathcal{L}'_G = log(1 - D(G(z)))$$

⬇

$$\mathcal{L}_G = -log(D(G(z)))$$

1. Learning to discriminate is easier than learning to generate (especially when the generated output is bad)
2. Discriminator will converge first and become very confident
   → D(G(z)) will be near zero
   → Loss will be nearly zero regardless of G:
   log(1-D(G(z))) = log (1-0) = log(1) = 0
   → small gradient
3. On the other hand, log(D(G(z))) can change a lot
   → large gradient

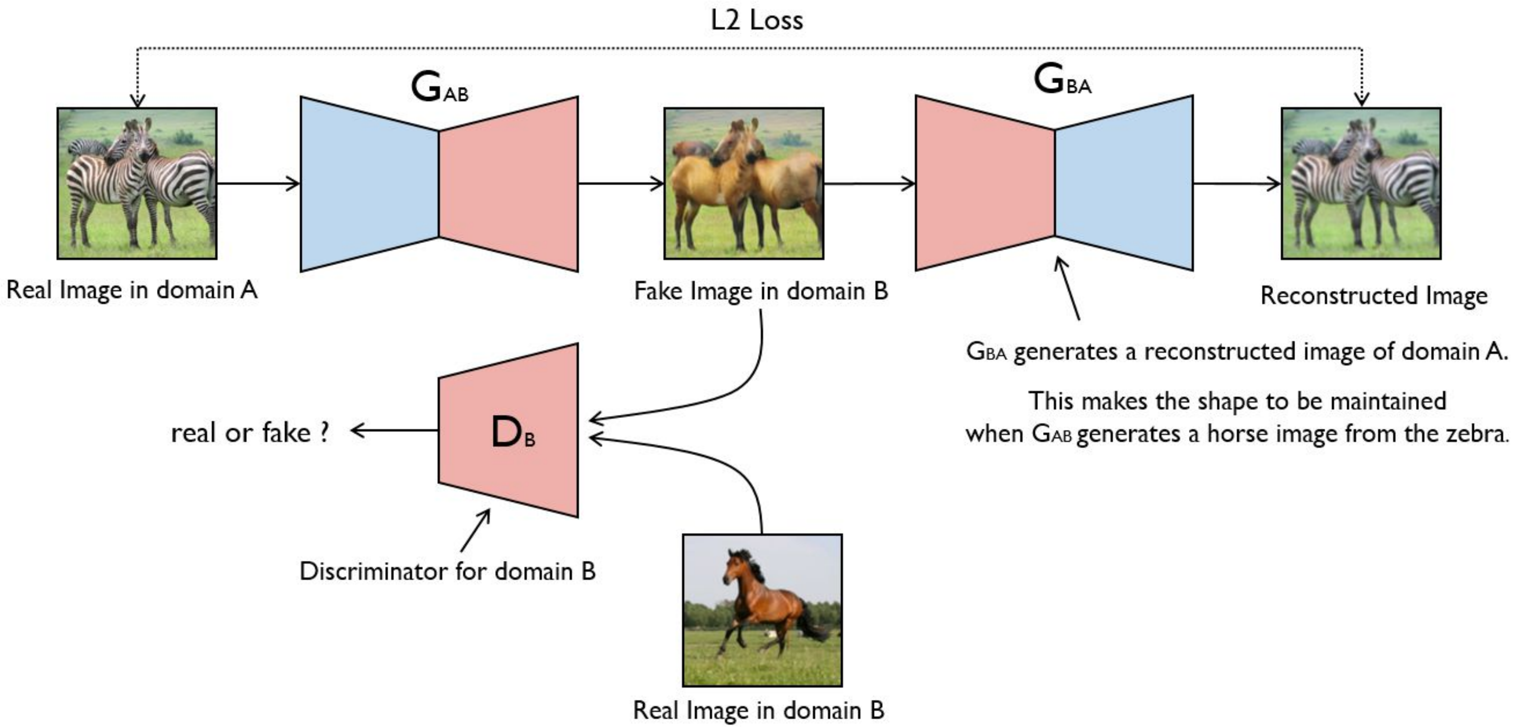
- From the original paper

In practice, equation 1 may not provide sufficient gradient for G to learn well. Early in learning, when G is poor, D can reject samples with high confidence because they are clearly different from the training data. In this case, $\log(1 - D(G(\boldsymbol{z})))$ saturates. Rather than training G to minimize $\log(1 - D(G(\boldsymbol{z})))$ we can train G to maximize $\log D(G(\boldsymbol{z}))$. This objective function results in the same fixed point of the dynamics of G and D but provides much stronger gradients early in learning.

