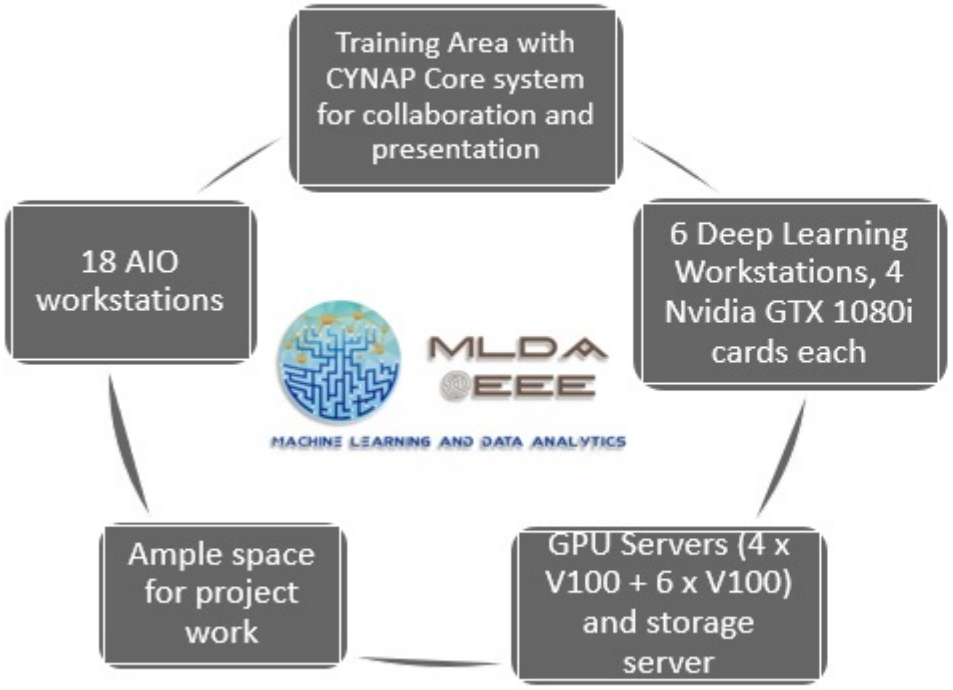
Generative Adversarial Networks (GANs) Workshop

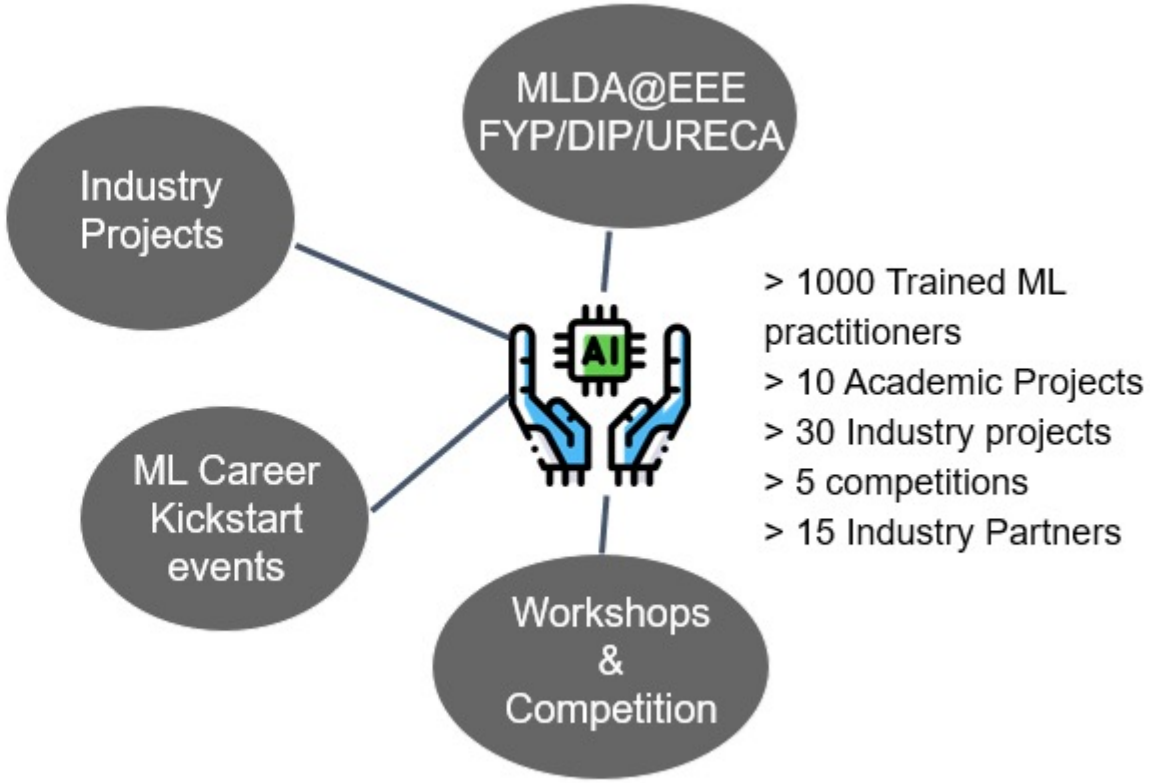
Presented by:
Dewi & Van

Slides by Chan Yi Xuan

MLDA's Mission

Provide an integrated platform for EEE/IEM students to learn and implement Machine Learning, Data Science & AI, as well as facilitate connections with the industry.

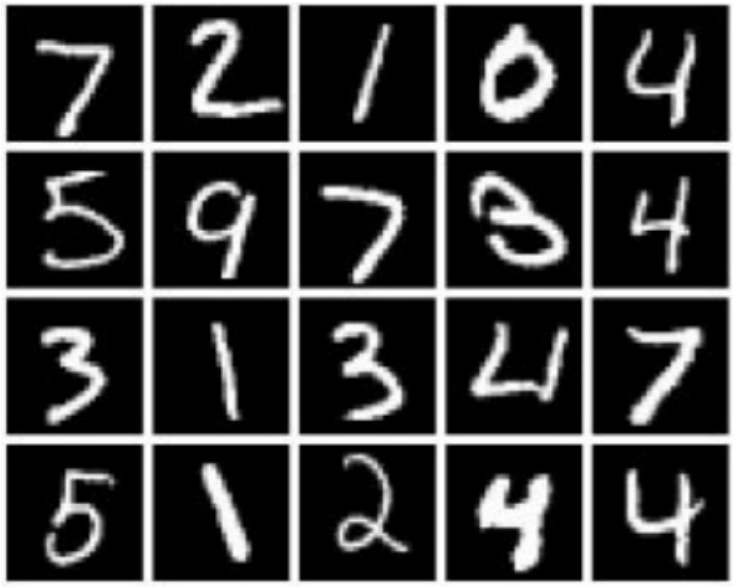




Which one is FAKE?

Generative Models

Given training data, generative models aim at learning the true data distribution of the training set to generate new data points from this distribution with some variations.



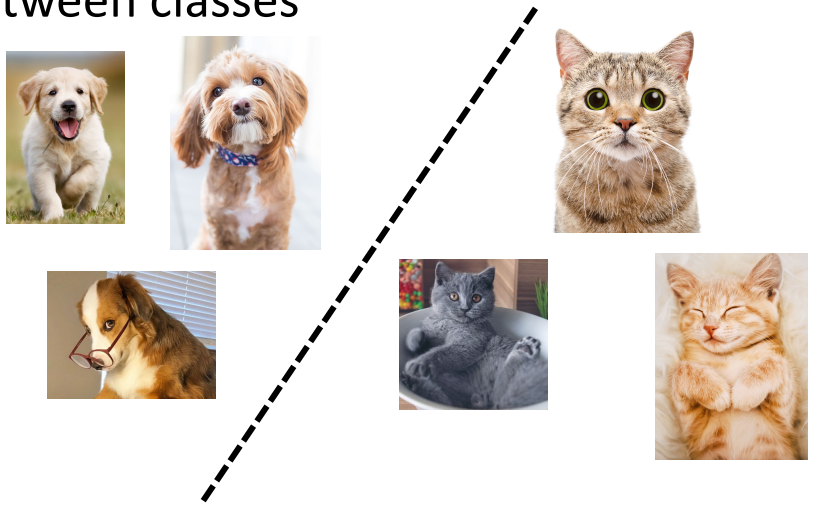
Training samples

Generated samples

Classifier Model vs. Generative Model

Classifier Model

- Goal: Models the **decision boundary** between classes



Features **Class**

$X \longrightarrow Y$

$P(Y \mid X)$

Generative Model

- Goal: Models the **actual distribution** of each classes

Noise **Class** **Features**

$\xi, Y \longrightarrow X$

$P(X \mid Y)$

Generative Models

Given training data, generative models aim at learning the true data distribution of the training set to generate new data points from this distribution with some variations.

