Lecture 29 : Introduction to Generative Adversarial Networks (GANS)

CS109B Data Science 2

Pavlos Protopapas, Mark Glickman, and Chris Tanner


"Generative Adversarial Networks is the **most interesting idea in the last ten years** in machine learning."
Yann LeCun, Director, Facebook AI

Outline

- Generative Modeling Motivation
- High Level Formalism
- Mathematics
- Architecture
- Conditional GANS

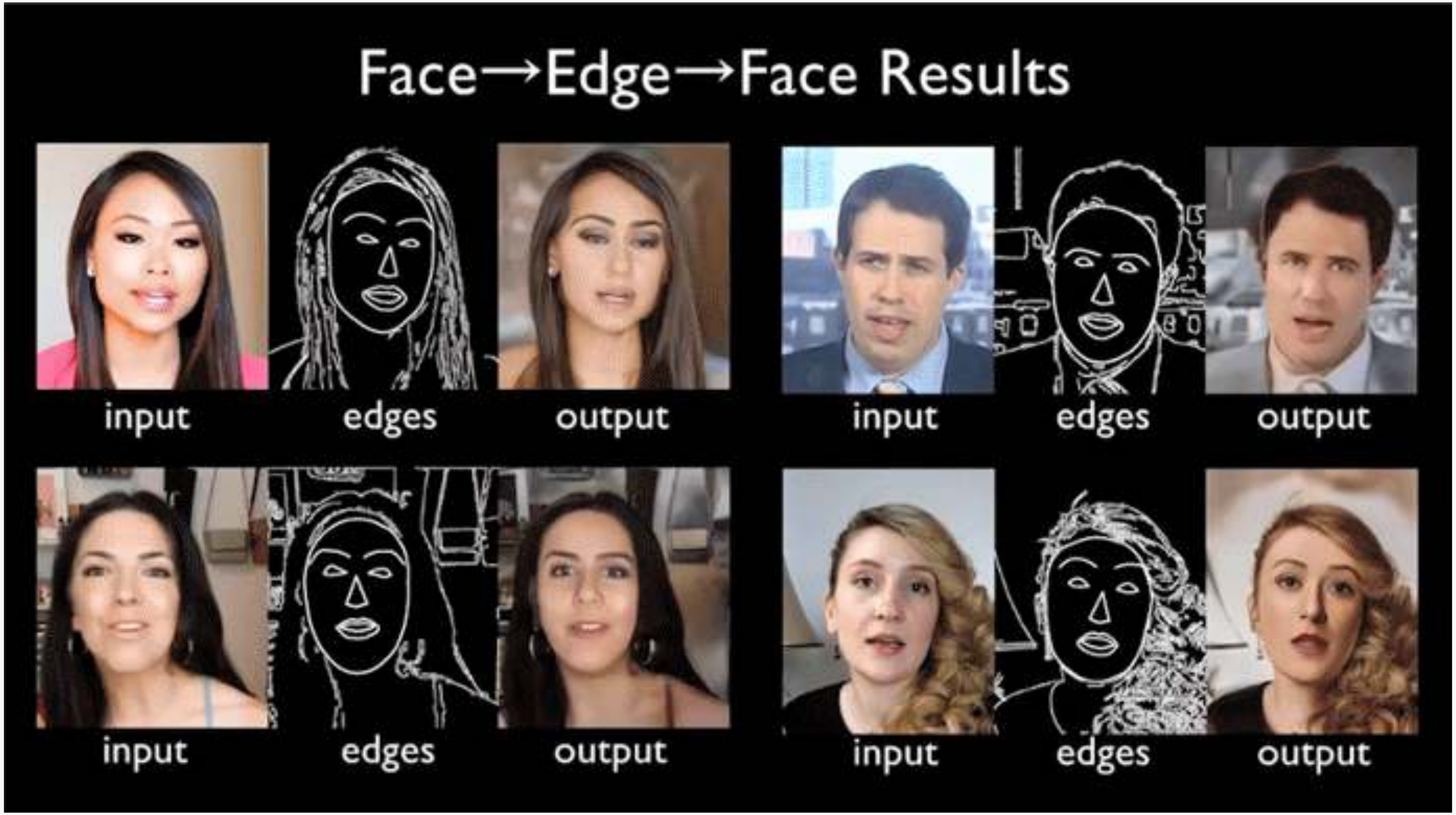
Outline

- **Generative Modeling Motivation**
- High Level Formalism
- Mathematics
- Architecture
- Conditional GANS

Unpaired Image-to-Image Translation using Cycle-GANs

[Zhu et al. 2017]

Video-to-Video Synthesis

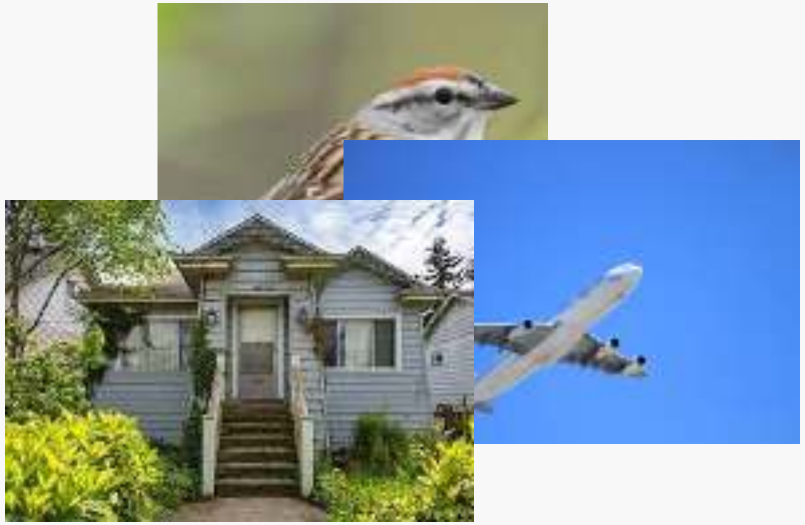


[Wang et al. 2018]

What is generative modeling?

Given samples $\sim p_{\text{data}}$, we would like to sample from the same distribution?

Training data $\sim p_{\text{data}}(x)$



Generated samples $\sim p_{\text{model}}(x)$

What is generative modeling?

How do we generate samples from the same distribution as p_{data}(x) ?

Explicit sampling: p_{model}(x) has analytical expression:

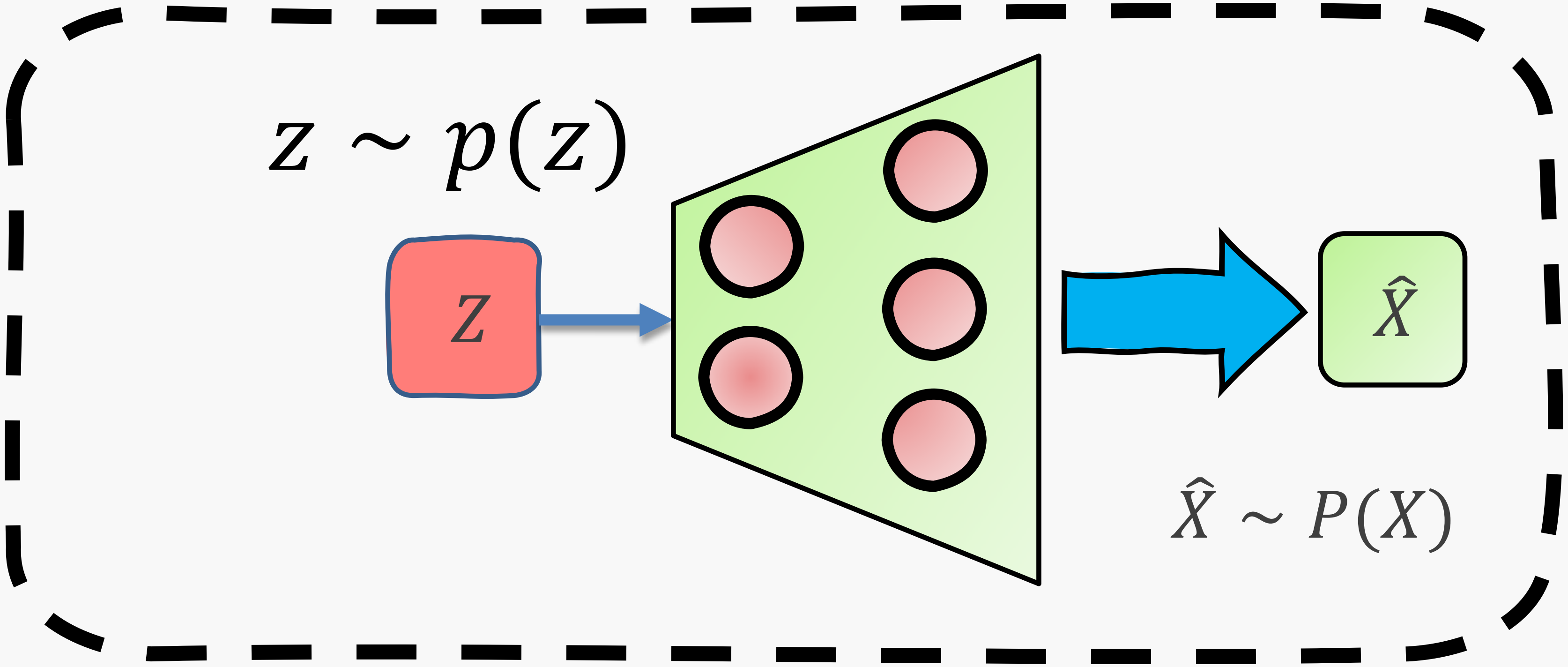
- MCMC
- Variational methods

Implicit sampling: learn only how to sample from p_{data}(x)

- Generator part of VAR
- GANS

Variational Auto Encoder as a generative model

Generative model



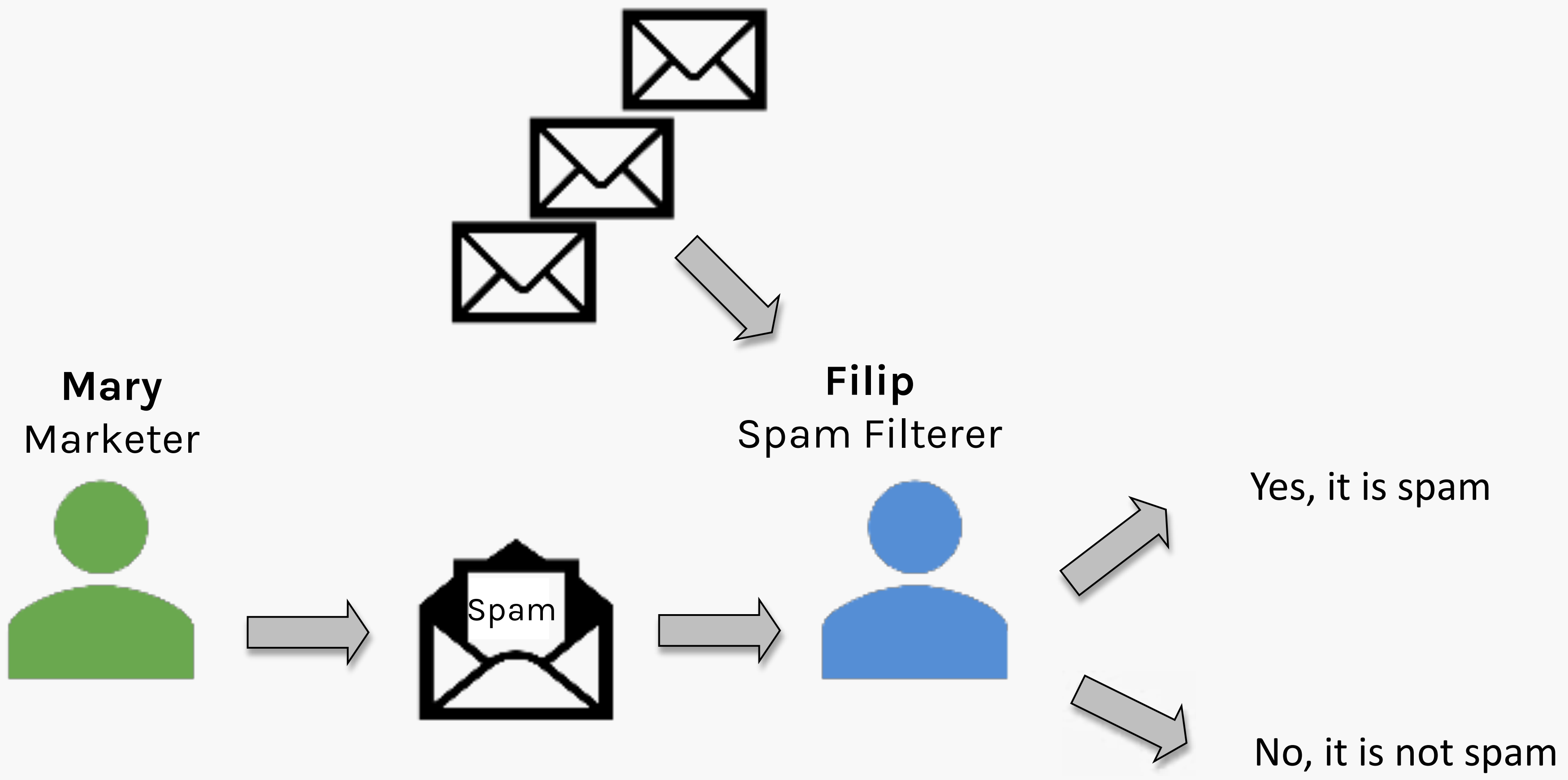
Though we used Variational inference to sample of the latent space, at the end we created a model that given z in generates $\hat{X}$ with a distribution similar to X.

Why should we study it?

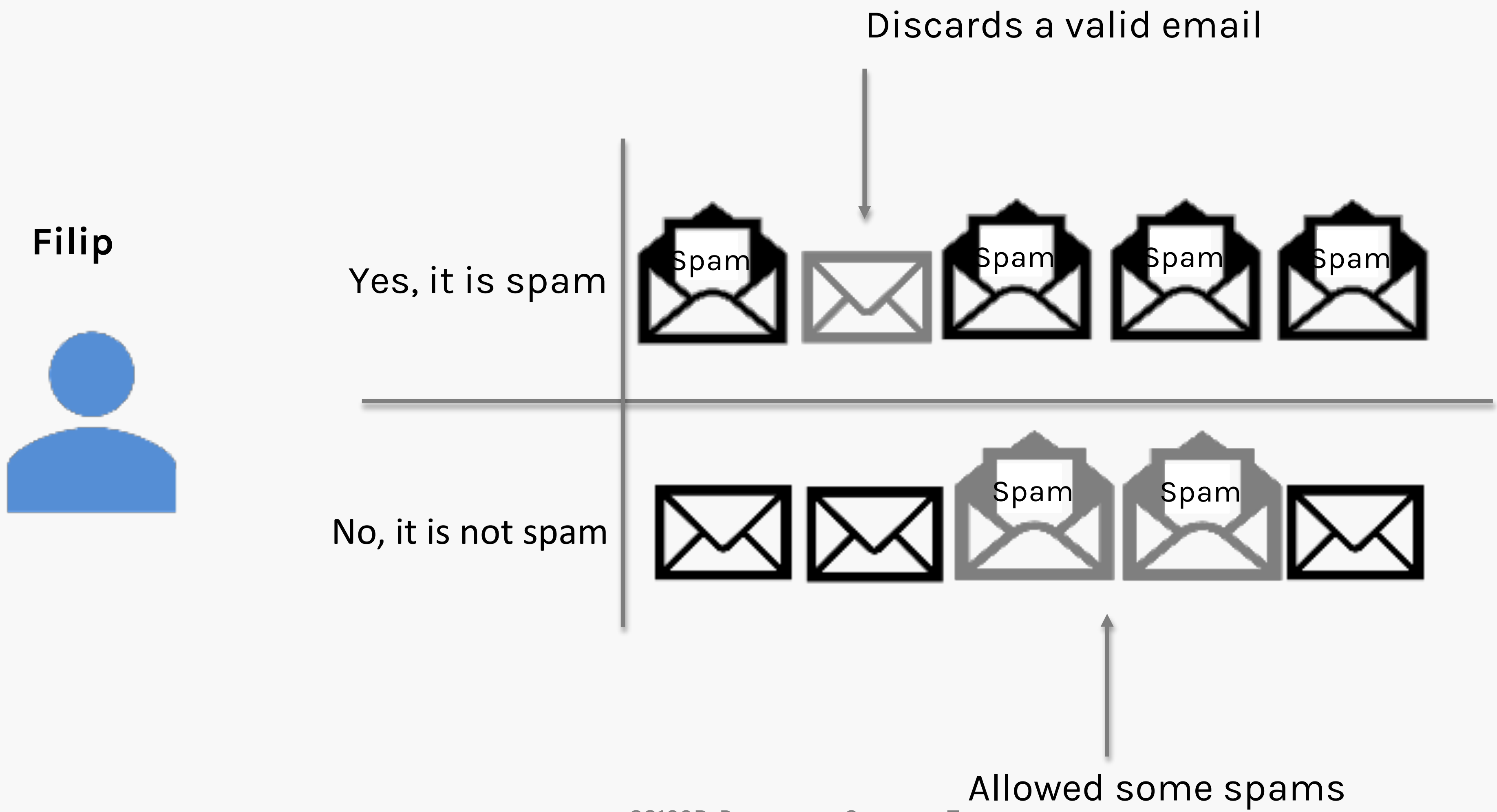
1. Realistic generation tasks
2. Debiasing and data augmentation
3. Missing data
4. Simulation and planning (RL)