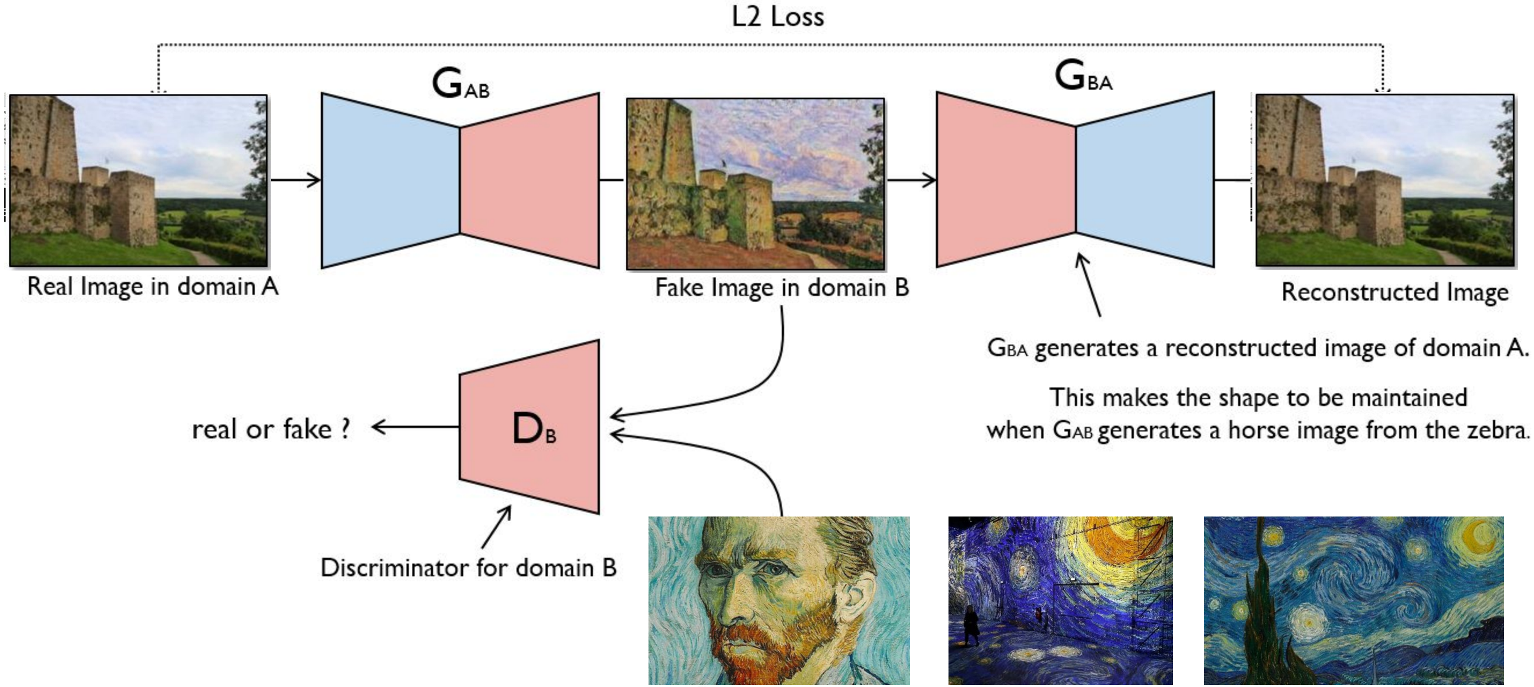
How to ...?

J.-Y Zhu et al, Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks (2017)

CycleGAN


L2 Loss
GAB
GBA
Real Image in domain A
Fake Image in domain B
Reconstructed Image
GBA generates a reconstructed image of domain A.
This makes the shape to be maintained
when GAB generates a horse image from the zebra.
real or fake ?
DB
Discriminator for domain B
Real Image in domain B

CycleGAN



Real Image in domain B

Summary

- Joint Probability Distribution Function
  - Different (e.g., cat vs dog) data have different PDF
- Discriminative vs Generative model
  - Knowing boundary vs. Knowing PDF
- Data generation requires (explicit or implicit) PDF
- Generative adversarial network is a powerful framework for (implicitly) learning the PDF of a dataset and generate data
- GAN is based on two competing networks
- GAN can be extended in many interesting ways

References

- Original GAN paper
  - https://papers.nips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf
- CycleGAN
  - https://junyanz.github.io/CycleGAN/