Two famous deep generative model algorithms:

- **Variational Autoencoder (VAE)**
- **Generative Adversarial Network (GAN)**

GANs: Cool Applications

Face Generation StyleGAN2

More from https://thispersondoesnotexist.com/

GANs: Cool Applications

Cat Generation
StyleGAN2

GANs: Cool Applications

Super Resolution
SRGAN

GANs: Cool Applications

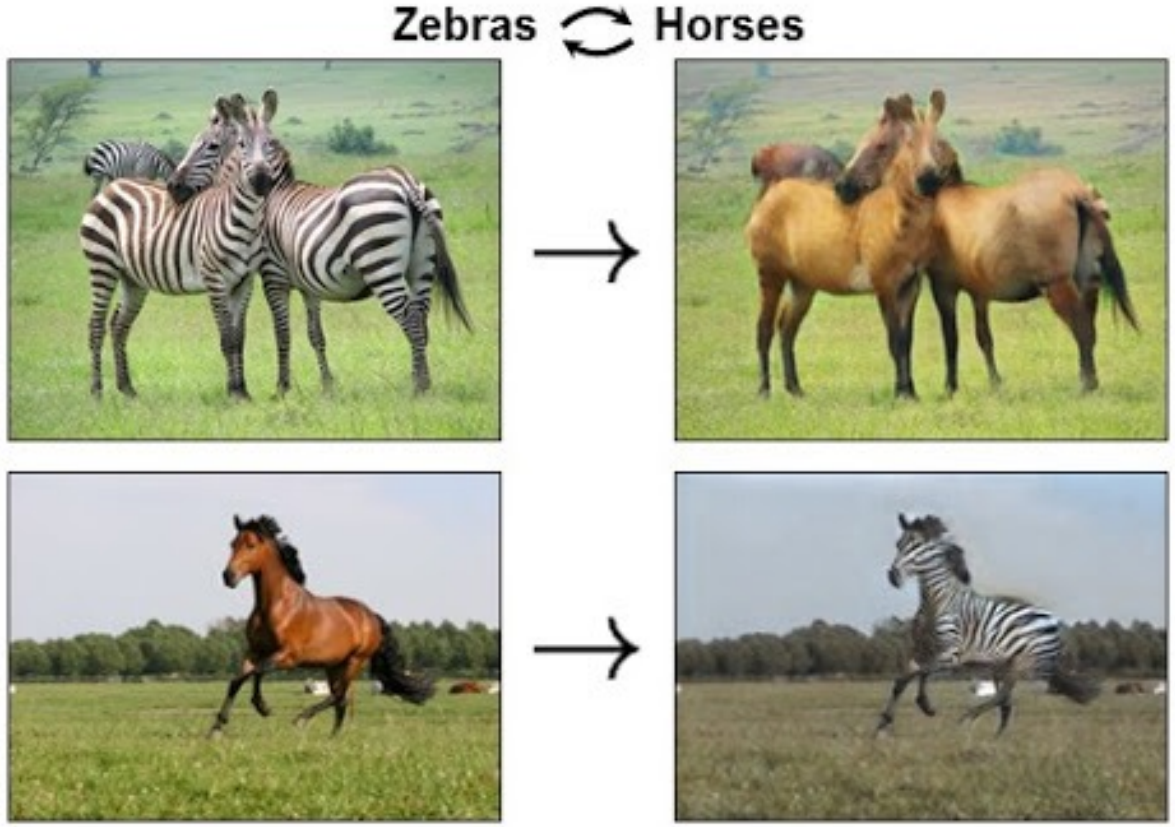


Image Translation
CycleGAN

Generative Adversarial Networks

Generator

Learns to make fakes that look real

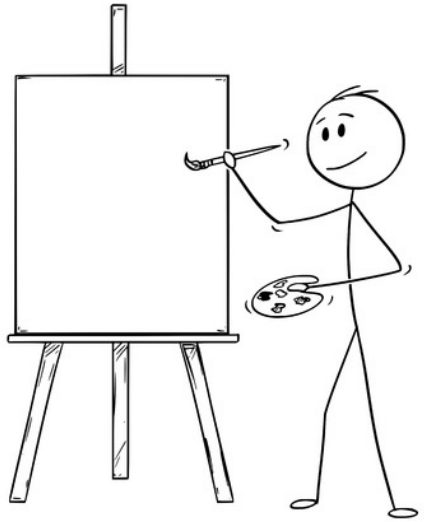
Discriminator

Learns to distinguish real from fake

Generative Adversarial Networks

Generator vs. **Discriminator**

Learns to make fakes that look real



Art Forger

vs.