[MIT 6.S191: Introduction to Deep Learning]

Generative Adversarial Networks (GANs)

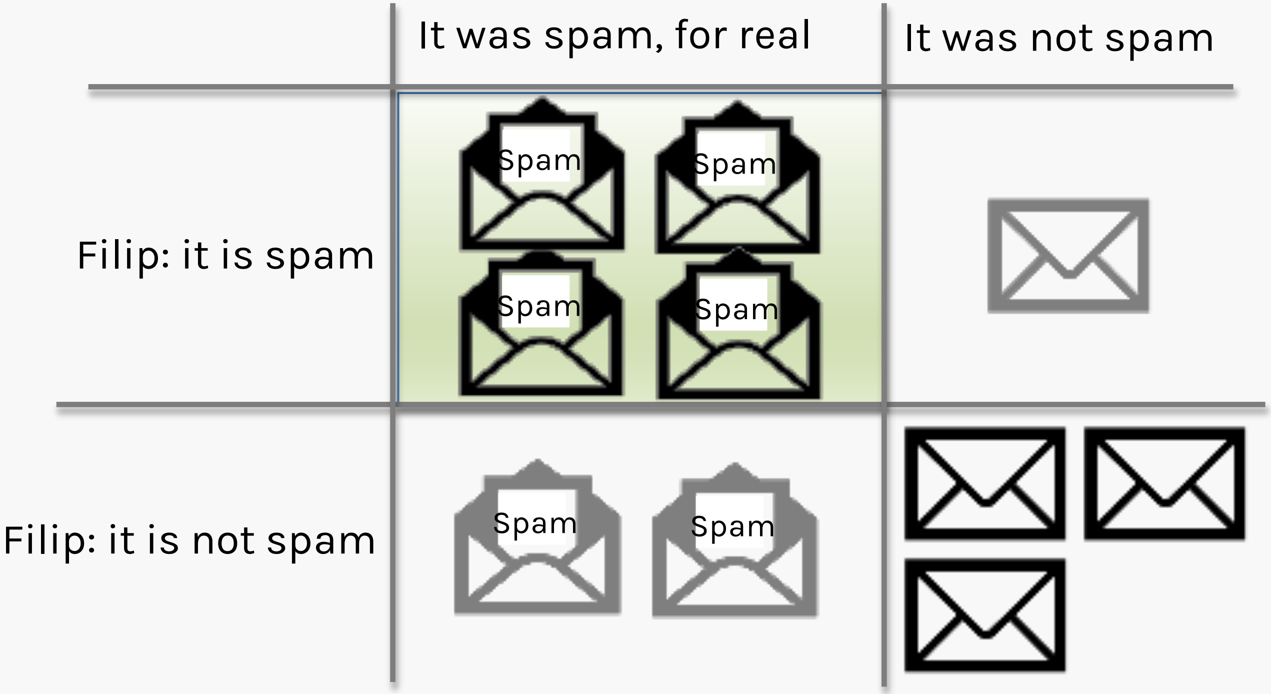


Generative Adversarial Networks (GANs)



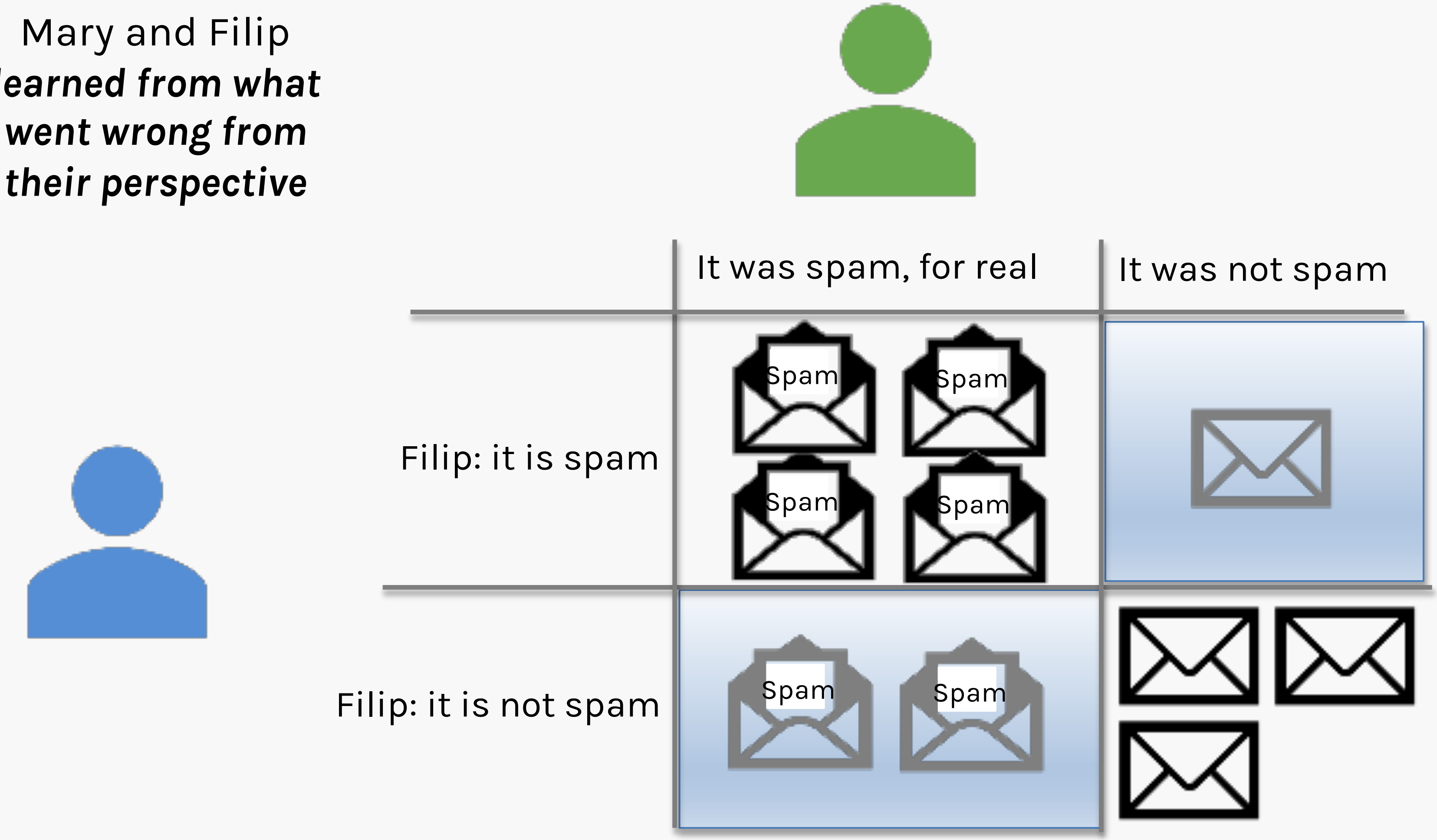
Generative Adversarial Networks (GANs)

Mary and Filip
learned from what went wrong from their perspective

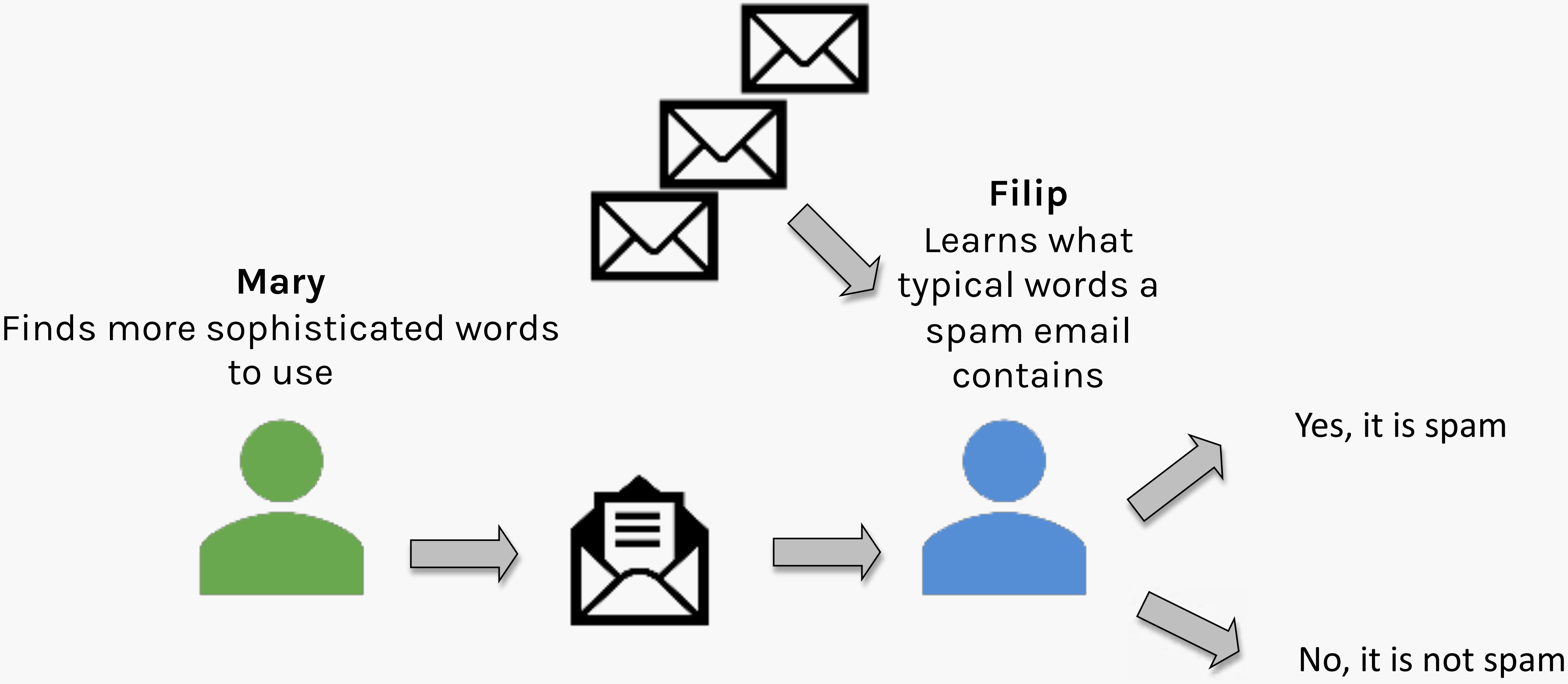


Generative Adversarial Networks (GANs)

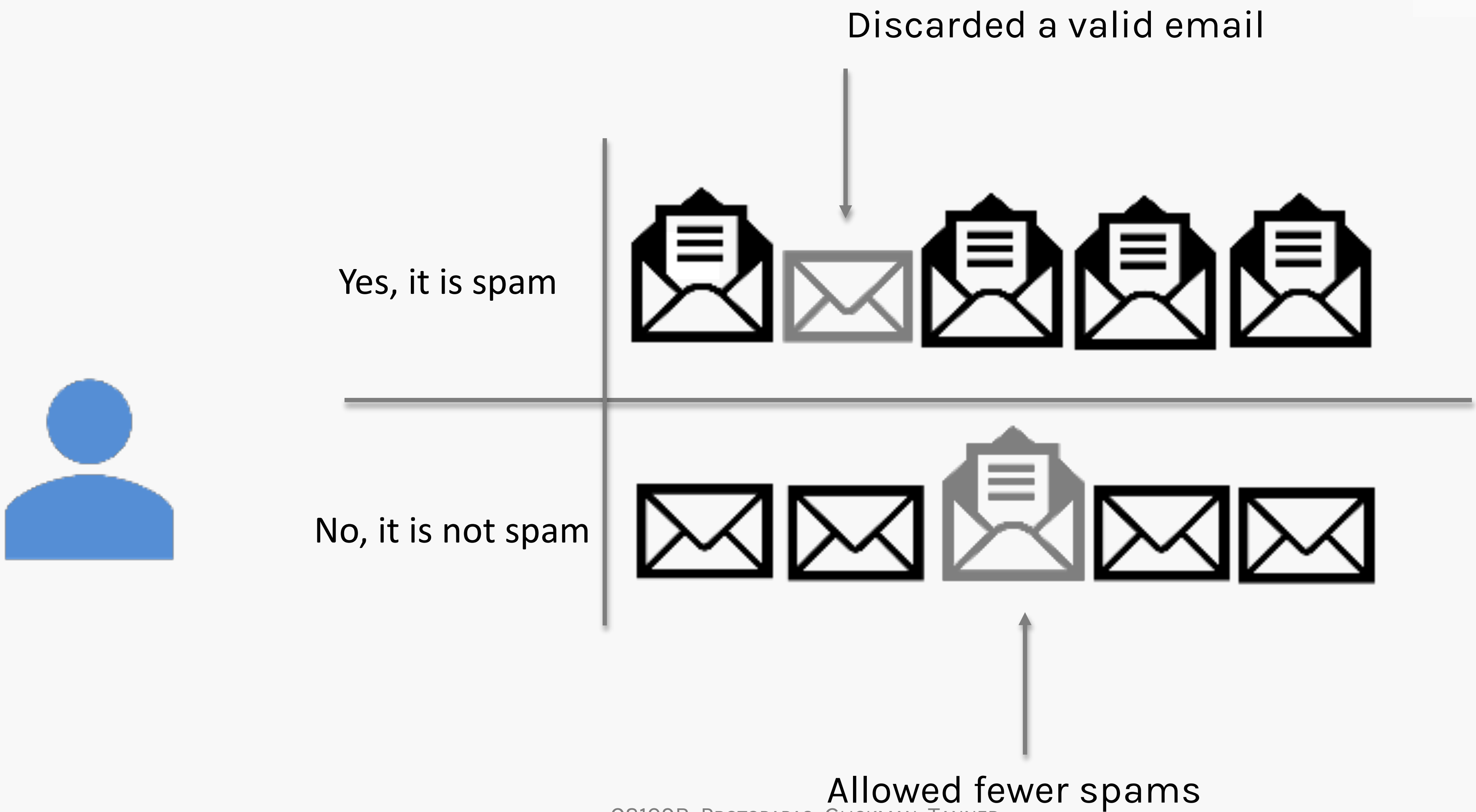
Mary and Filip ***learned from what went wrong from their perspective***



Generative Adversarial Networks (GANs)

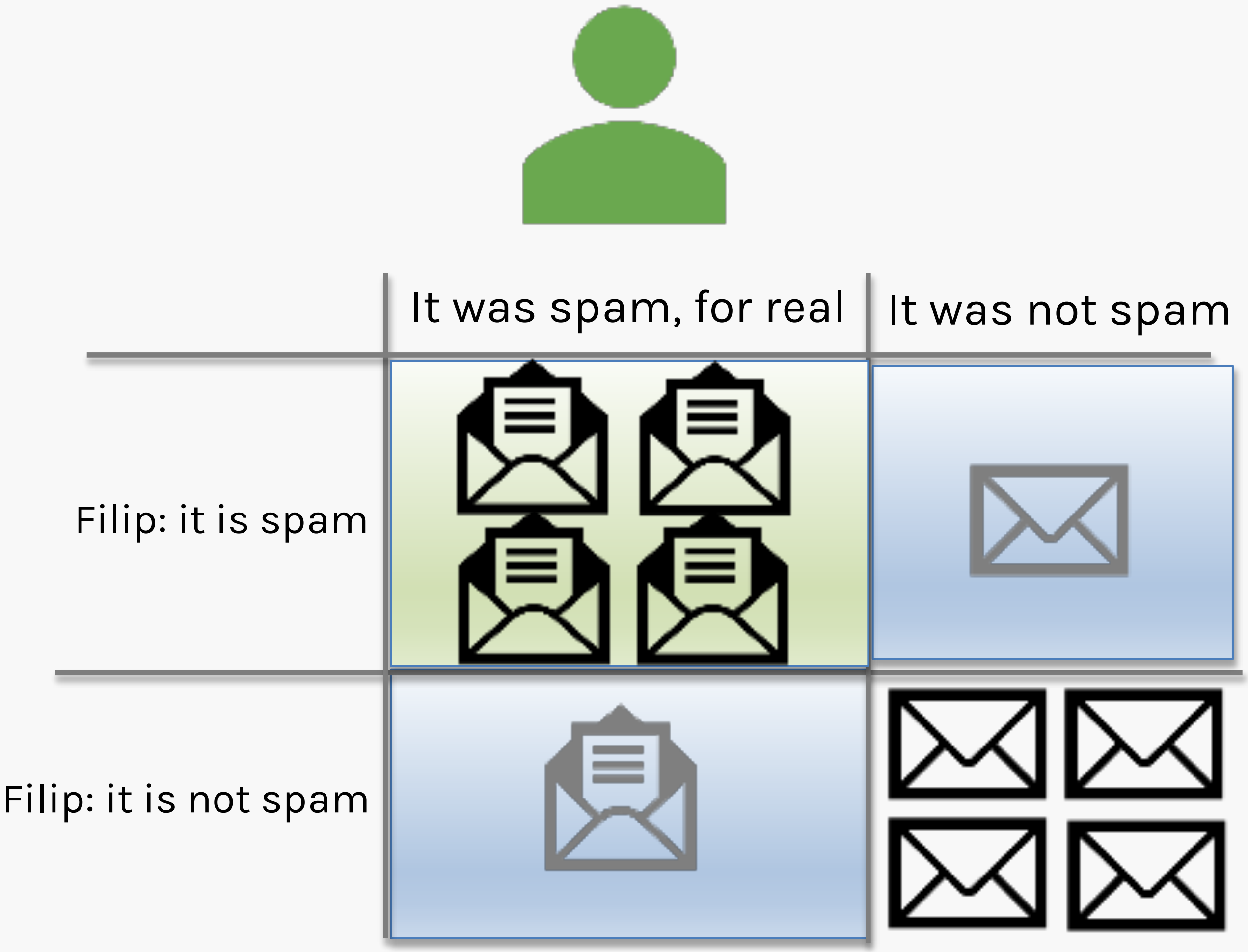


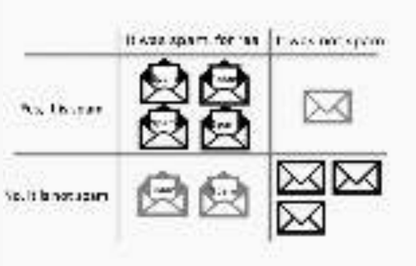
Generative Adversarial Networks (GANs)



Generative Adversarial Networks (GANs)

Mary and Filip
learned from what went wrong from their perspective





Generative **Adversarial** Networks (GANs)

Adversaries: Mary and Filip

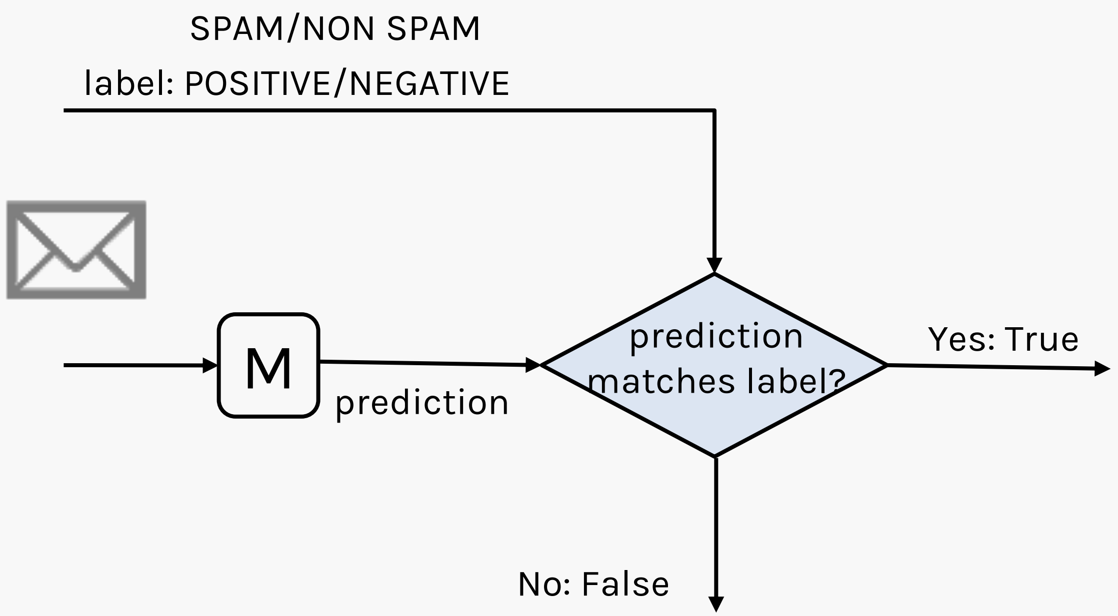
Mary
Generator

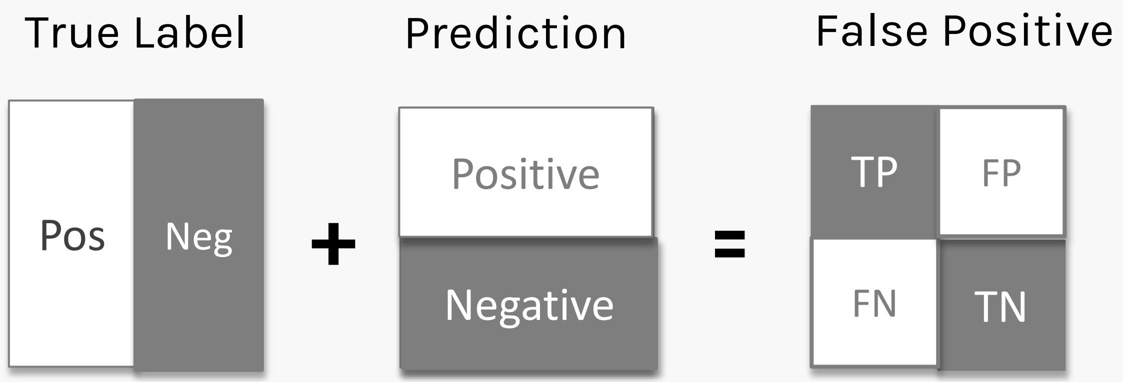
Filip
Discriminator

Generative Adversarial Networks (GANs)

Understanding confusion matrix

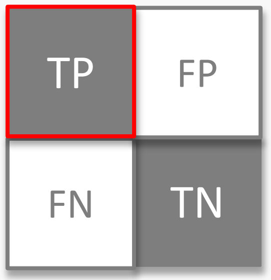
TRUE/FALSE: If prediction and true label match / do not match
POSITIVE/NEGATIVE: Prediction class (SPAM = POSITIVE)

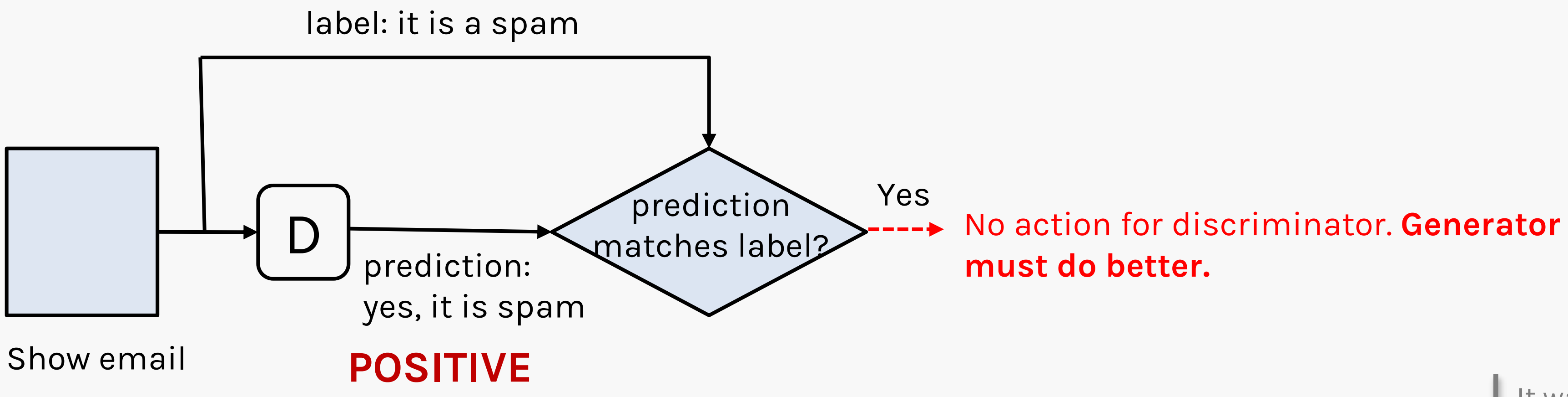


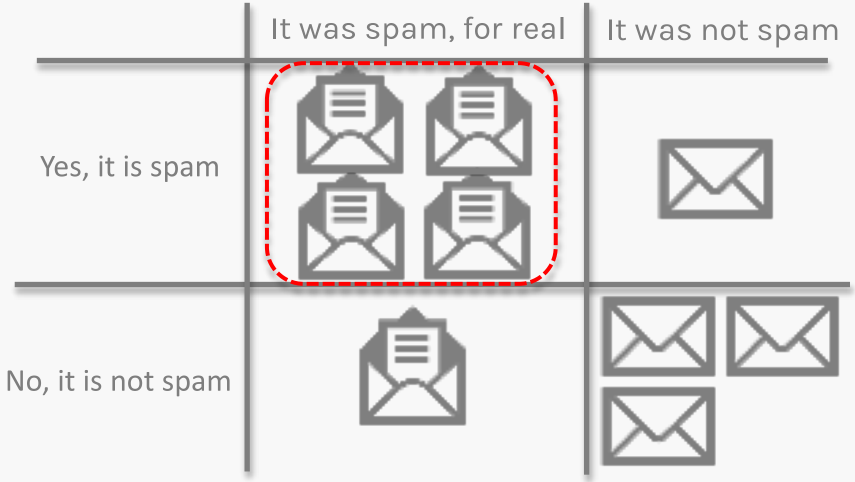


Generative Adversarial Networks (GAN)

TRUE/FALSE: If prediction and true label match / do not match
POSITIVE/NEGATIVE: Prediction class (SPAM = POSITIVE)





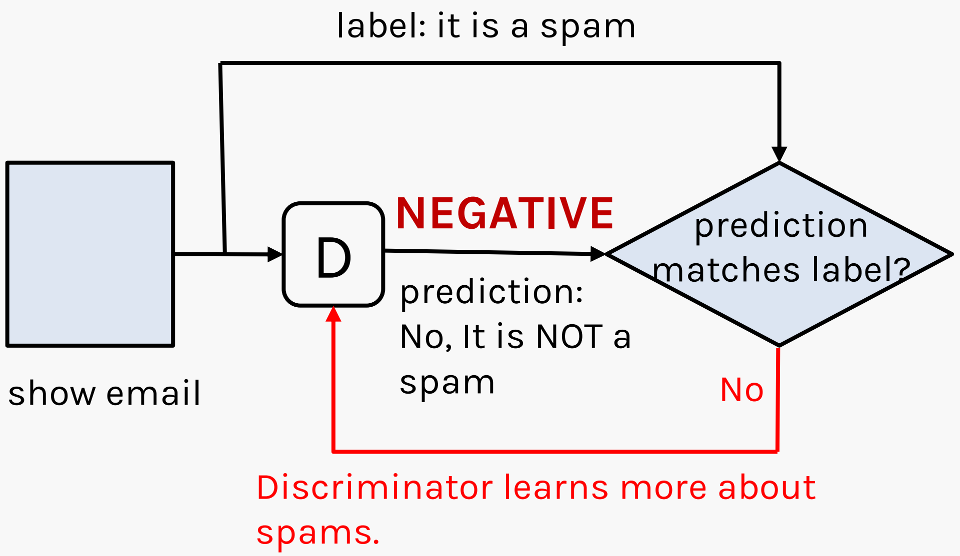


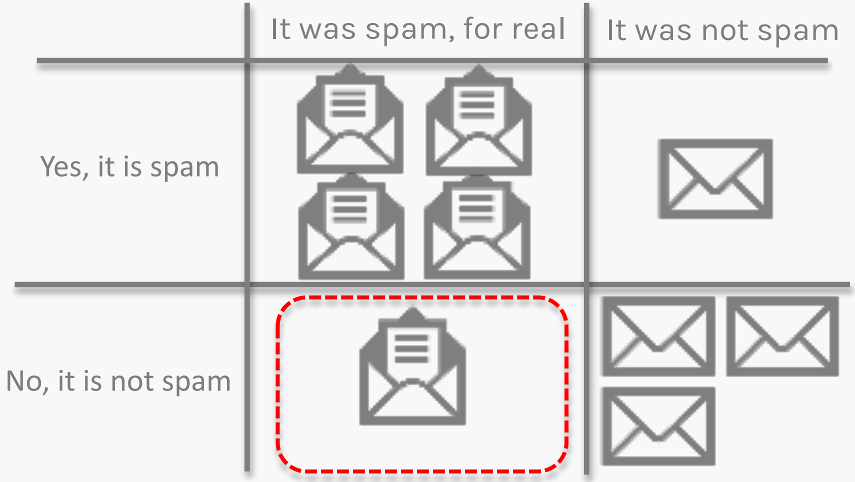
True positive (TP): the discriminator sees a spam and predicts correctly. No need for further actions for discriminator. Generator must do a better job.

Generative Adversarial Networks (GANs)

TRUE/FALSE: If prediction and true label match / do not match
POSITIVE/NEGATIVE: Prediction class (SPAM = POSITIVE)

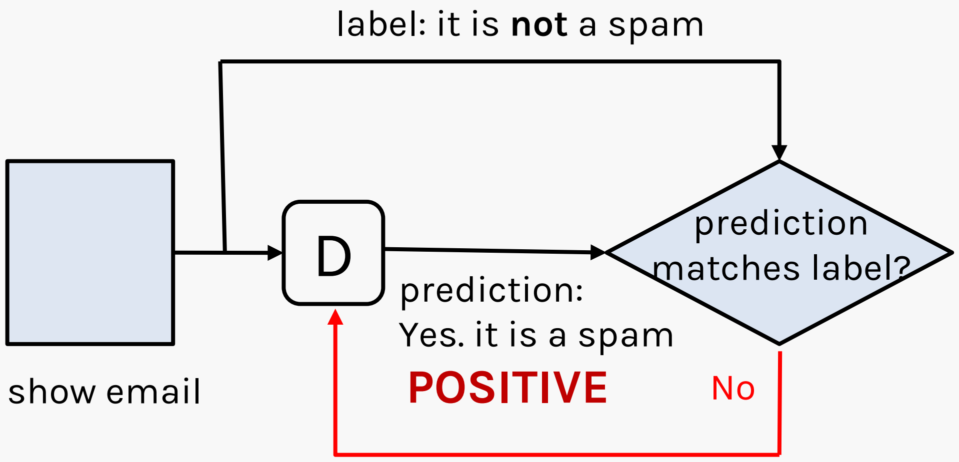
TP	FP
FN	TN



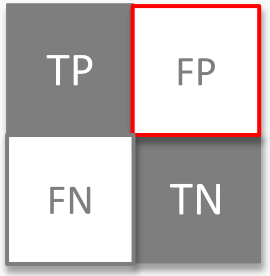


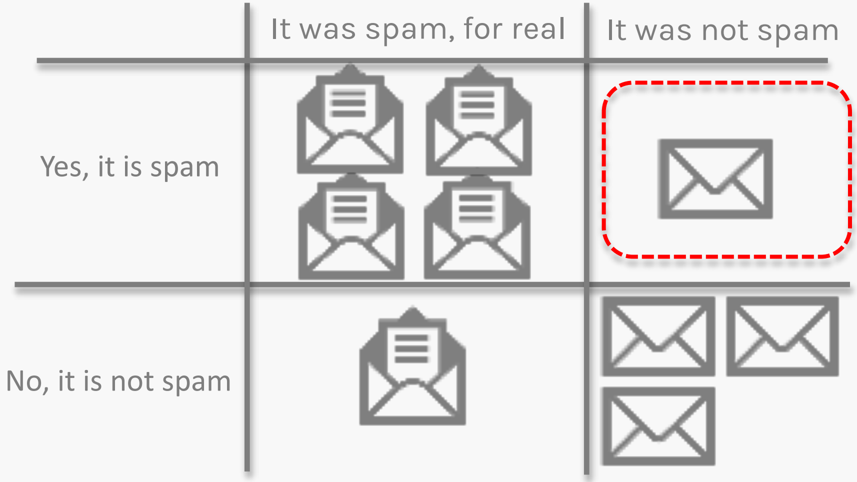
False Negative (FN): the discriminator sees an email and predicts it not a spam even though it is. The discriminator must learn more.

Generative Adversarial Networks (GANs)



Discriminator learns more about spams.