Art Critic

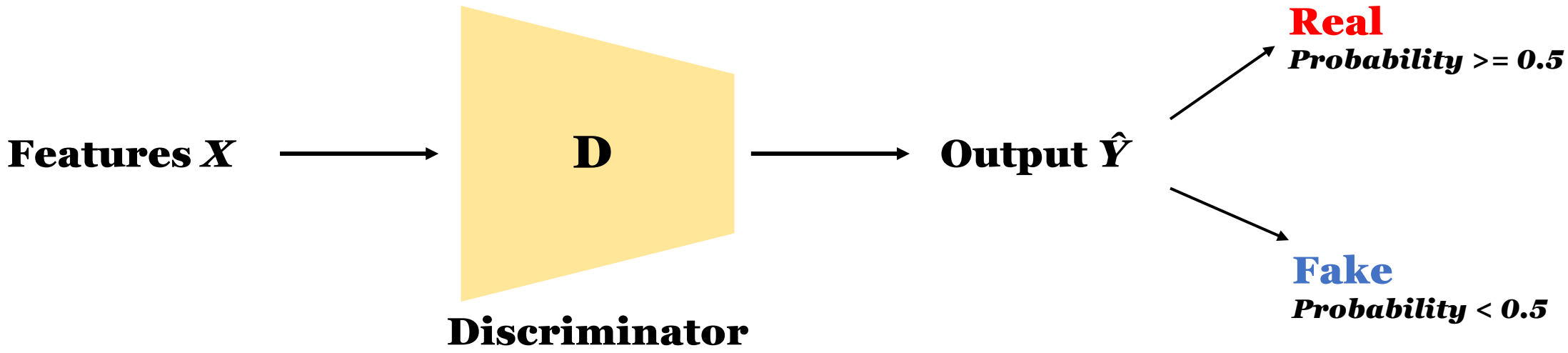
Learns to distinguish real from fake

Discriminator

Discriminator: A classifier which can distinguish between different classes

Input: Real or Fake Samples

Output: Probability whether the sample is real

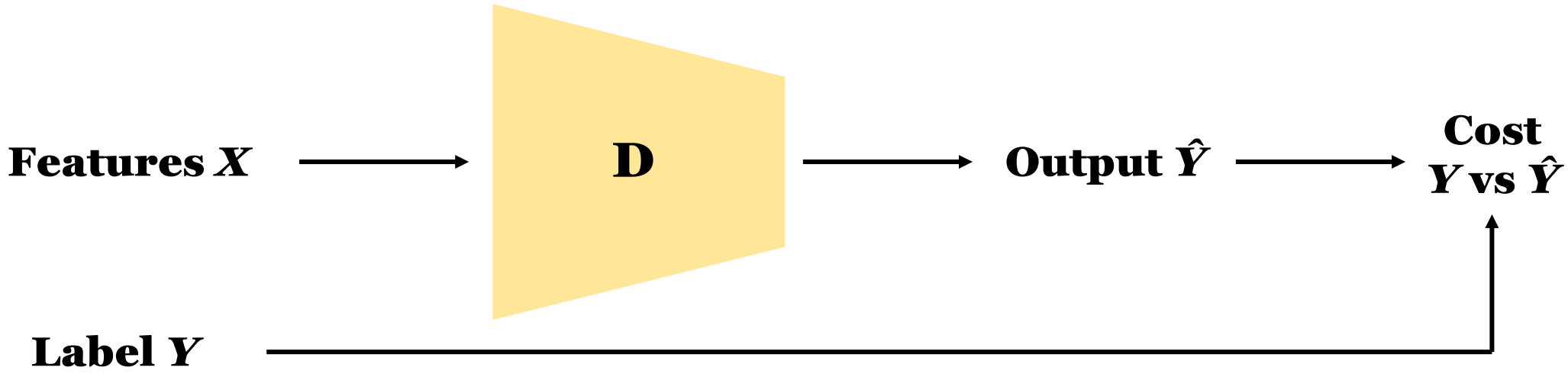


Discriminator

Discriminator: A classifier which can distinguish between different classes

Input: Real or Fake Samples

Output: Probability whether the sample is real

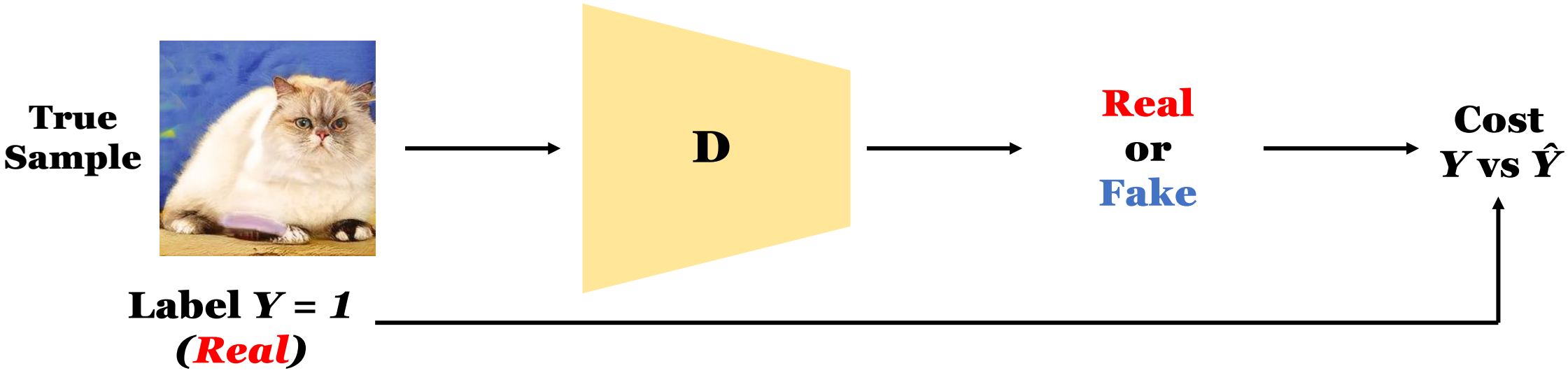


Discriminator

Discriminator: A classifier which can distinguish between different classes

Input: Real or Fake Samples

Output: Probability whether the sample is real

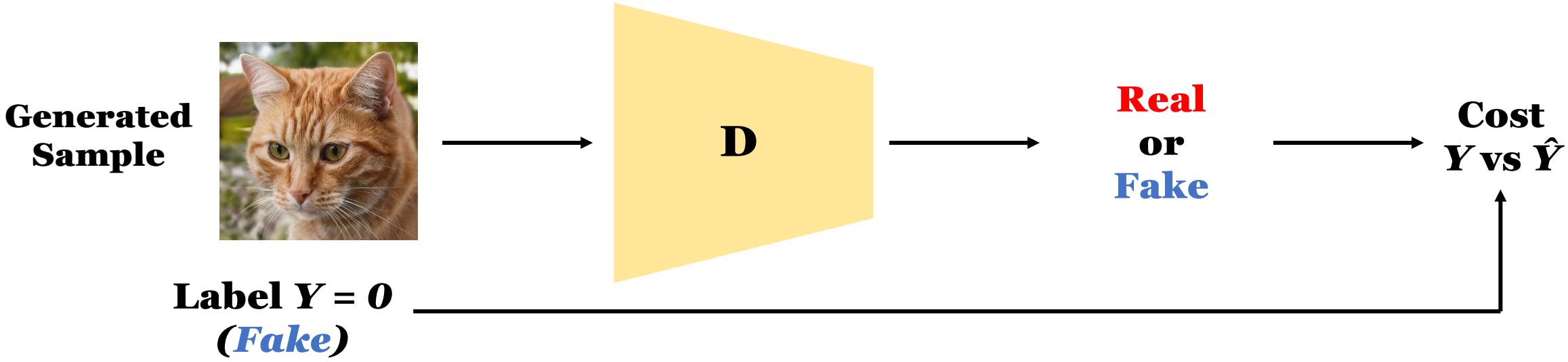


Discriminator

Discriminator: A classifier which can distinguish between different classes

Input: Real or Fake Samples

Output: Probability whether the sample is real

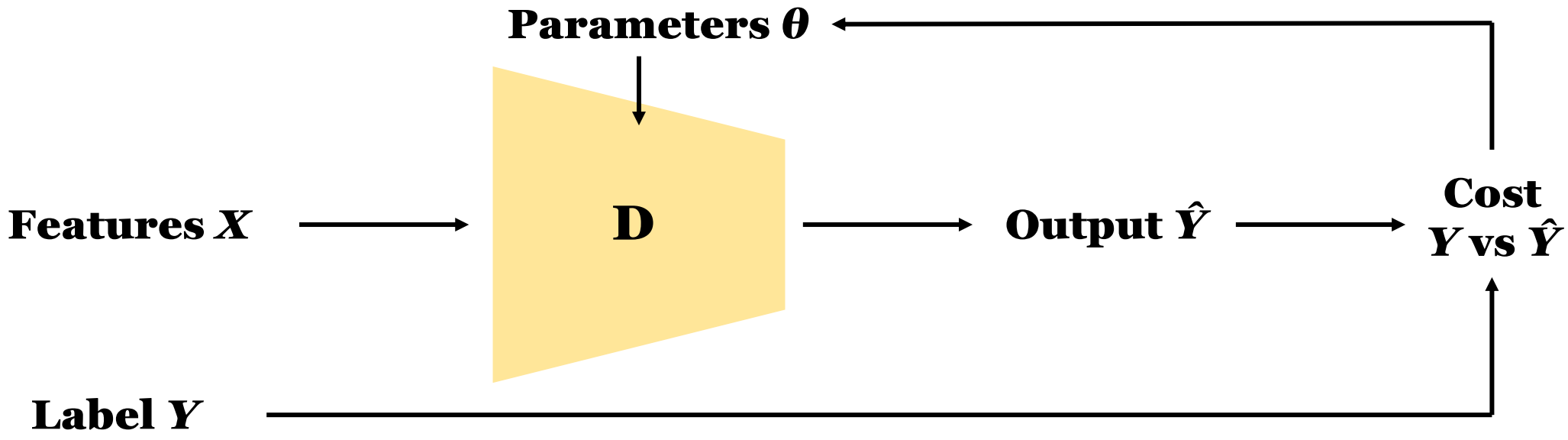


Discriminator

Discriminator: A classifier which can distinguish between different classes

Input: Real or Fake Samples

Output: Probability whether the sample is real



Discriminator

Discriminator: A classifier which can distinguish between different classes

Input: Real or Fake Samples

Output: Probability whether the sample is real

$$P(Y \mid X)$$

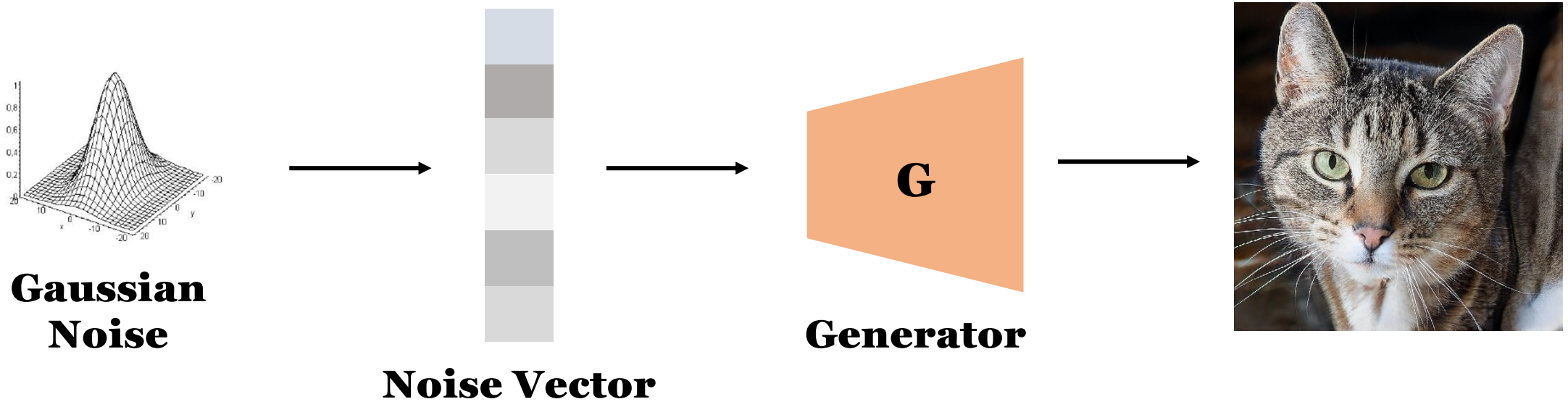
Class Features

Generator

Generator: Model that is used to generate new plausible examples from the problem domain

Input: Random noise

Output: Fake data

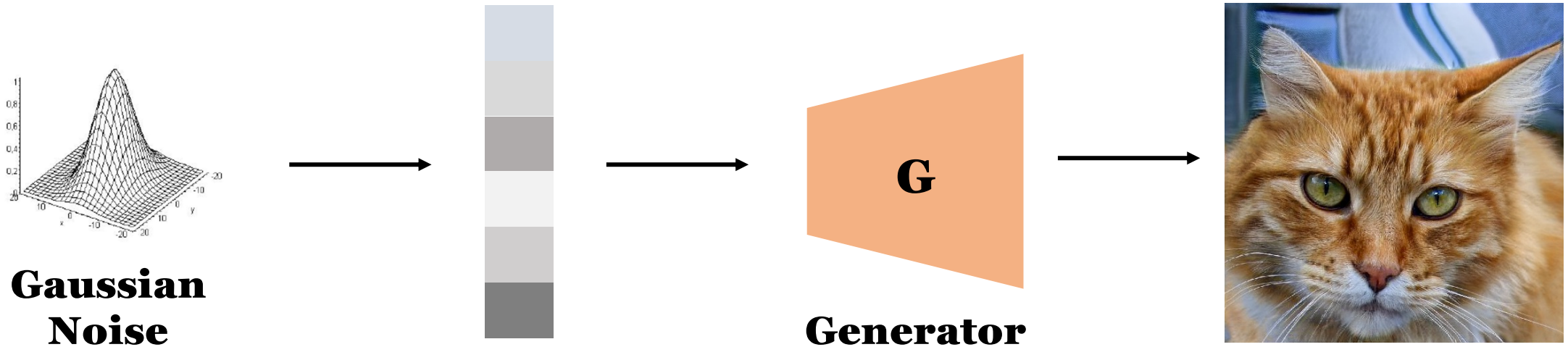


Generator

Generator: Model that is used to generate new plausible examples from the problem domain

Input: Random noise

Output: Fake data



Generator

Generator: Model that is used to generate new plausible examples from the problem domain

Input: Random noise

Output: Fake data

Noise Vector

Noise Vector

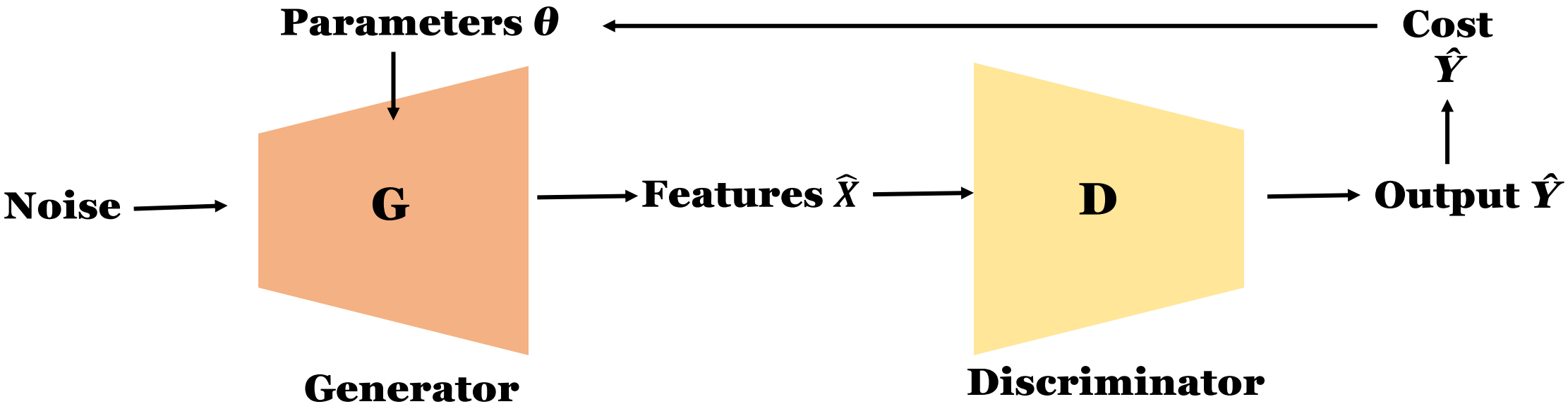
- **Also known as latent variable, they are those variables that are important for a domain but are not directly observable.**
- **Random noise vector's dimensionality is smaller than the target output's dimensionality**
- **Noise helps GAN to produce a wide variety of data**

Generator

Generator: Model that is used to generate new plausible examples from the problem domain

Input: Random noise

Output: Fake data

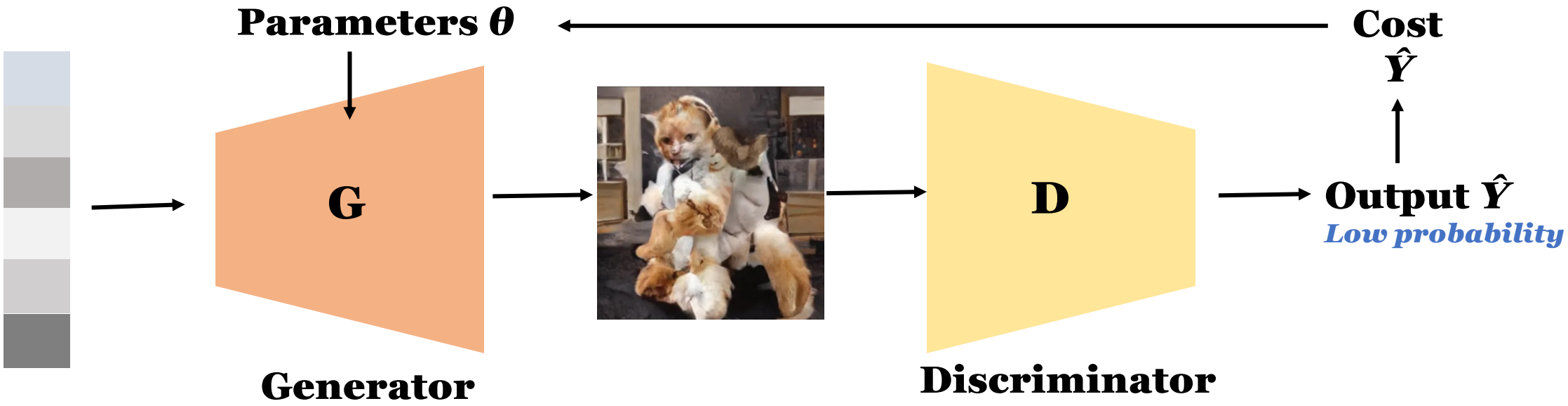


Generator

Generator: Model that is used to generate new plausible examples from the problem domain

Input: Random noise

Output: Fake data

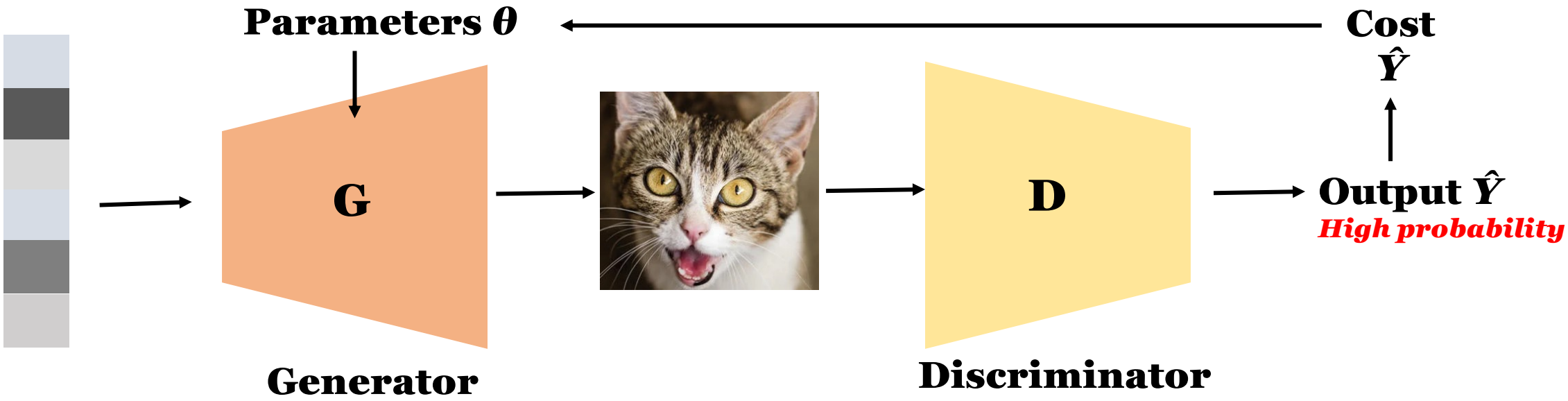


Generator

Generator: Model that is used to generate new plausible examples from the problem domain

Input: Random noise

Output: Fake data



Generator

Generator: Model that is used to generate new plausible examples from the problem domain

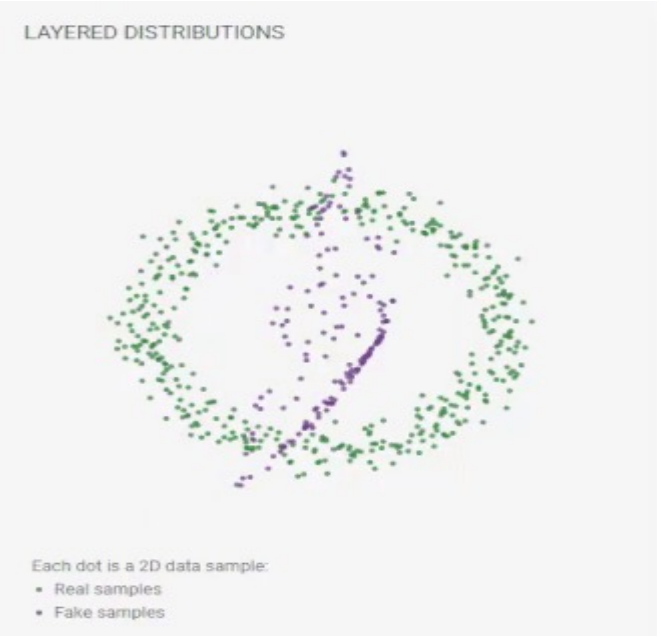
Input: Random noise

Output: Fake data

$$P(X \mid Y)$$

Features **Class**

Approximate the distribution of problem domain



https://poloclub.github.io/ganlab/

Cost Function

Loss Function: Binary Cross Entropy (BCE) Loss

m: sample size
y: label
ŷ: prediction

$$BCE = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right)$$

Cost Function

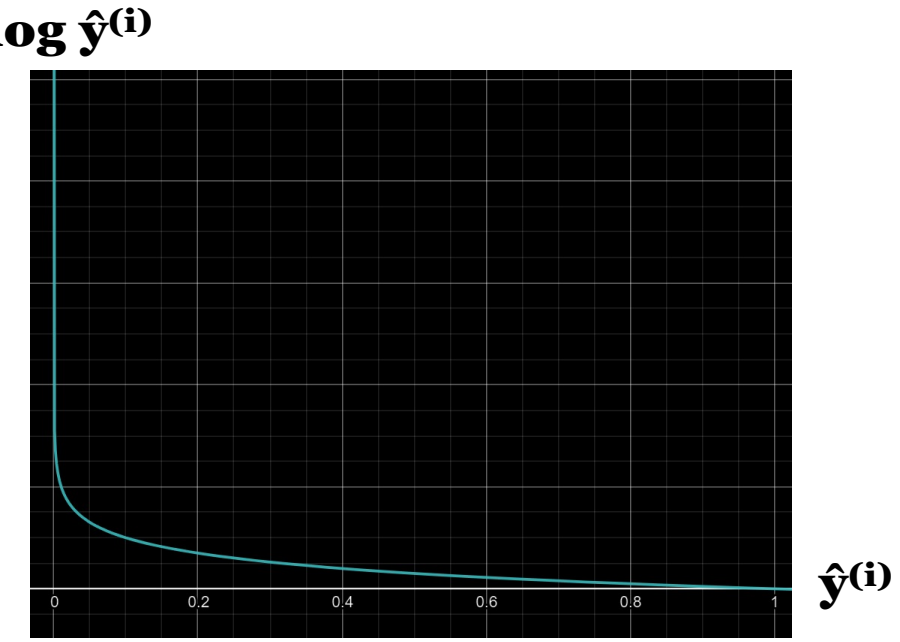
Loss Function: Binary Cross Entropy (BCE) Loss

m: sample size
y: label
ŷ: prediction

$$BCE = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right)$$

Label y = 1 (real)

Note: *log(1) = 0*



Cost Function

Loss Function: Binary Cross Entropy (BCE) Loss

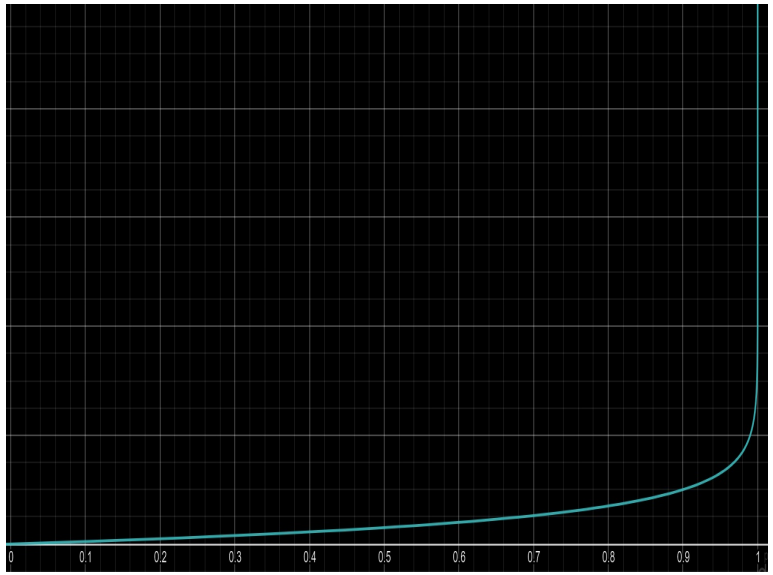
m: sample size
y: label
ŷ: prediction

$$BCE = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right)$$

Label y = 0 (fake)