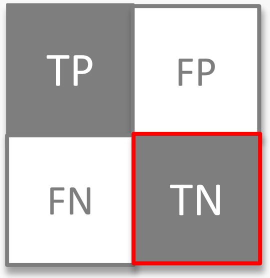


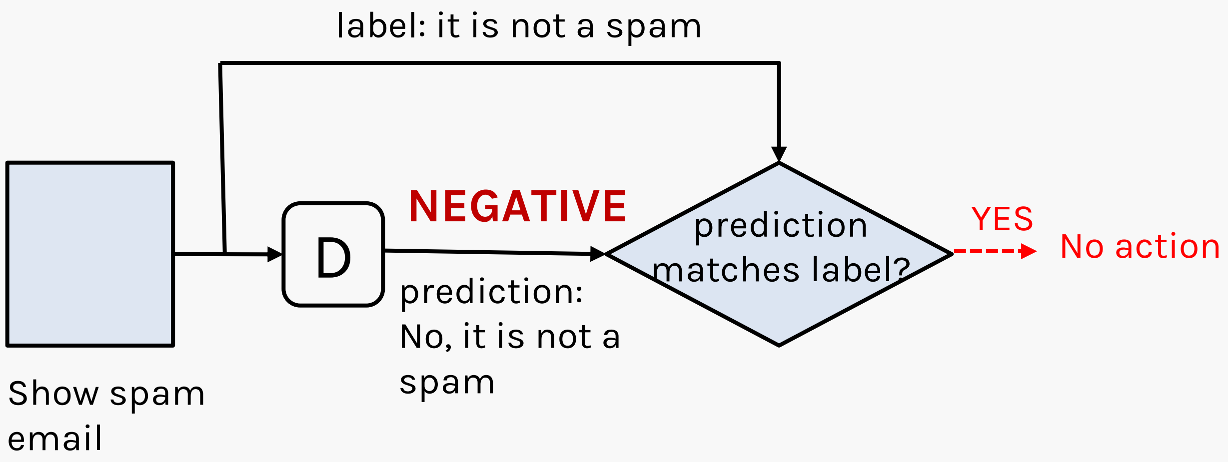
False Positive (FP): the discriminator sees an email and predicts it is a spam even though it is NOT. The discriminator must learn more.

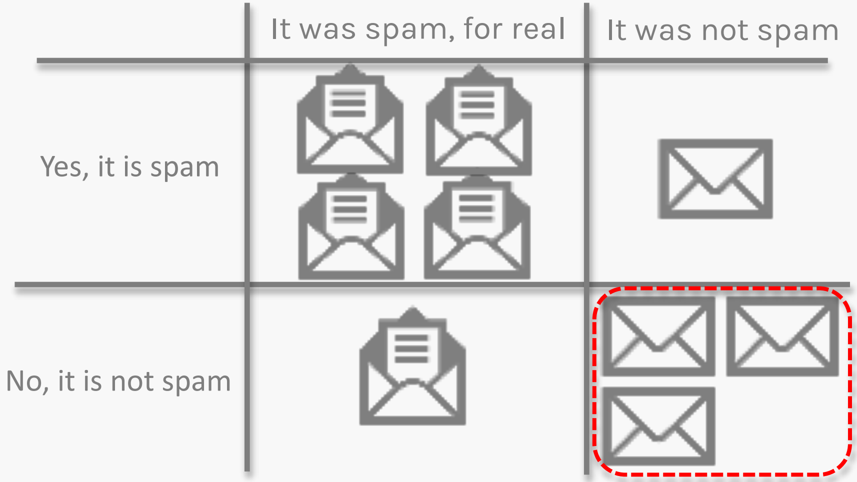
TRUE/FALSE: If prediction and true label match / do not match
POSITIVE/NEGATIVE: Prediction class (SPAM = POSITIVE)

Generative Adversarial Networks (GANs)

TRUE/FALSE: If prediction and true label match / do not match
POSITIVE/NEGATIVE: Prediction class (SPAM = POSITIVE)







True negative (TN): No action required by Generator or Discriminator.

Generative **Adversarial** Networks (GANs)

Adversaries: Mary and Filip

Mary
Generator

Filip
Discriminator

Generative **Adversarial** Networks (GANs)

Adversaries: Mary and Filip
Become: Two player game between a **generator** G and a **discriminator** D.

Discriminator

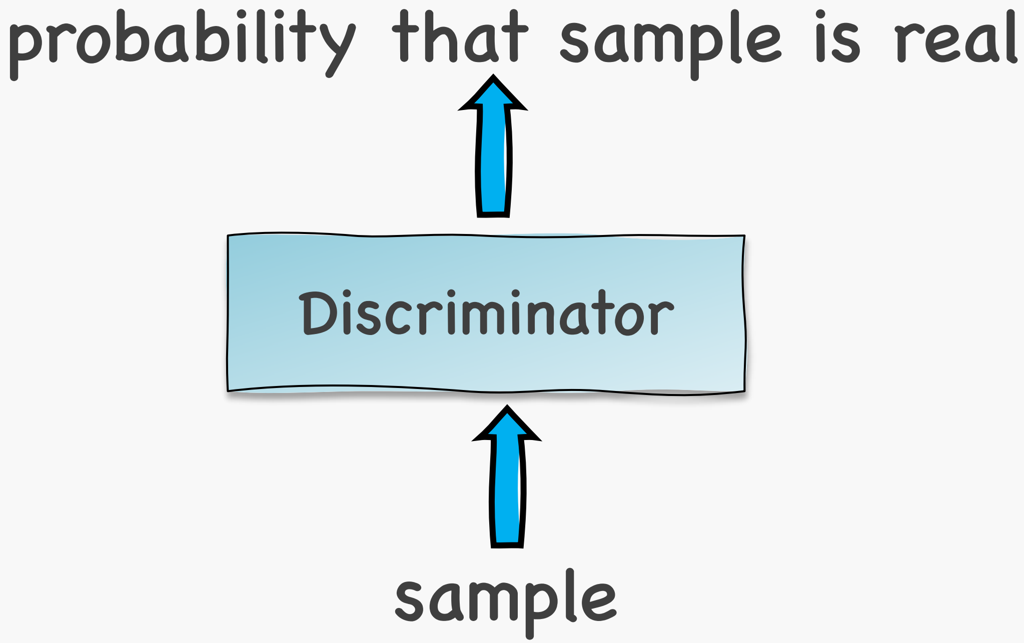
Generator

Why is it a “game” ?

The Discriminator

The discriminator is very simple. It takes a sample as input, and its output is a single value that reports the network's probability that the input is from the training set rather than being a fake.

There are not many restrictions on what the discriminator is.

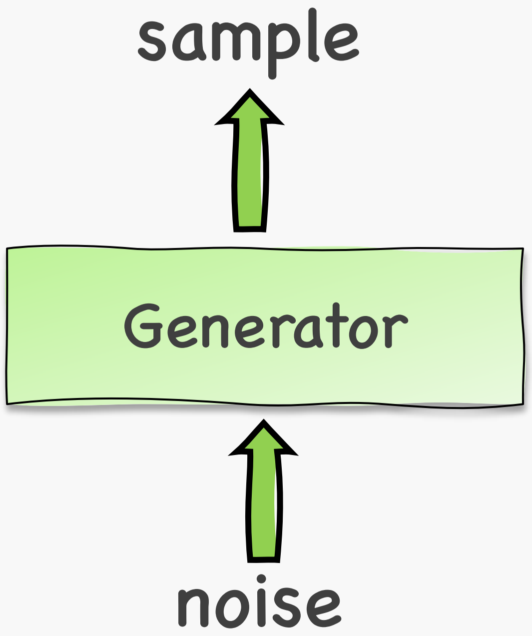


The Generator

The generator takes as input a bunch of **random numbers and generates a sample.**

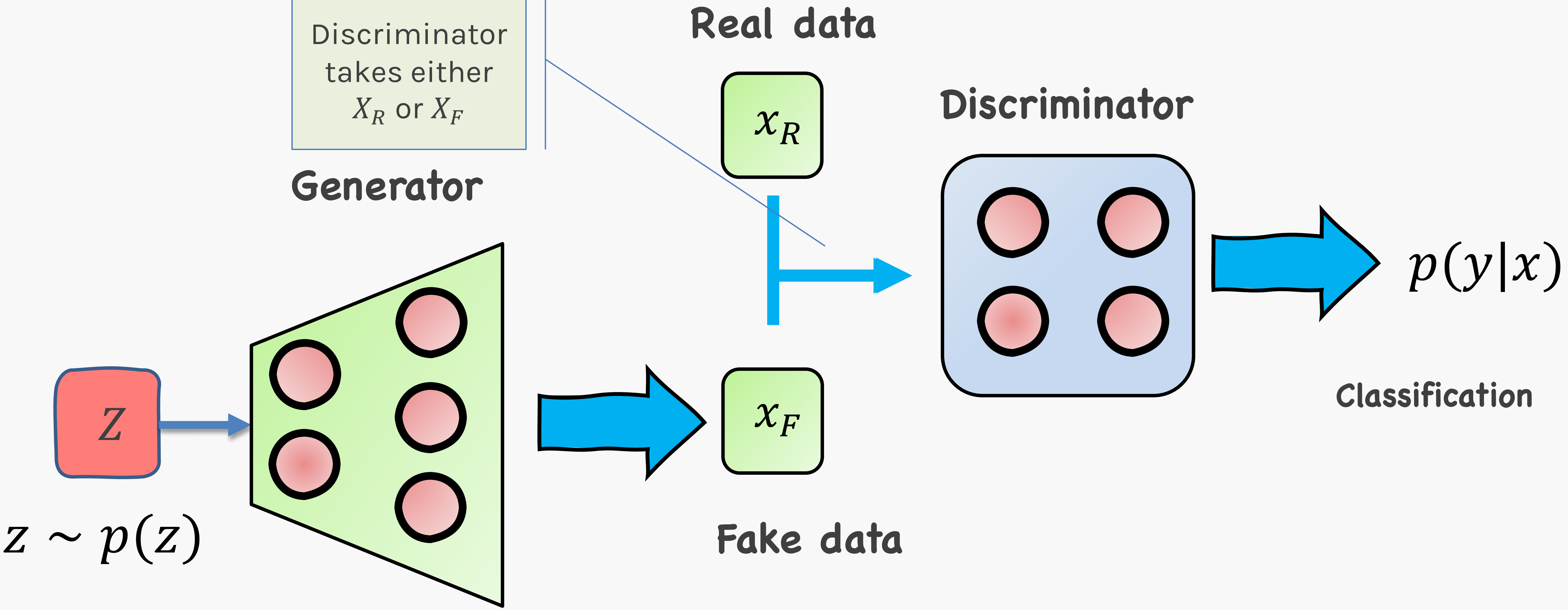
If we build our generator to be deterministic, then the same input will always produce the same output.

We want to generate data from a distribution. In that sense, we can think of the input values as latent variables.



GANS Architecture

Discriminator takes either X_R or X_F



Training: Loss Function

In a binary classification problem, the loss function is given by:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log\big(p(y_i)\big) + (1 - y_i)\log\big(1 - p(y_i)\big)$$

Where y is the label for $real$=1 or $fake$=0. The input to the **Discriminator** can be the real data or the fake data generated by the **Generator**. Splitting the loss function, we have:

$$\mathcal{L} = -\frac{1}{N_{Real}}\sum_{i=1}^{N_{Real}} y_i \log\big(p(y_i)\big) + (1 - y_i)\log\big(p(1 - y_i)\big) -$$
$$\frac{1}{N_{Fake}}\sum_{i=1}^{N_{Fake}} y_i \log\big(p(y_i)\big) + (1 - y_i)\log\big(1 - p(y_i)\big)$$

Learning

Real: $y_i = 1$

$$\mathcal{L} = -\frac{1}{N_{Real}}\sum_{i=1}^{N_{Real}} y_i \log\left(p(y_i)\right) + \cancel{(1 - y_i)\log\left(1 - p(y_i)\right)}$$

$$-\frac{1}{N_{Fake}}\sum_{i=1}^{N_{Fake}} \cancel{y_i\log\left(p(y_i)\right)} + (1 - y_i)\log\left(1 - p(y_i)\right)$$

Fake: $y_i = 0$

Learning

$$\mathcal{L} = -\frac{1}{N_{Real}}\sum_{i=1}^{N_{Real}} \log\left(p(y_i)\right) - \frac{1}{N_{Fake}}\sum_{i=1}^{N_{Fake}} \log\left(1 - p(y_i)\right)$$

Learning

$$\mathcal{L} = -\frac{1}{N_{Real}}\sum_{i=1}^{N_{Real}} \log\big(p(y_i)\big) - \frac{1}{N_{Fake}}\sum_{i=1}^{N_{Fake}} \log\big(1 - p(y_i)\big)$$

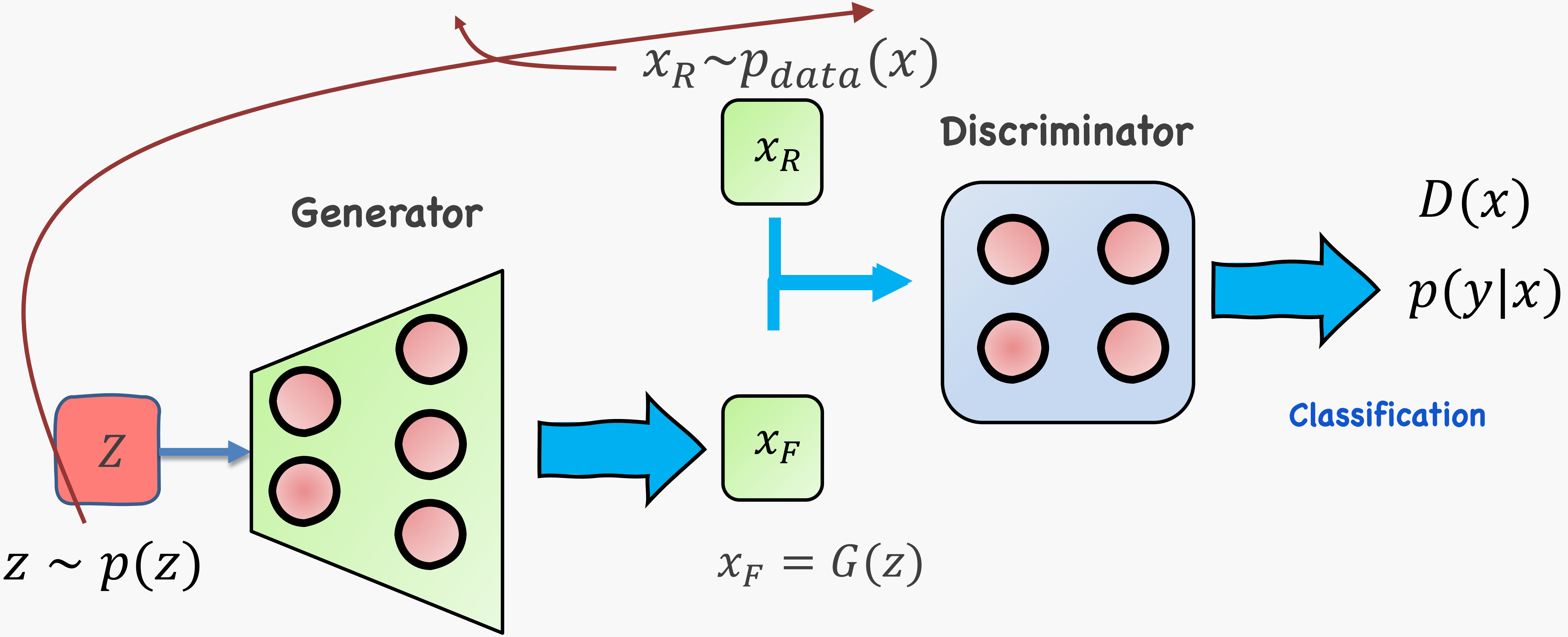
Rewriting in terms of discriminator **D** and generator **G** outputs:

$$\mathcal{L} = -\frac{1}{N_{Real}}\sum_{i=1}^{N_{Real}} \log\left(D\left(x_i^R\right)\right) - \frac{1}{N_{Fake}}\sum_{i=1}^{N_{Fake}} \log\big(1 - D(x_i^F)\big)$$

And noting that $x_i^F = G(z_i)$

$$\mathcal{L} = -\frac{1}{N_{Real}}\sum_{i=1}^{N_{Real}} \log\left(D\left(x_i^R\right)\right) - \frac{1}{N_{Fake}}\sum_{i=1}^{N_{Fake}} \log(1 - D(G(z_i)))$$

$$\mathcal{L} = -\frac{1}{N_{Real}}\sum_{i=1}^{N_{Real}} \log\left(D\left(x_i^R\right)\right) - \frac{1}{N_{Fake}}\sum_{i=1}^{N_{Fake}} \log(1 - D(G(z_i)))$$



$$\mathcal{L} = -\mathrm{E}_{x\sim p_{data}(x)}[\log(D(x))] - \mathrm{E}_{z\sim p_z(z)}[\log(1 - D(G(z))]$$

Learning

$$\mathcal{L} = -\mathrm{E}_{x\sim p_{data}(x)}[\log(D(x))] - \mathrm{E}_{z\sim p_z(z)}[\log(1 - D(G(z))]$$

The adversarial training can be described as though the **Generator G** and **Discriminator D** play the following two-player min-max game with the following value function V (G, D).

The **Discriminator's** job is to minimize the loss or maximize the –ve loss.