Note: *log(1) = 0*

log (1 - ŷ(i))



ŷ(i)

Cost Function

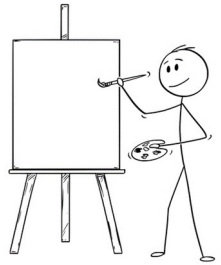
Loss Function: Binary Cross Entropy (BCE) Loss

$$BCE = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right)$$

BCE Loss -> Minimax Loss

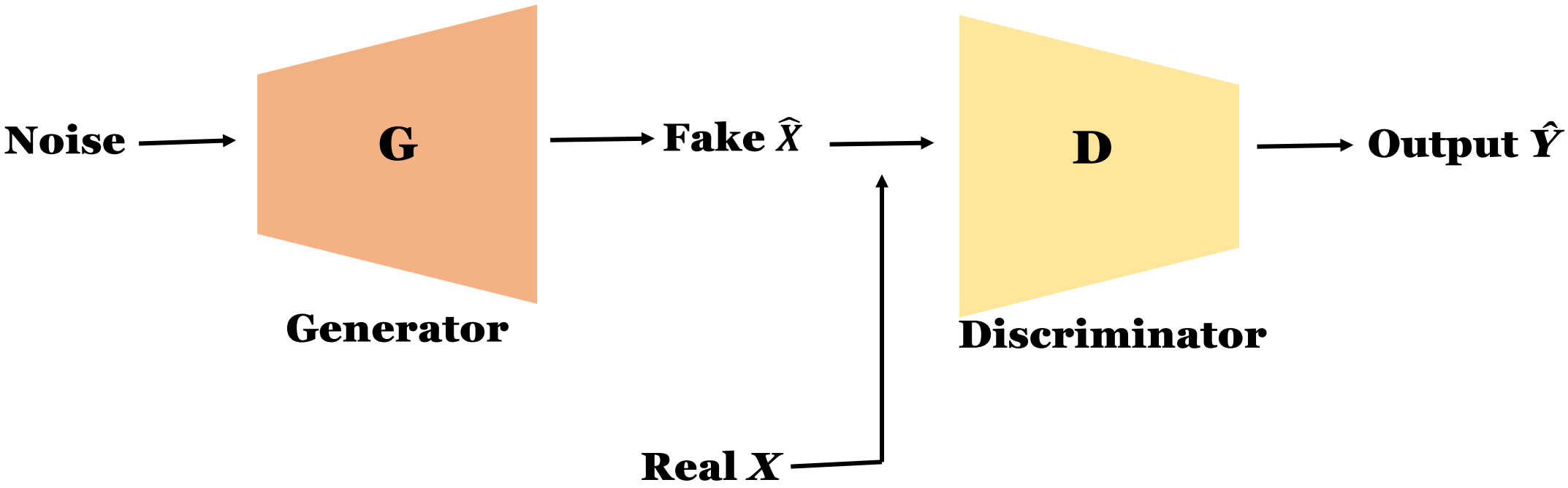
$$\min_{d} \max_{g} \; -\left[\mathbb{E}\left(\log\left(d(x)\right)\right) + \mathbb{E}\left(1 - \log\left(d\left(g(z)\right)\right)\right)\right]$$

Discriminator:
Minimize cost



Generator:
Maximize cost

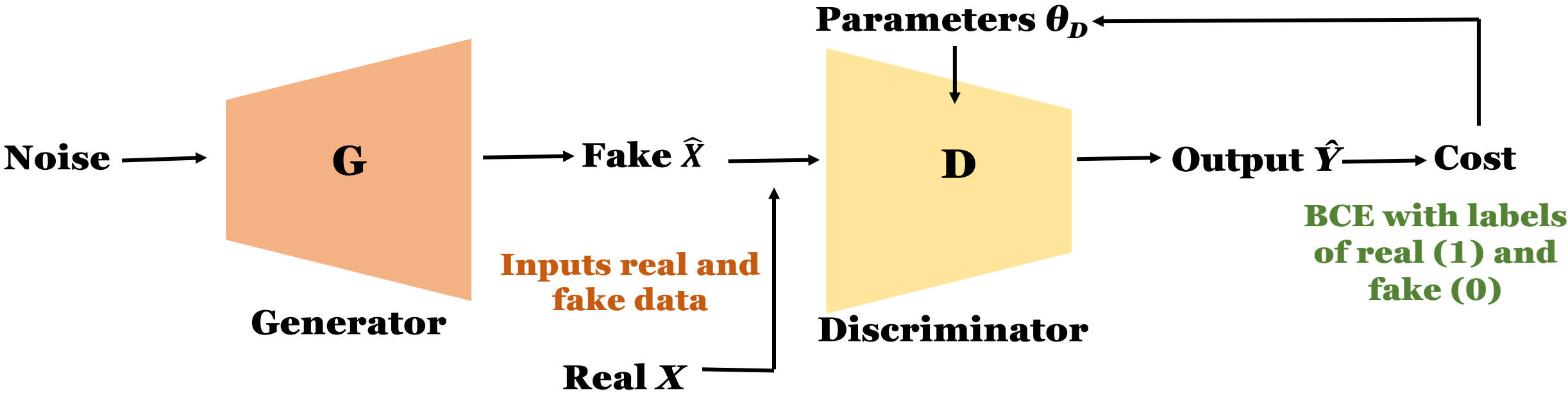
Complete GAN Model



Complete GAN Model

Training Discriminator

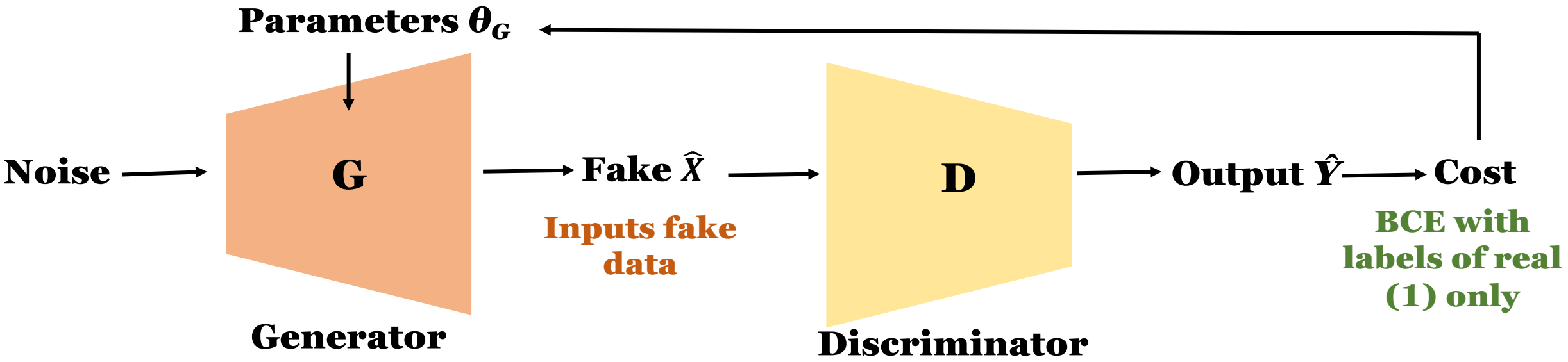
Keep generator constant



Complete GAN Model

Training Generator

Keep discriminator constant



GAN is trained in an alternative fashion

Do not use discriminator that is too strong!

Strong discriminator → **Identify fake images as 100% fake** → **Generator cannot improve**

Break Time & Quizzes

Question 1

The discriminator network and generator network influence each other solely through the data produced by the generator and the labels produced by the discriminator. When it comes to backpropagation, they are separate networks.

- ❑ **True**
- ❑ **False**

Break Time & Quizzes

Question 2

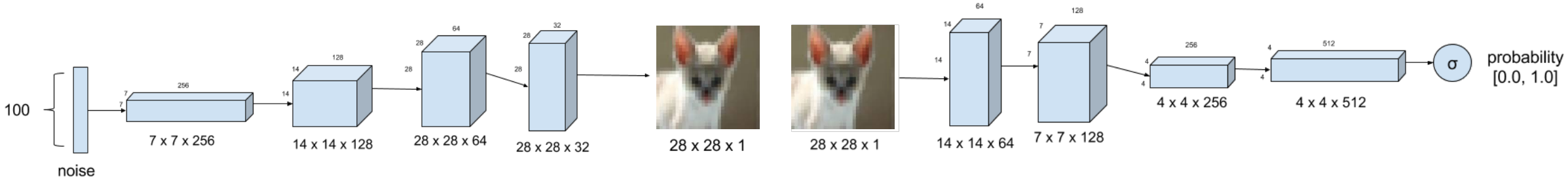
Which is NOT TRUE about the noise vectors?