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

The **Generator's job is to maximize the loss or minimize the –ve loss.**

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

Minimax value function

Discriminator's prediction on real data

Discriminator's prediction on fake data

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

Sample real data

Sample random noise

Generator's output: fake data

Minimax value function

Discriminator: Maximize

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

Generator: Minimize

Minimax value function

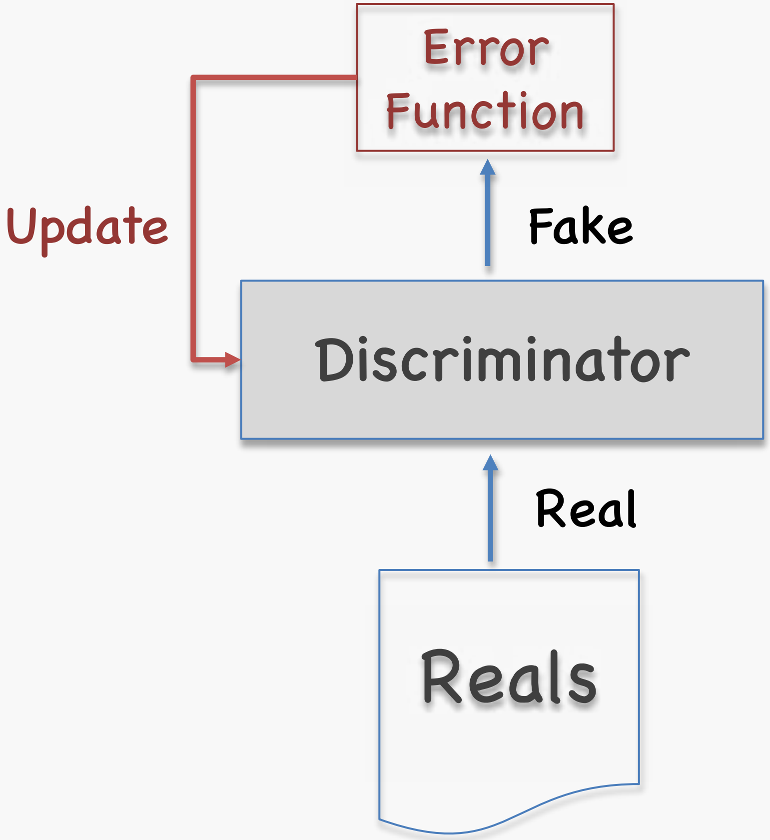
Discriminator: Maximize

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

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

Generator: Maximize

Training the GAN



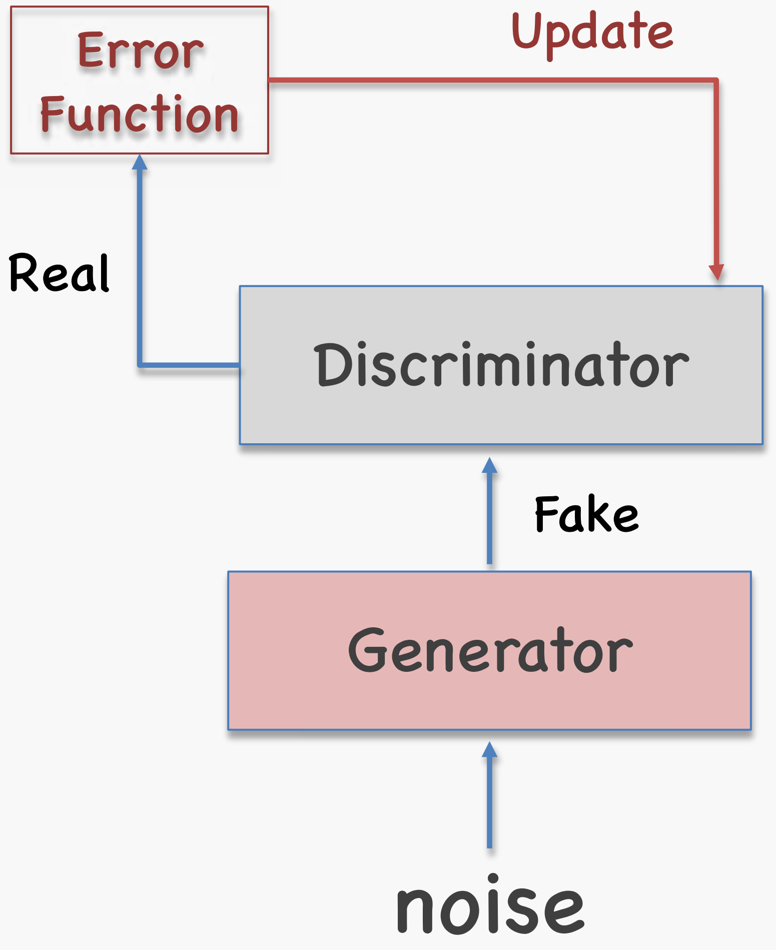
$$\max_D \mathrm{E}_{x \sim p_{data}(x)}[\log(D(x))]$$

False negative (I: Real/D: Fake):

In this case we feed reals to the discriminator. The Generator is not involved in this step at all.

The error function here only involves the Discriminator and if it makes a mistake the error drives a backpropagation step through the discriminator, updating its weights, so that it will get better at recognizing reals.

Training the GAN

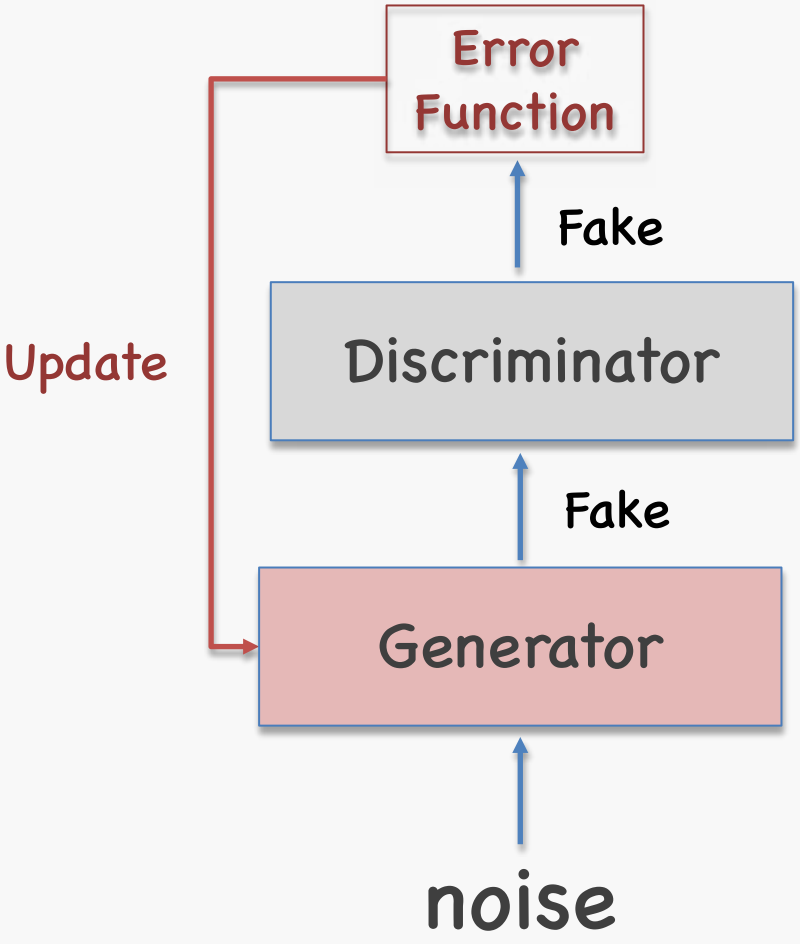


$$\max_{D} \mathrm{E}_{z \sim p_z(z)}[\log(1 - D(G(z))]$$

False positives (I:Fake/D:Real):

Here we generate a fake and punish the discriminator if it classifies it as real.

Training the GAN



$$\max_G \mathrm{E}_{z\sim p_z(z)}[\log(\, D(G(z))]$$

True negative (I: Fake/D: Fake):

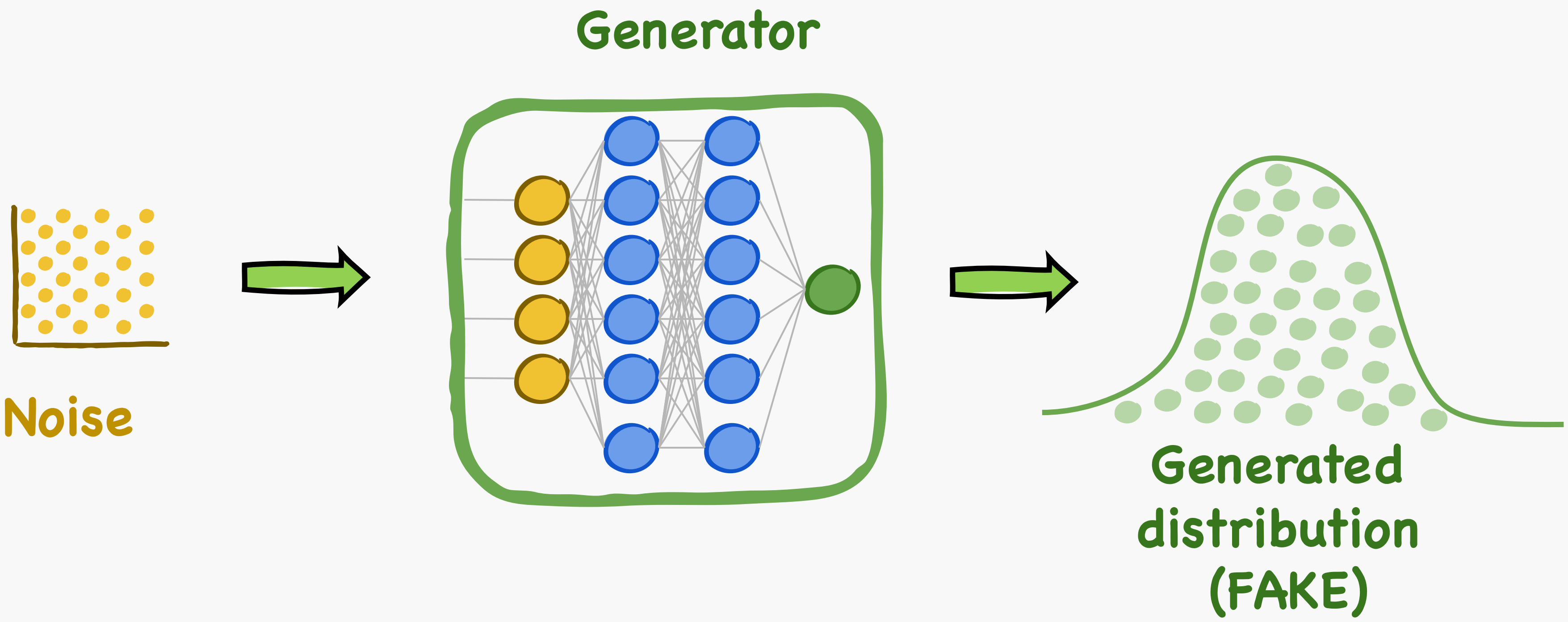
- We start with random numbers going into the generator.
- The generator's output is a fake.
- The objective function gets a large –ve value if this fake is correctly identified as fake. Meaning that the generator got caught.
- Backprop, goes through the discriminator (which is frozen) to the generator updating the generator's weight, so it can better learn how to fool the discriminator.

Learning

The process – known as **Learning Round** - accomplishes three jobs:

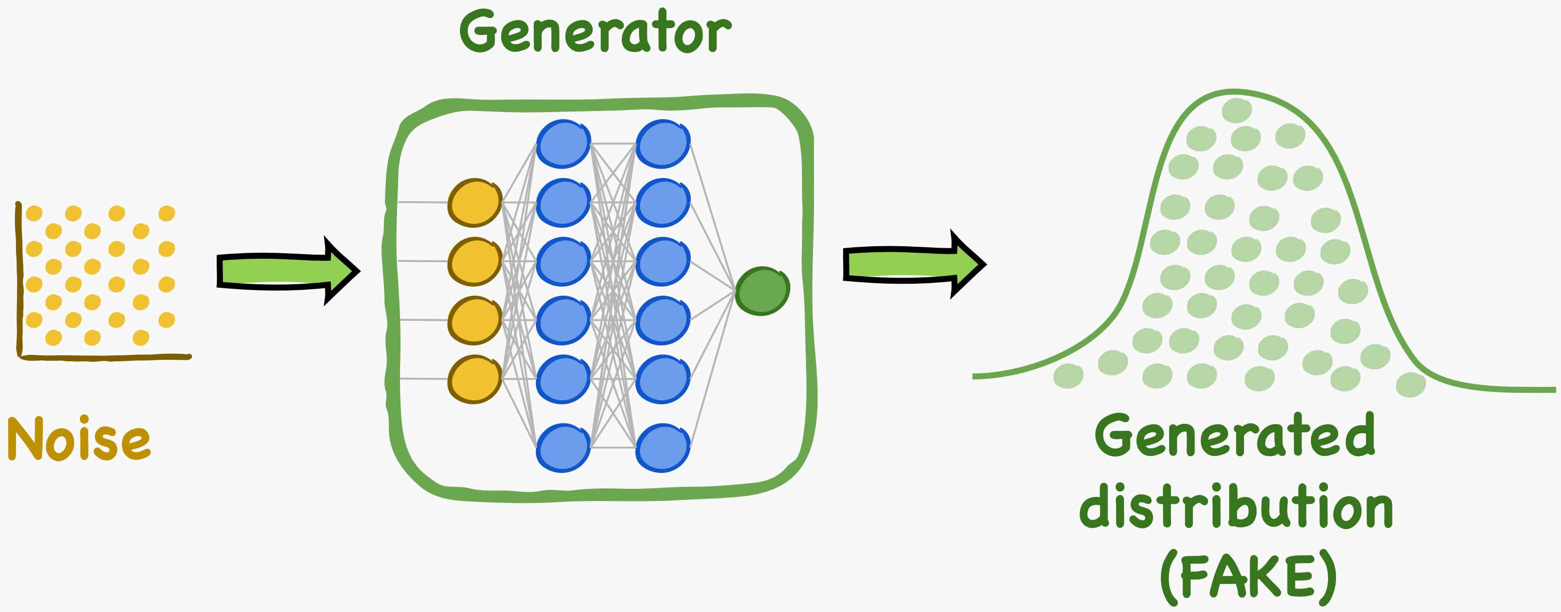
1. The discriminator learns to identify features that characterize a real sample
2. The discriminator learns to identify features that reveal a fake sample
3. The generator learns how to avoid including the features that the discriminator has learned to spot