- ❑ **The random noise vectors are also known as latent space vectors**
- ❑ **Generator will produce different image every time we feed the same noise vector into the network**
- ❑ **By introducing noise, the GAN can produce a wide variety of data, sampling from different places in the target distribution.**

DCGAN

Generator

Discriminator

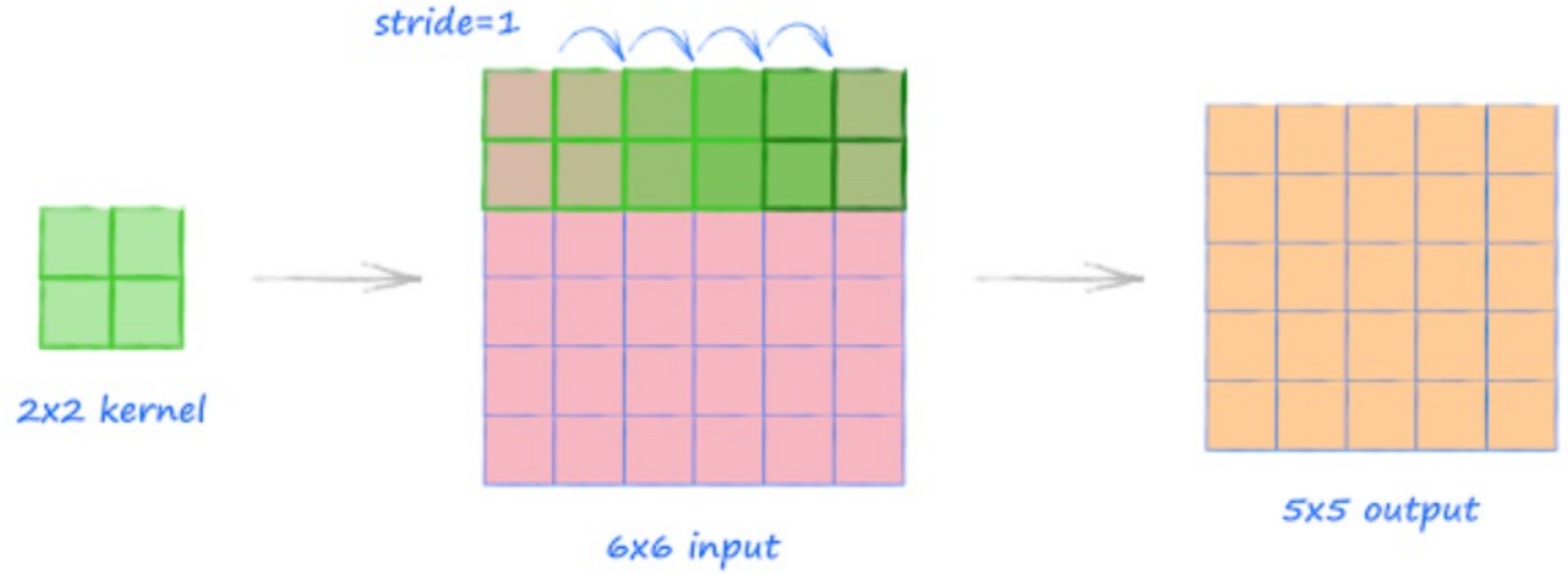


- **Replace all pooling layers with convolutional stride**
- **Use transposed convolution for upsampling**
- **Use batch normalization in both the generator and the discriminator**
- **Remove fully connected hidden layers**
- **Use ReLU activation in generator for all layers except for the output, which uses Tanh**
- **Use LeakyReLU activation in the discriminator for all layers**

Credit: https://towardsdatascience.com/gan-by-example-using-keras-on-tensorflow-backend-1a6d515a60d0