The Generator



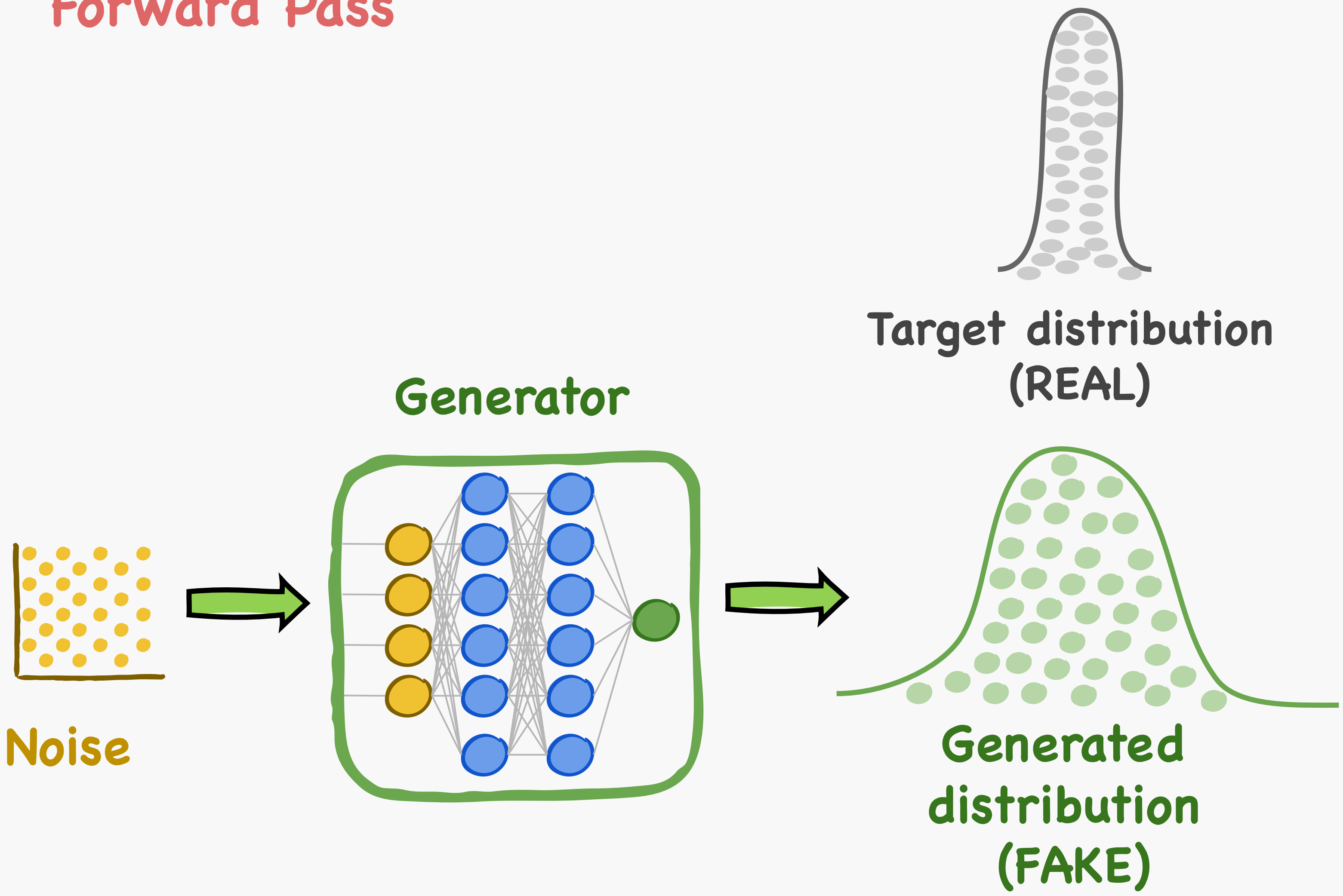
Training GANs

Forward Pass

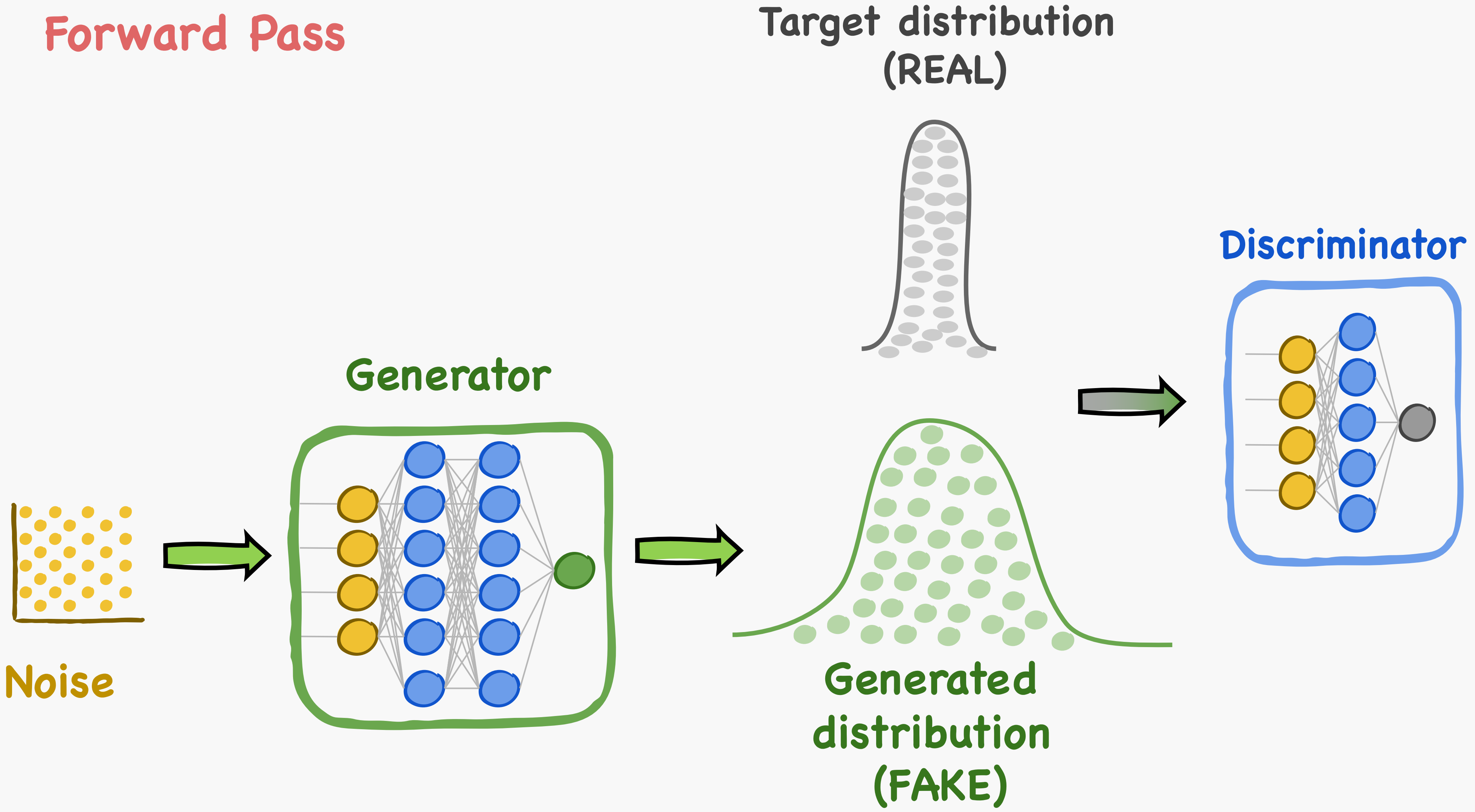


Training GANs

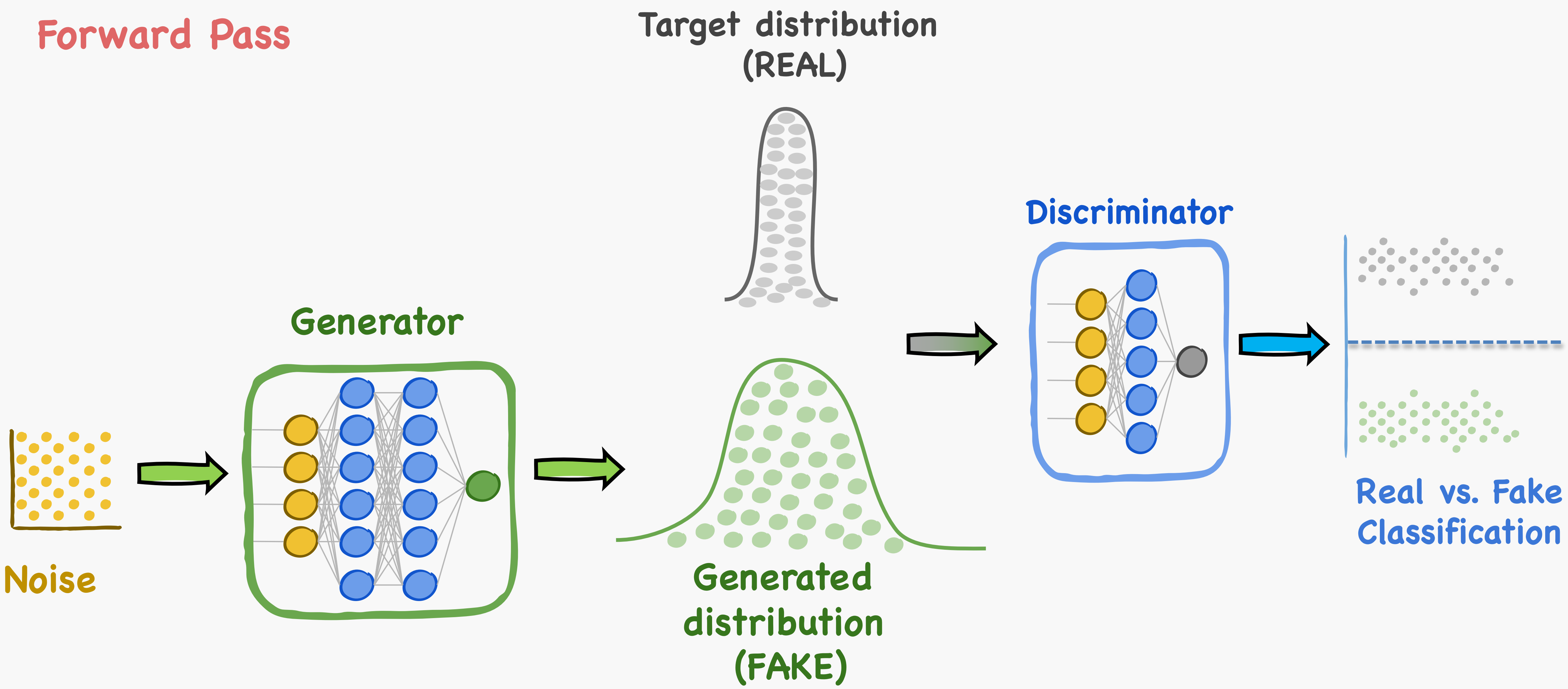
Forward Pass



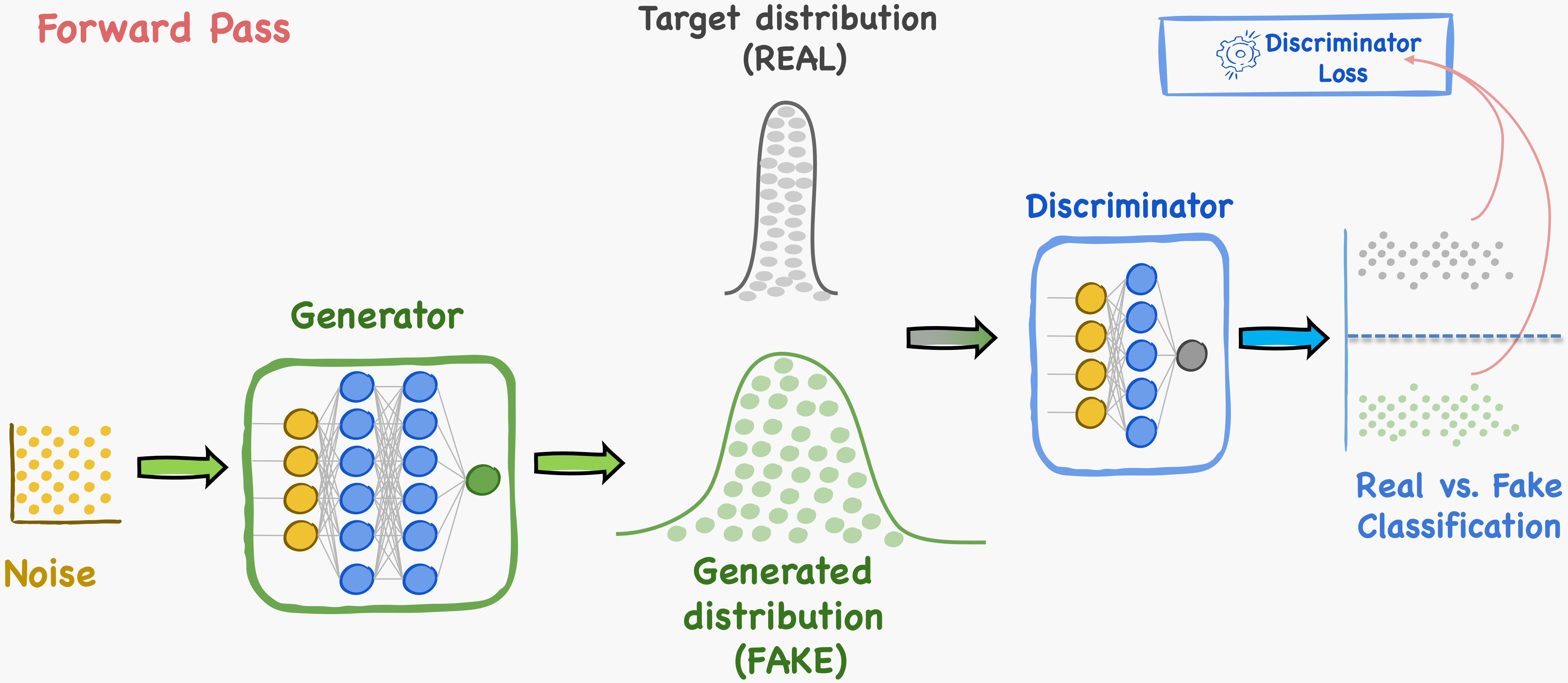
Training GANs



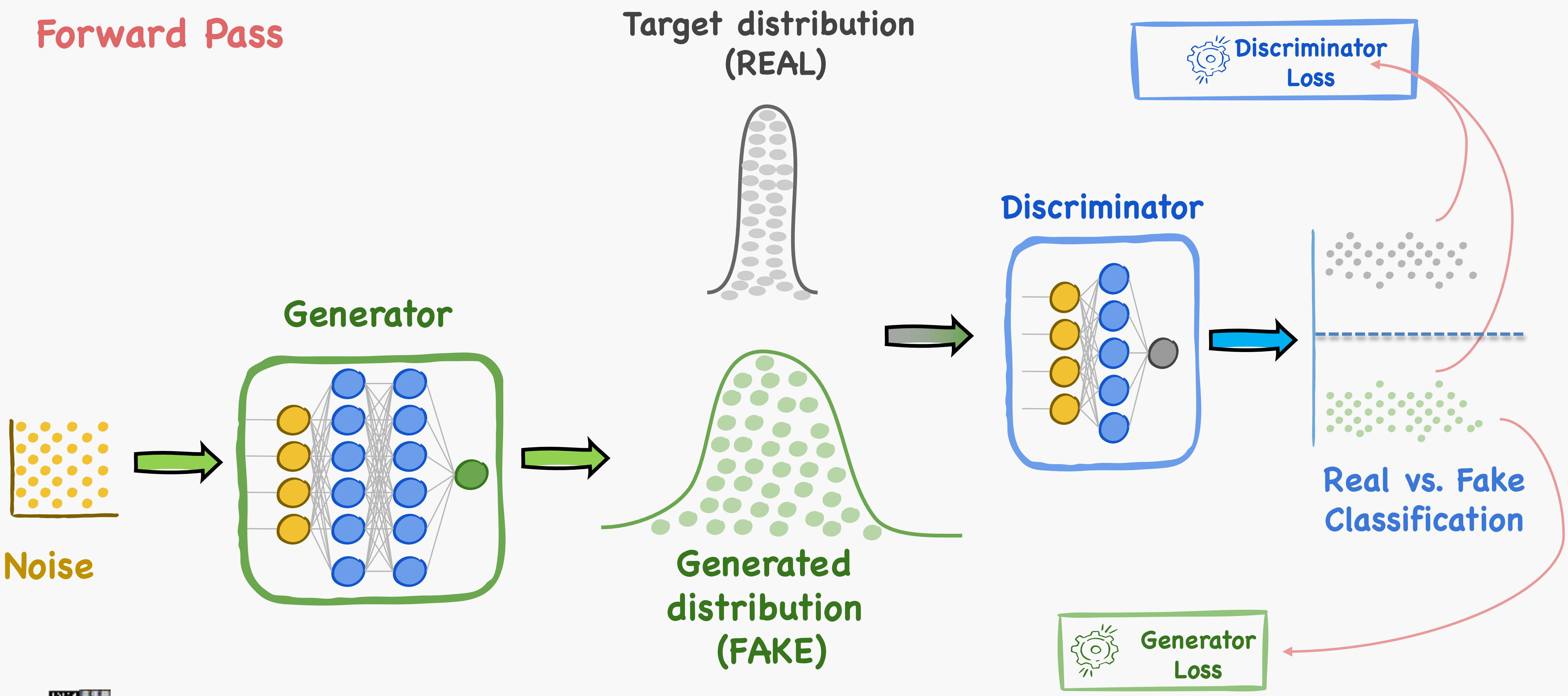
Training GANs



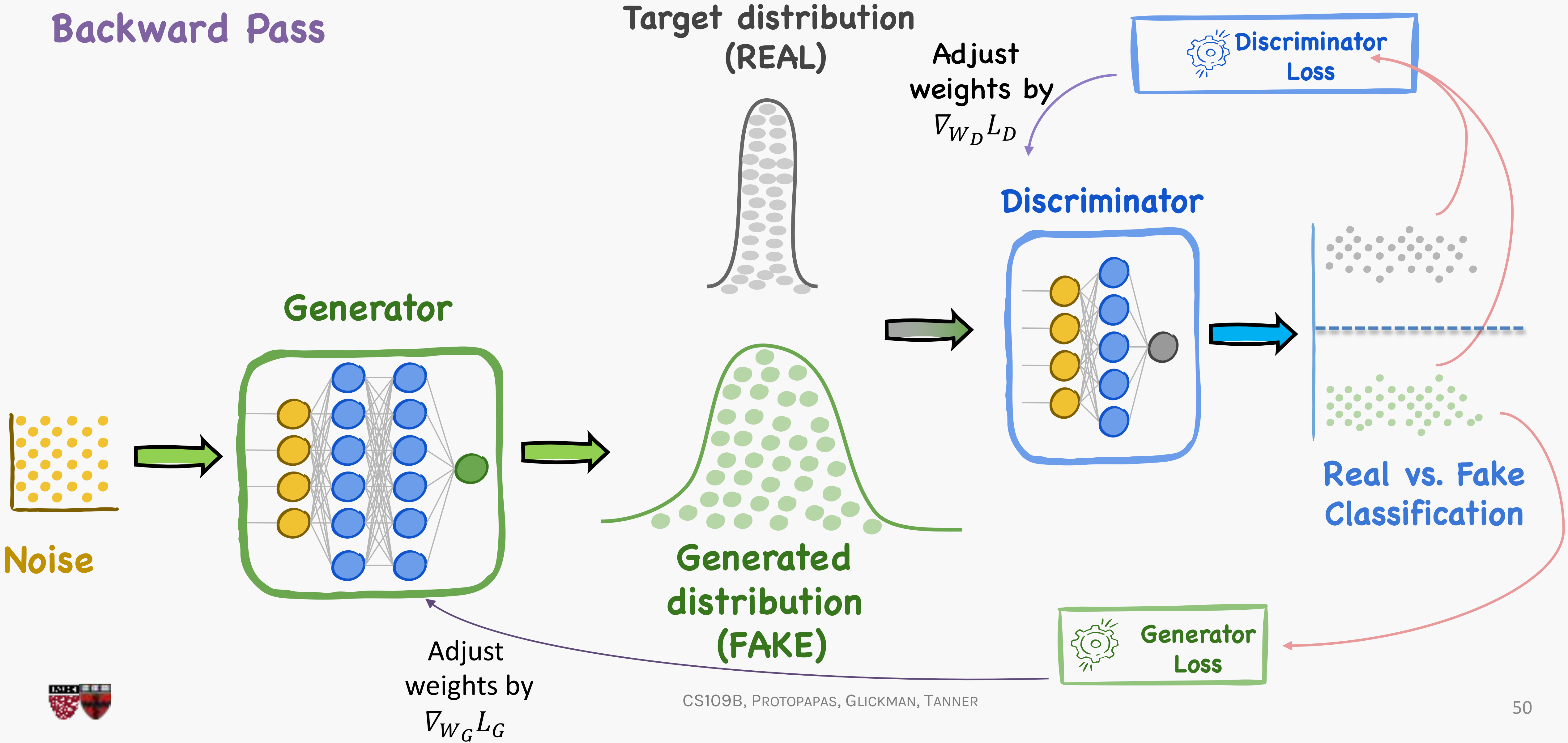
Training GANs



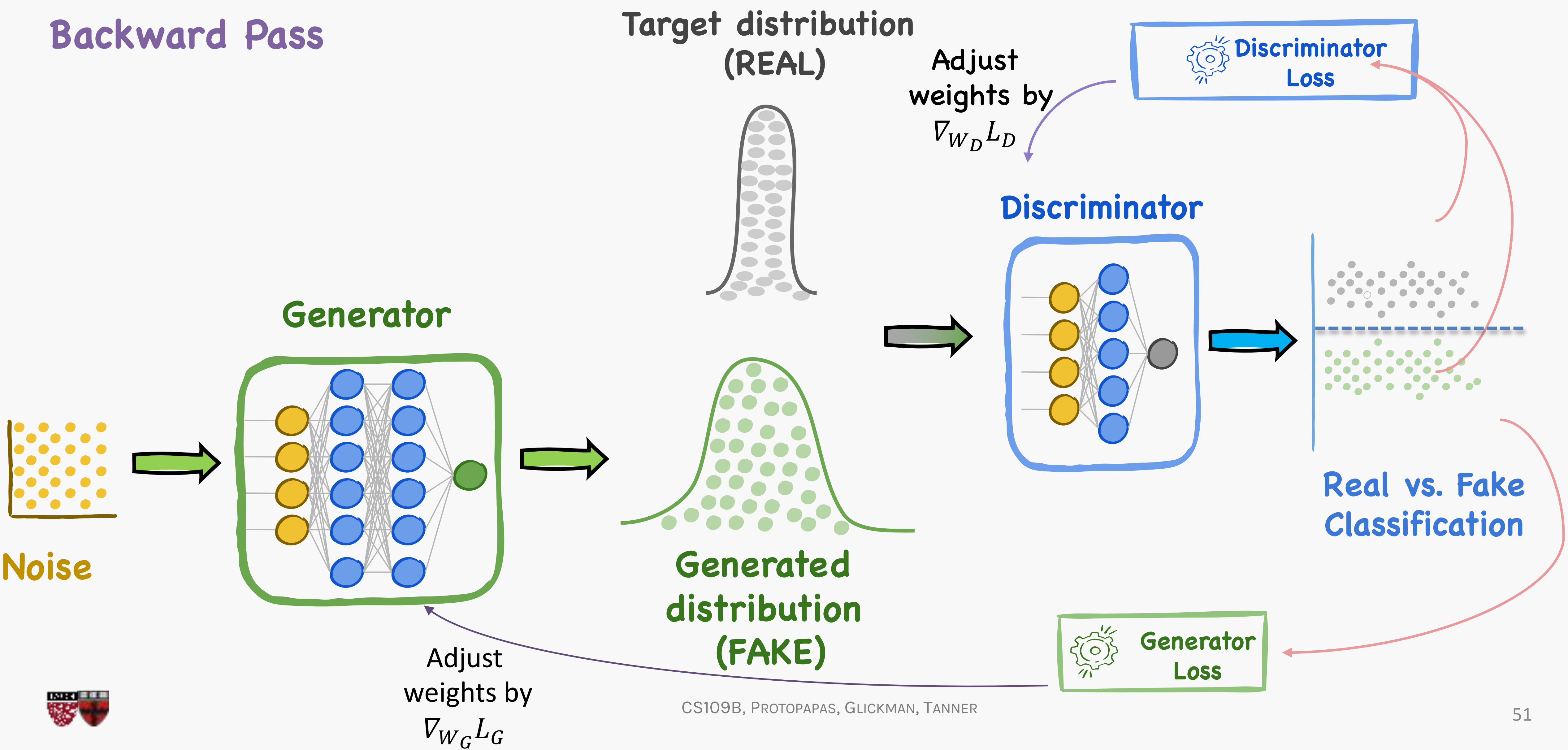
Training GANs



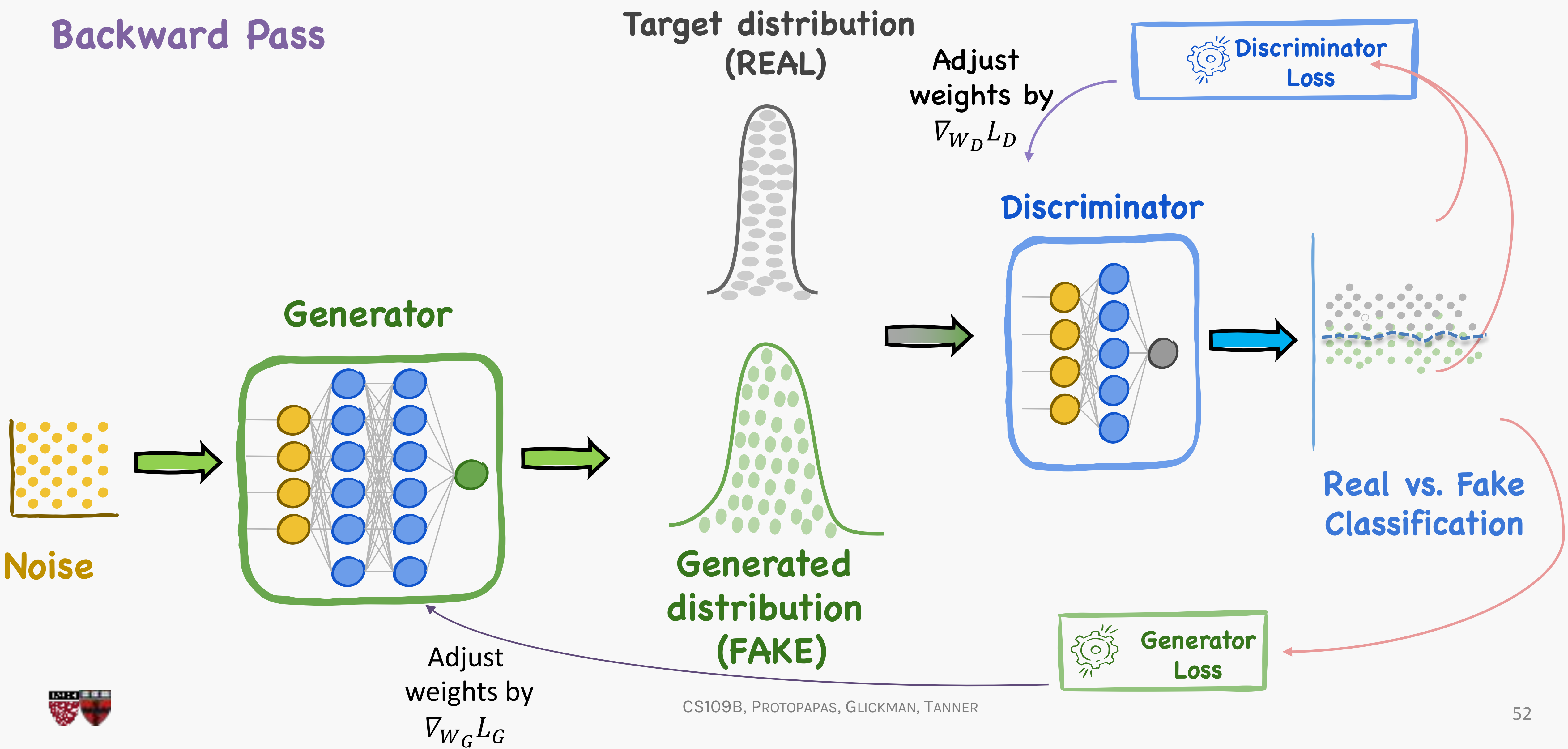
Training GANs



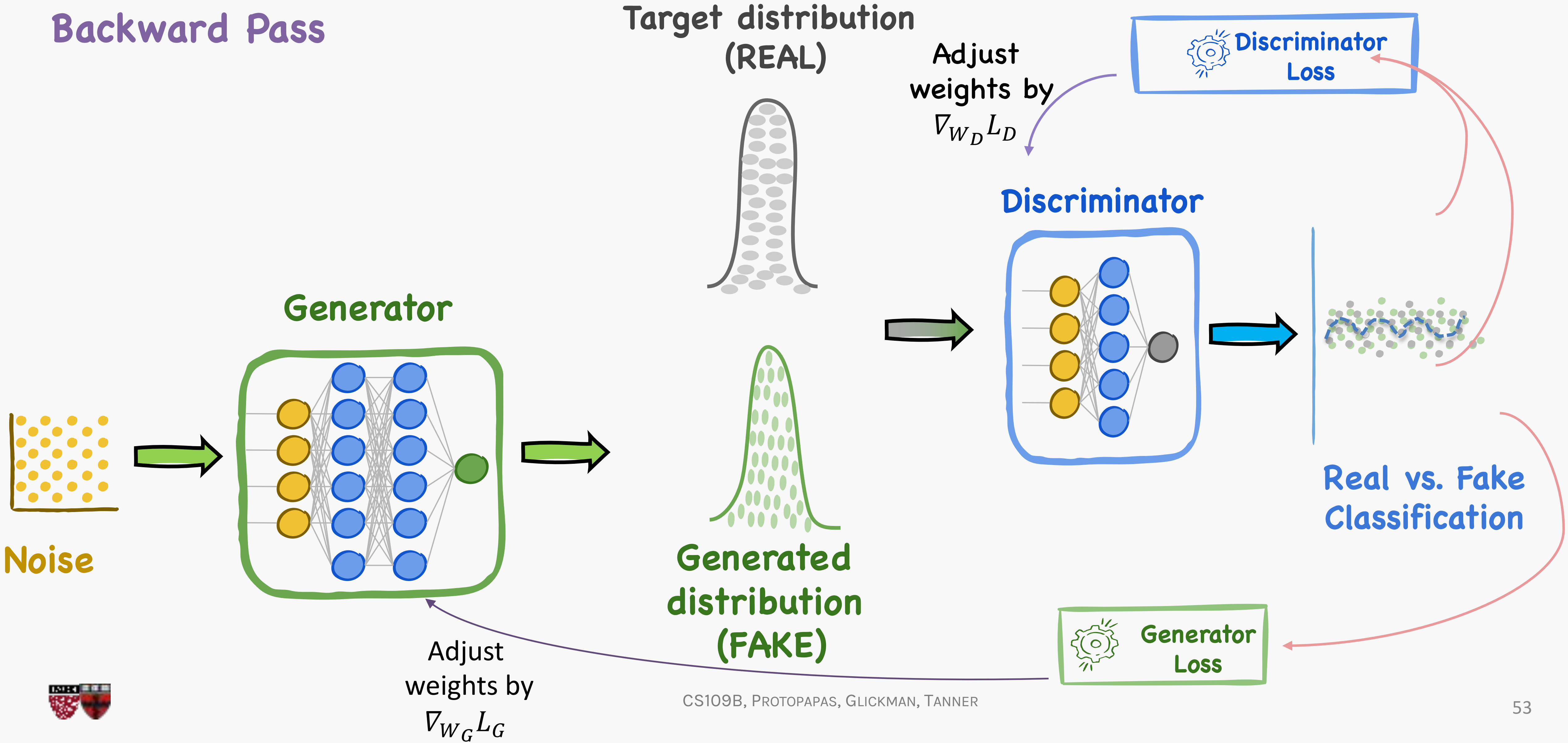
Training GANs



Training GANs



Training GANs



Generative Adversarial Networks