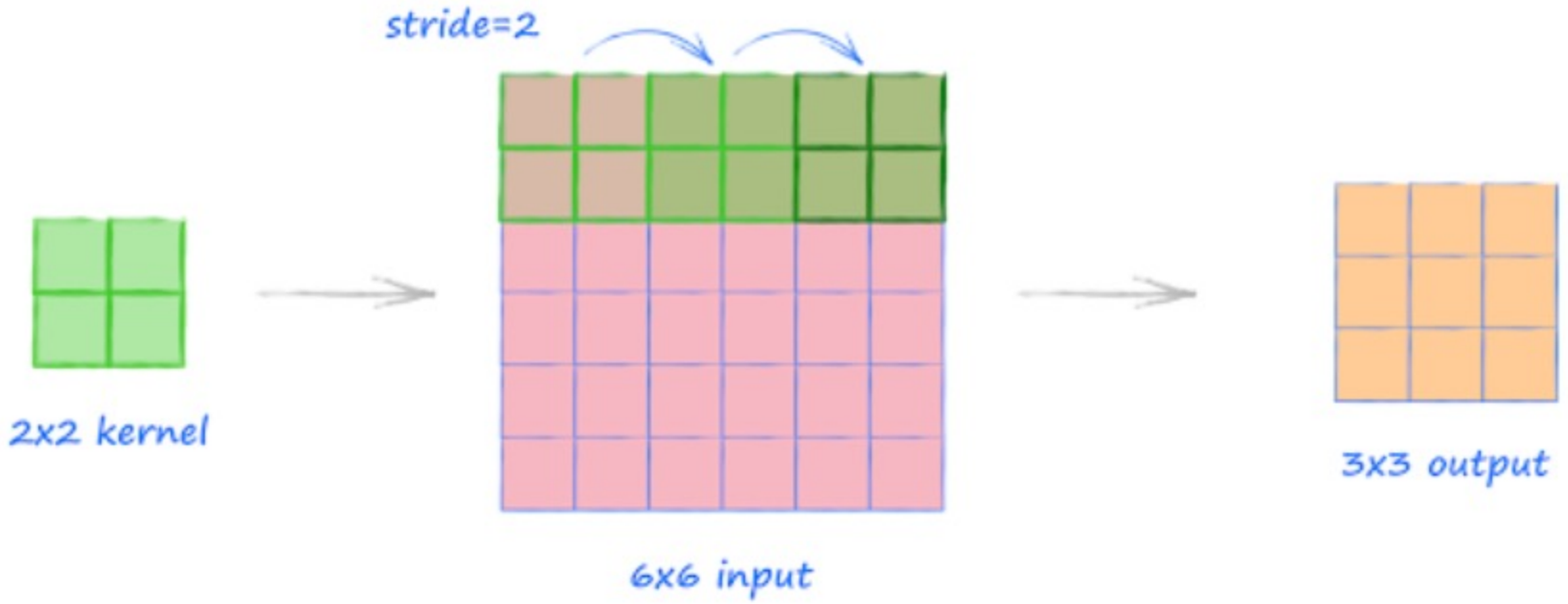
DCGAN

Discriminator: Convolution ~ downsampling



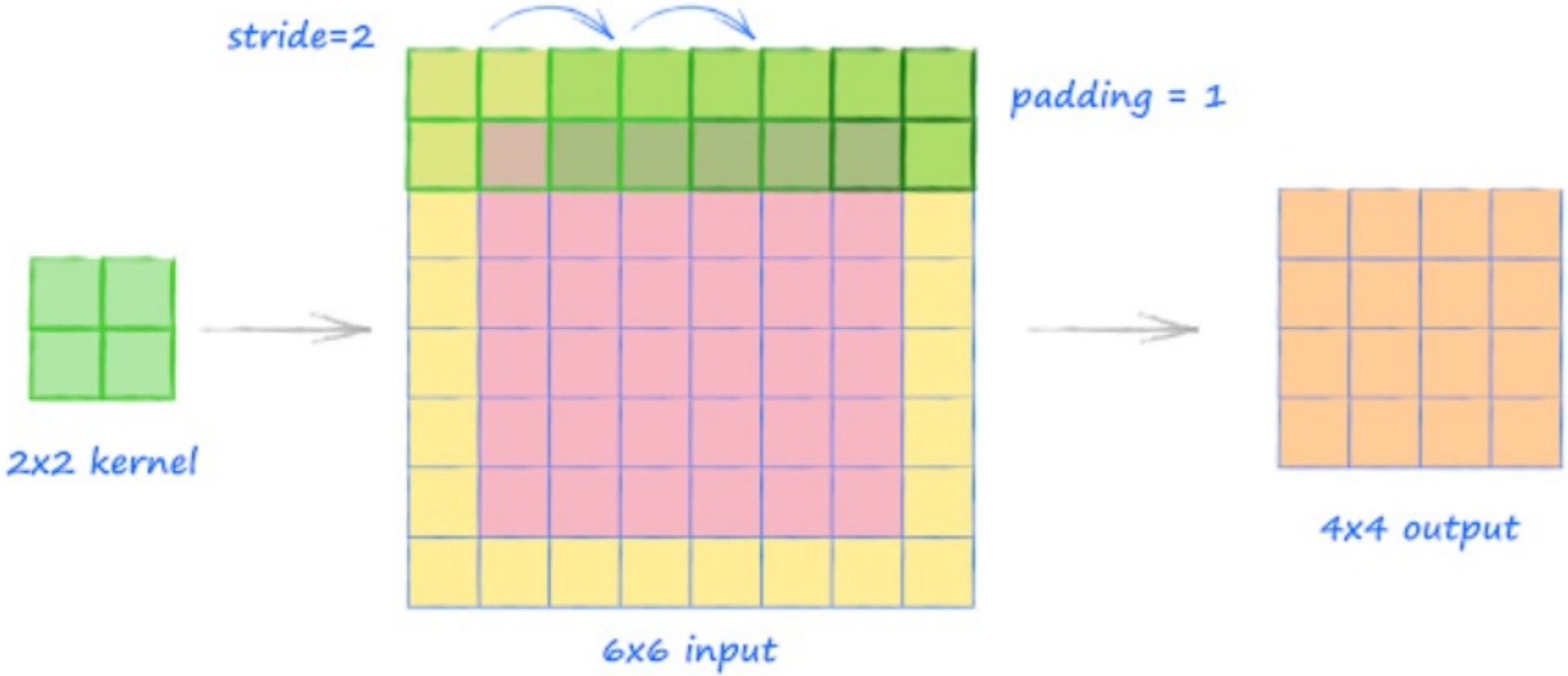
DCGAN

Discriminator: Convolution ~ downsampling



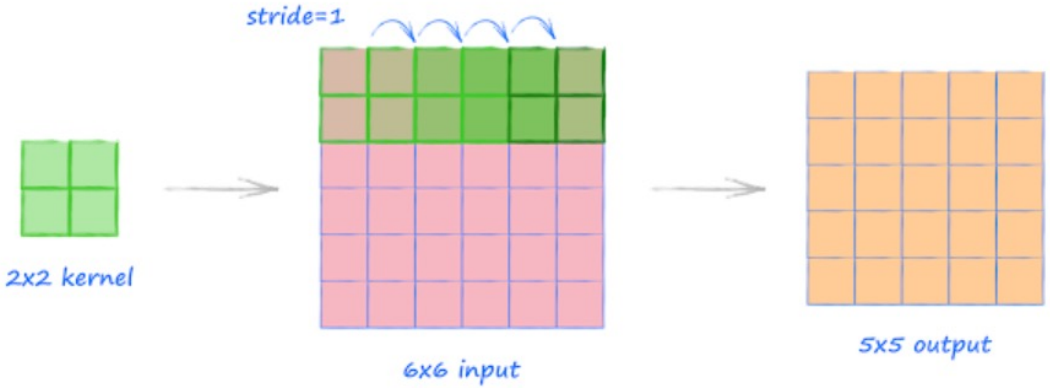
DCGAN

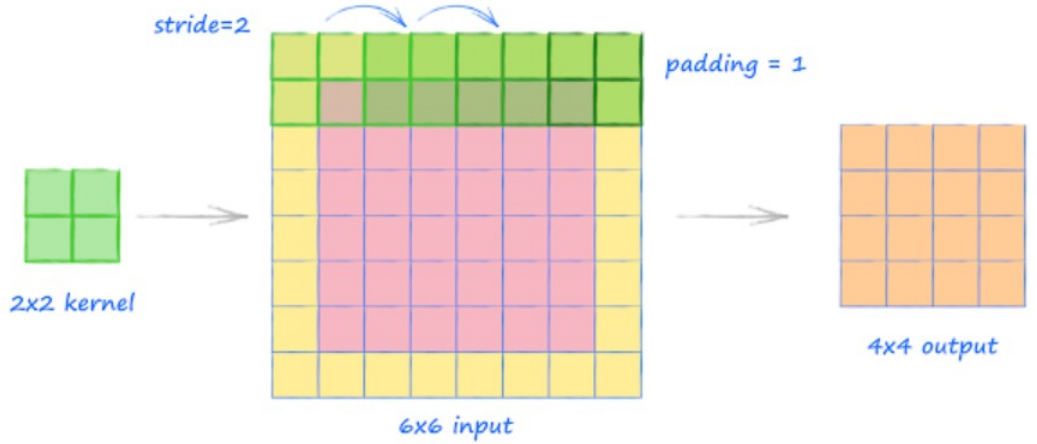
Discriminator: Convolution ~ downsampling

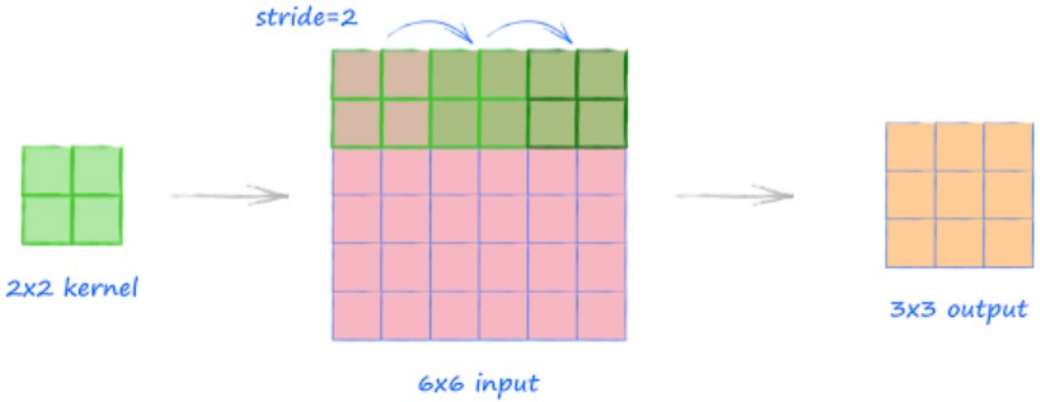


DCGAN

Discriminator: Convolution ~ downsampling



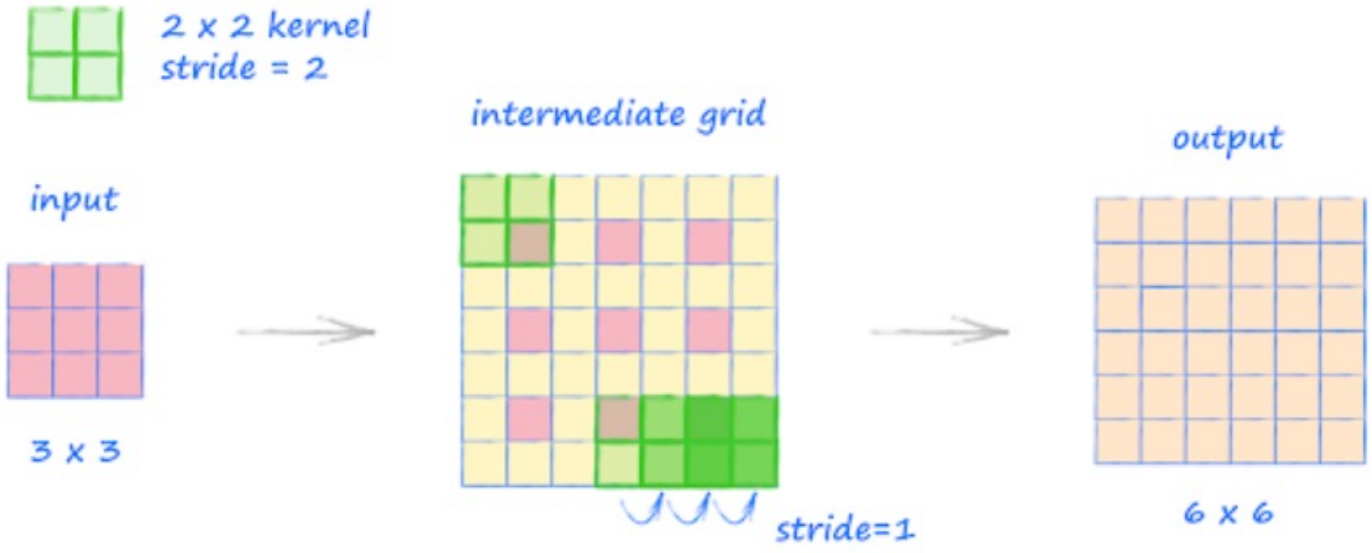




$$output\ size = \frac{input\ size + 2 * padding\ - kernel\ size}{stride} + 1$$

DCGAN

Generator: transposed convolution ~ upsampling

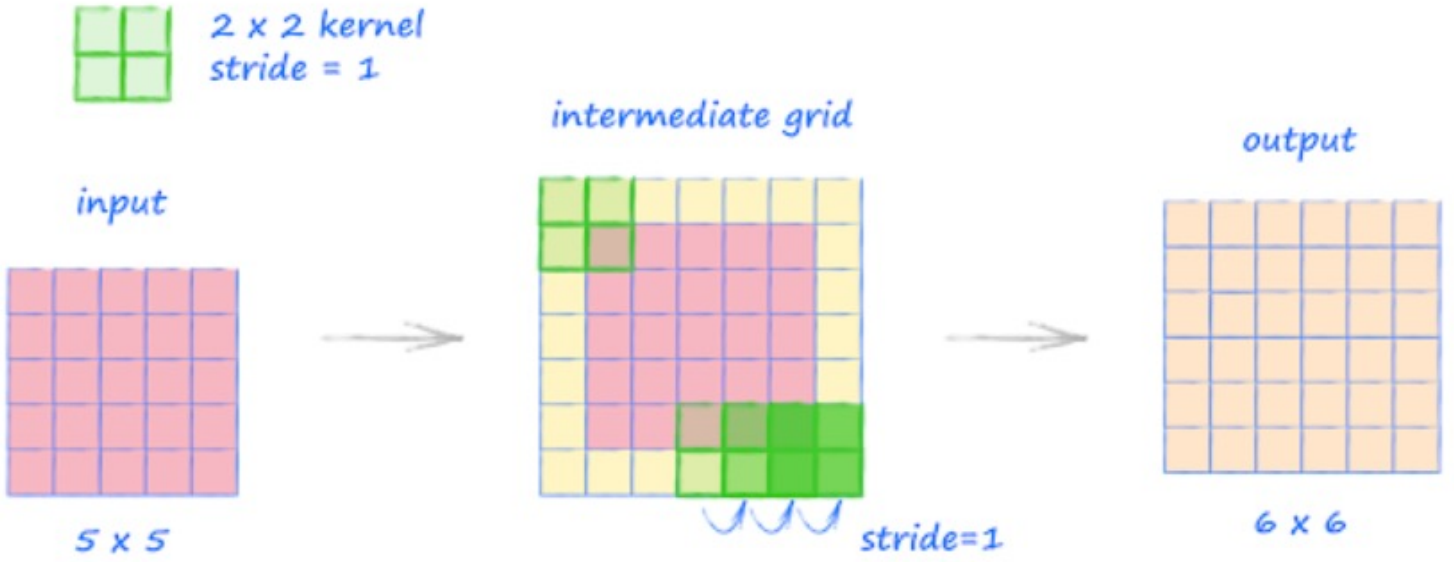


4 step process:

1. Calculate new parameters *z* and *p'*
   z = s -1; p' = k – p – 1
2. Insert ***z* number of zeros** between each rows and columns of the input
3. **Pad** the modified input images with ***p'* number of zeros**
4. Perform standard convolution with **stride length of 1**

DCGAN

Generator: transposed convolution ~ upsampling

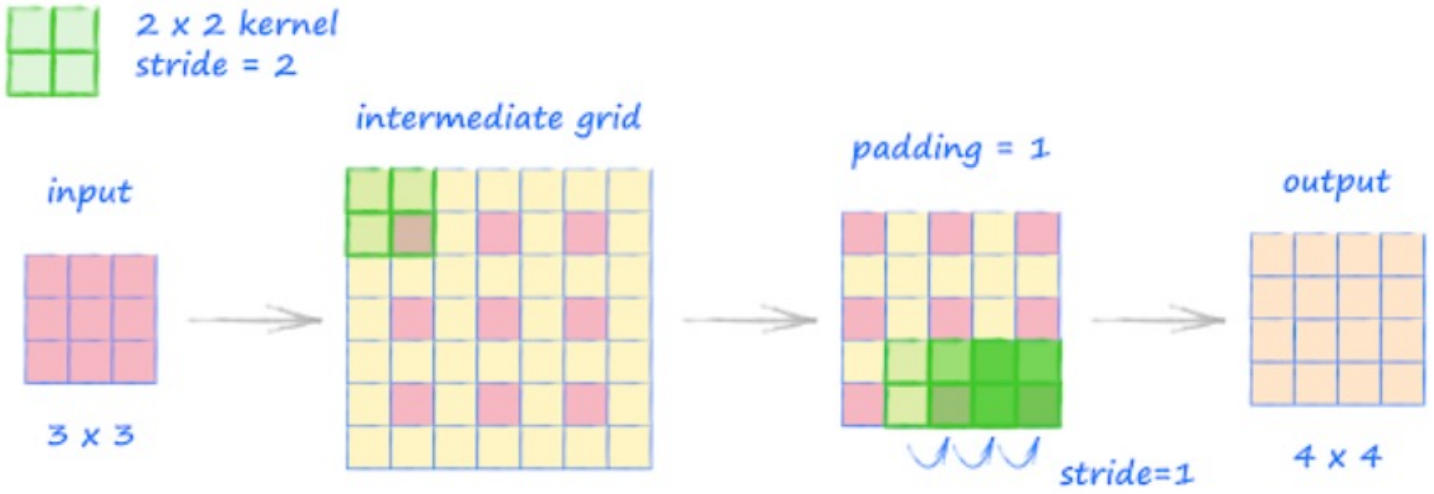


4 step process:

1. Calculate new parameters *z* and *p'*
   z = s -1; p' = k – p – 1
2. Insert ***z* number of zeros** between each rows and columns of the input
3. **Pad** the modified input images with ***p'* number of zeros**
4. Perform standard convolution with **stride length of 1**

DCGAN

Generator: transposed convolution ~ upsampling



4 step process:

1. Calculate new parameters *z* and *p'*
   z = s -1; p' = k – p – 1
2. Insert ***z* number of zeros** between each rows and columns of the input
3. **Pad** the modified input images with ***p'* number of zeros**
4. Perform standard convolution with **stride length of 1**

$$output\ size = (input\ size\ - 1) * stride\ - 2 * padding + kernel\ size$$

Hands-on Session

Build a DCGAN to generate CIFAR images

Exercise:

Replace ‘None’ with your own code.

THANKS FOR YOUR PARTICIPATION

We value your feedback!

https://forms.gle/jR4Xpy2ywFtyM2wm7

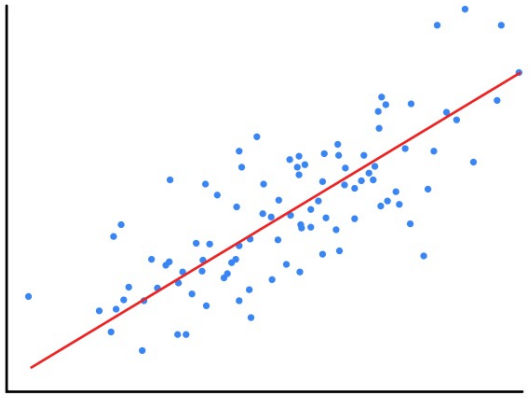
Fill in the feedback form to get the complete slides and notebook!

EXTRA MATERIALS

Supervised vs. Unsupervised Learning

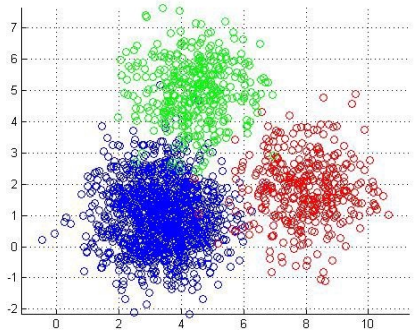
Supervised

- Data: (x, y)
  x is data, y is label
- Goal: Learn function to map x -> y
- Examples: Regression, classification, object detection, etc.



Unsupervised

- Data: x
  x is data, no labels
- Goal: Learn some hidden or underlying structure of the data
- Examples: Clustering, dimensionality reduction, feature learning, etc.



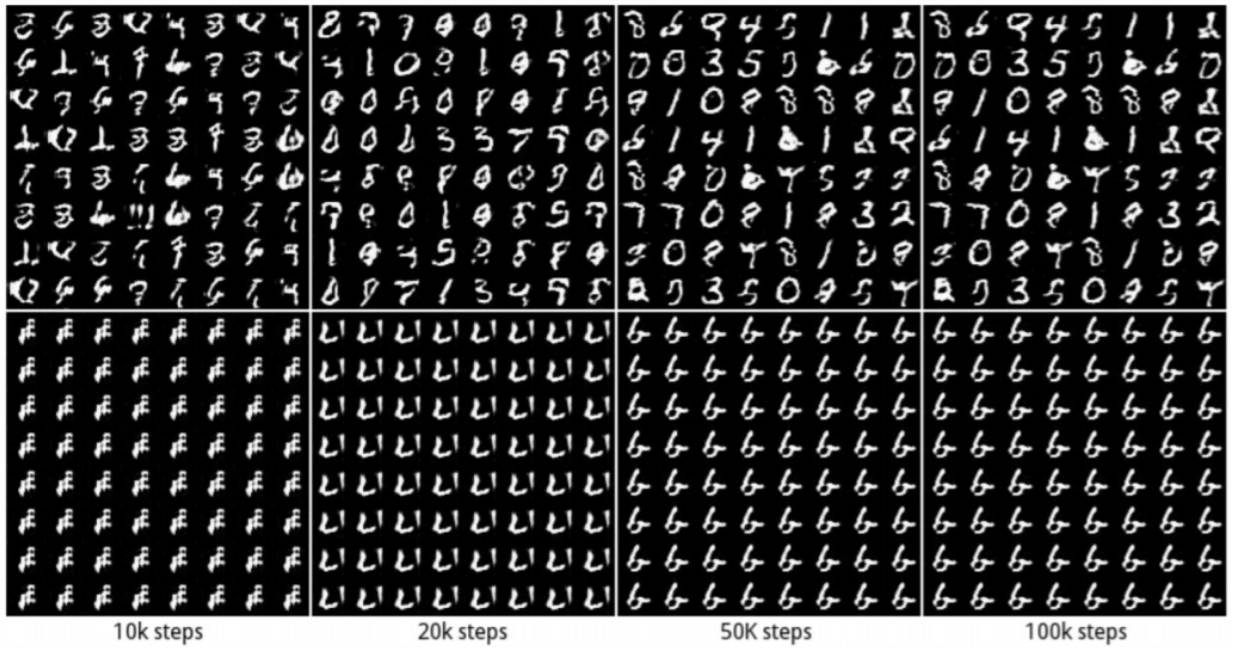
DCGAN

Batch Normalization

- **Stabilize generator's learning process**
- **Prevent mode collapse**

10 modes

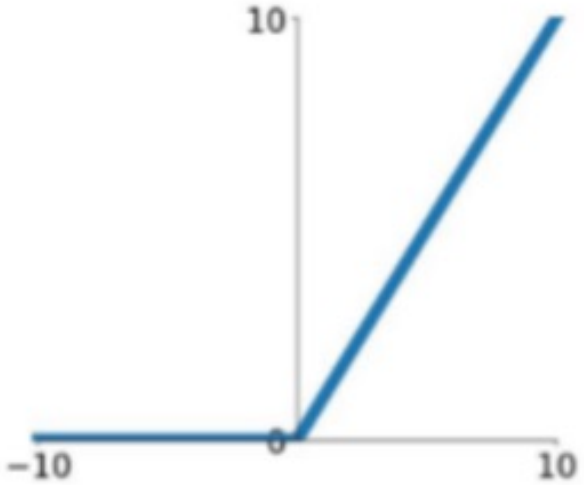
Collapse to 1 mode



DCGAN

Activations:

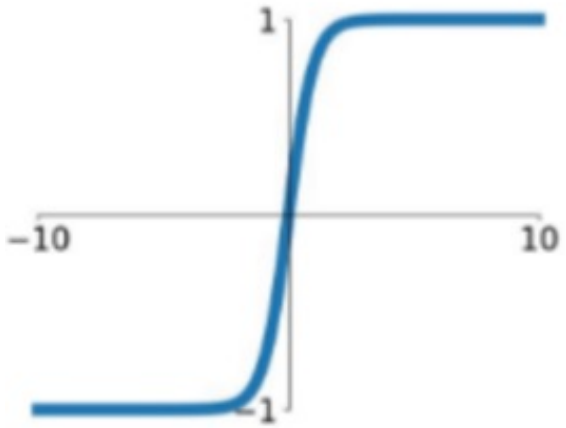
ReLU



$$a = \max(0, z)$$

Every layer of generator, except the last layer

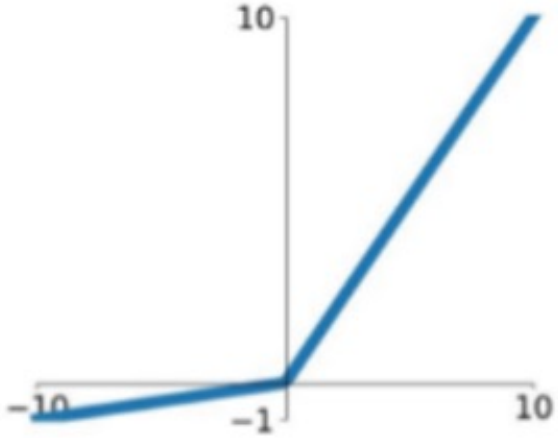
tanh



$$a = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

Last layer of generator

Leaky ReLU



$$a = \max(0.2z, z)$$

Every layer of discriminator