Training procedure

for number of training iterations **do**

  for k steps **do**

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

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

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

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

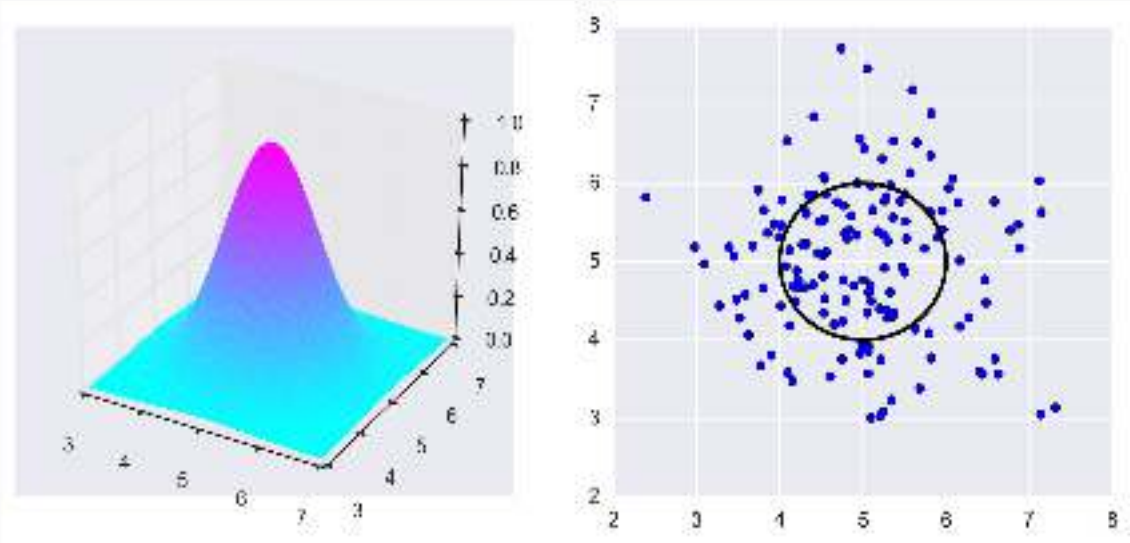
end for

Building GANS: Fully Connected Case

4 noise → [16 neurons ReLU Dropout 0.1] [16 neurons Linear] → [16 neurons ReLU Dropout 0.1] [16 neurons ReLU Dropout 0.1] [1 neurons Sigmoid] → 1

Let's build a FC simple GAN to generate points from a 2-dimensional Gaussian Distribution.

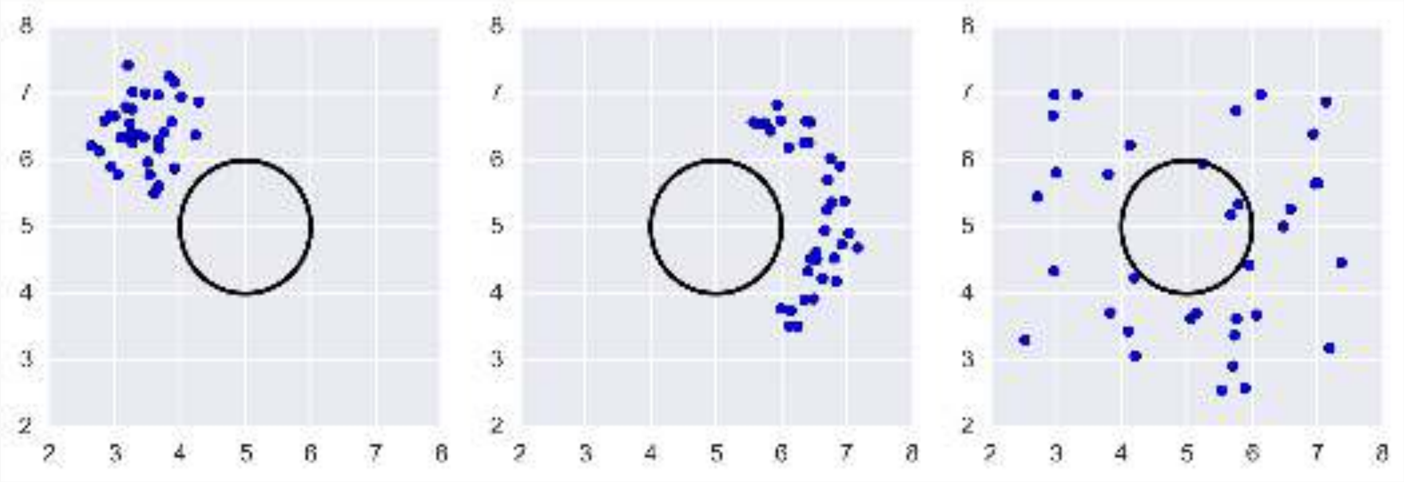
- **Generator**
  - Takes 4 random numbers
  - Generates a coordinate pair
- **Discriminator**
  - Takes an input point in the form of a coordinate pair
  - Determines whether the point is drawn from a specific 2-D Gaussian



Building GANS: Fully Connected Case

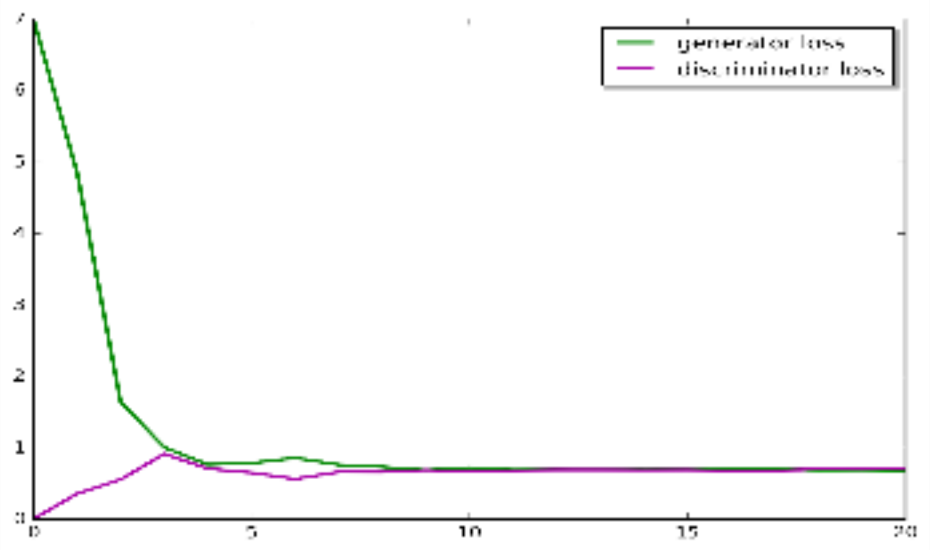
Train the Networks based on their ability to generate/discriminate batches of points drawn from the distribution.

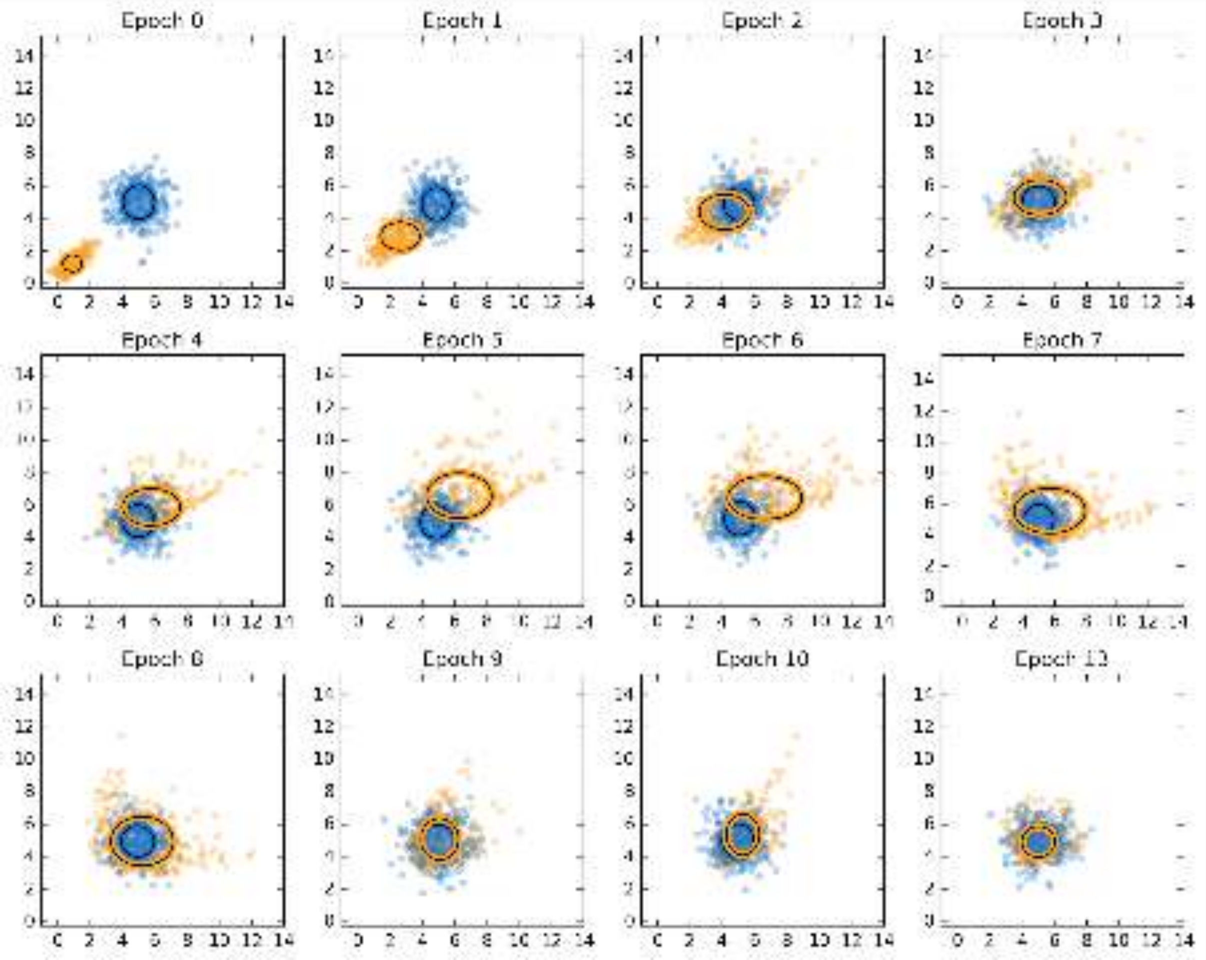
Are these batches of points drawn from the right distribution?



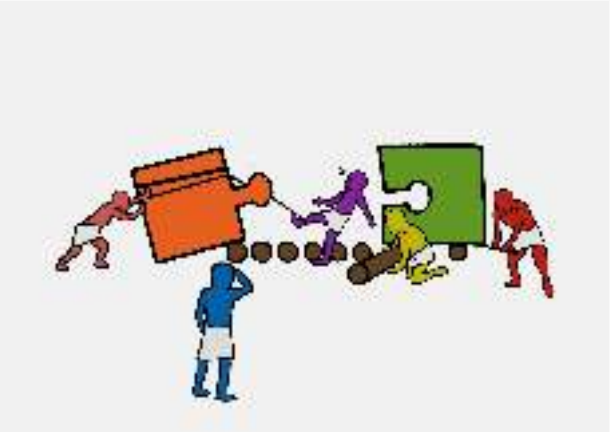
Building GANS: Fully Connected Case

As the generator and discriminator loss converges, the batch of points generated by the generator (in the yellow) approaches the real batch of points (in the blue)





Exercise:

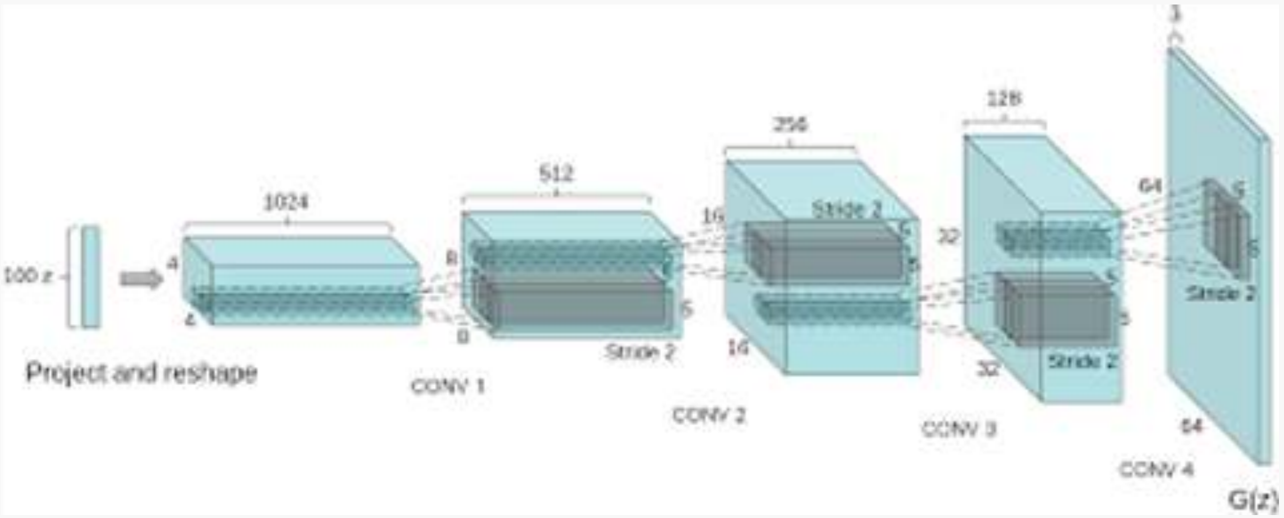


In this exercise, we are going to generate 1-D Gaussian distribution from a n-D uniform distribution. This is a toy exercise in order to understand the ability of GANs (generators) to generate arbitrary distributions from random noise.

Deep Convolutional GAN: DCGAN

Deep Convolutional GAN (**DCGAN**)
-- Alex Radford et al. 2016

- Eliminate fully connected layers.
- Max Pooling BAD! Replace all max pooling with convolutional stride
- Use transposed convolution for upsampling or simply upsampling.
- Use Batch normalization



Building GANS: DCGAN

DCGAN on MNIST

Generated digits

Evolution of GANs

5 Years of Improvement in Artificially Generated Faces

2021

https://twitter.com/goodfellow_ian/status/969776035649675265?lang=en

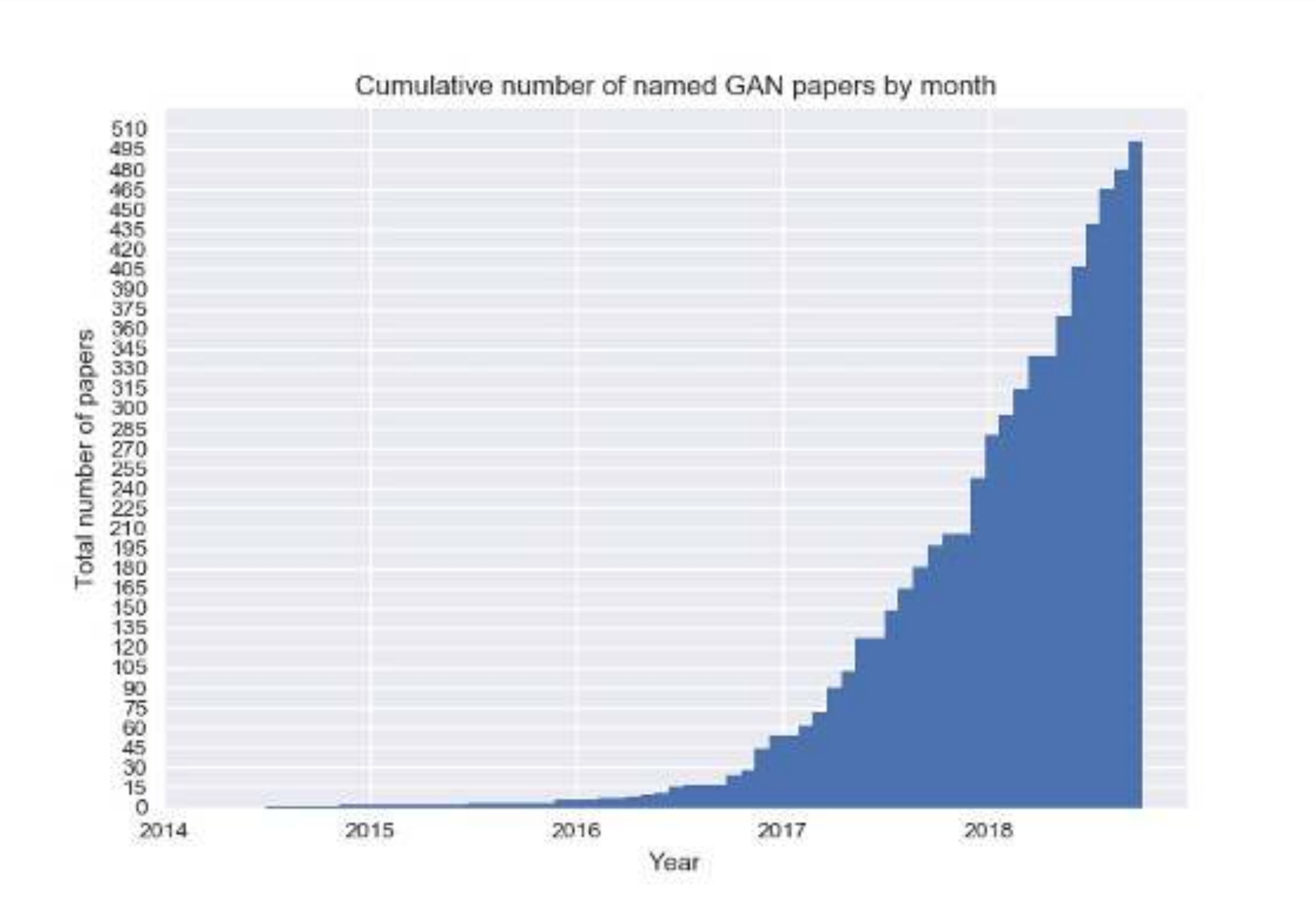
Ian Goodfellow
@goodfellow_ian

Follow

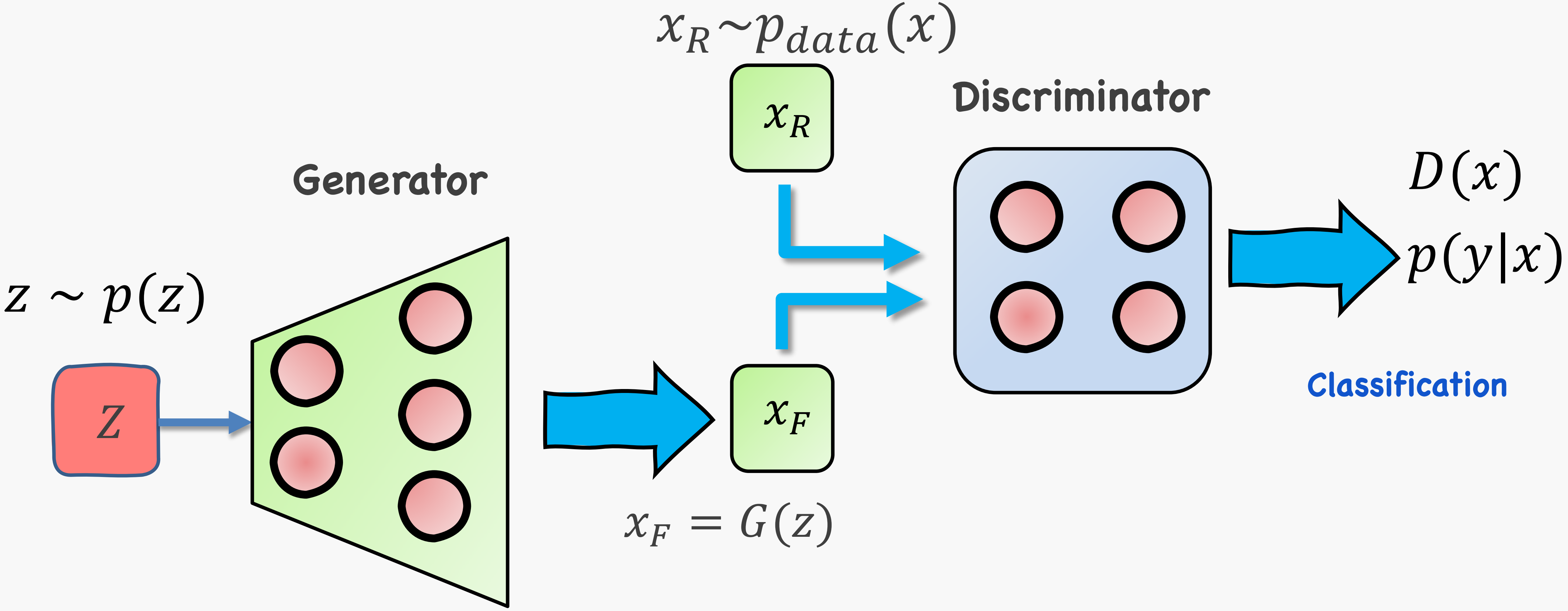
One of my favorite samples from the Progressive GANs paper is this one from the "cat" category. Apparently some of the cat training photos were memes with text. The GAN doesn't know what text is so it has made up new text-like imagery in the right place for a meme caption.

11:41 AM - 3 Dec 2017

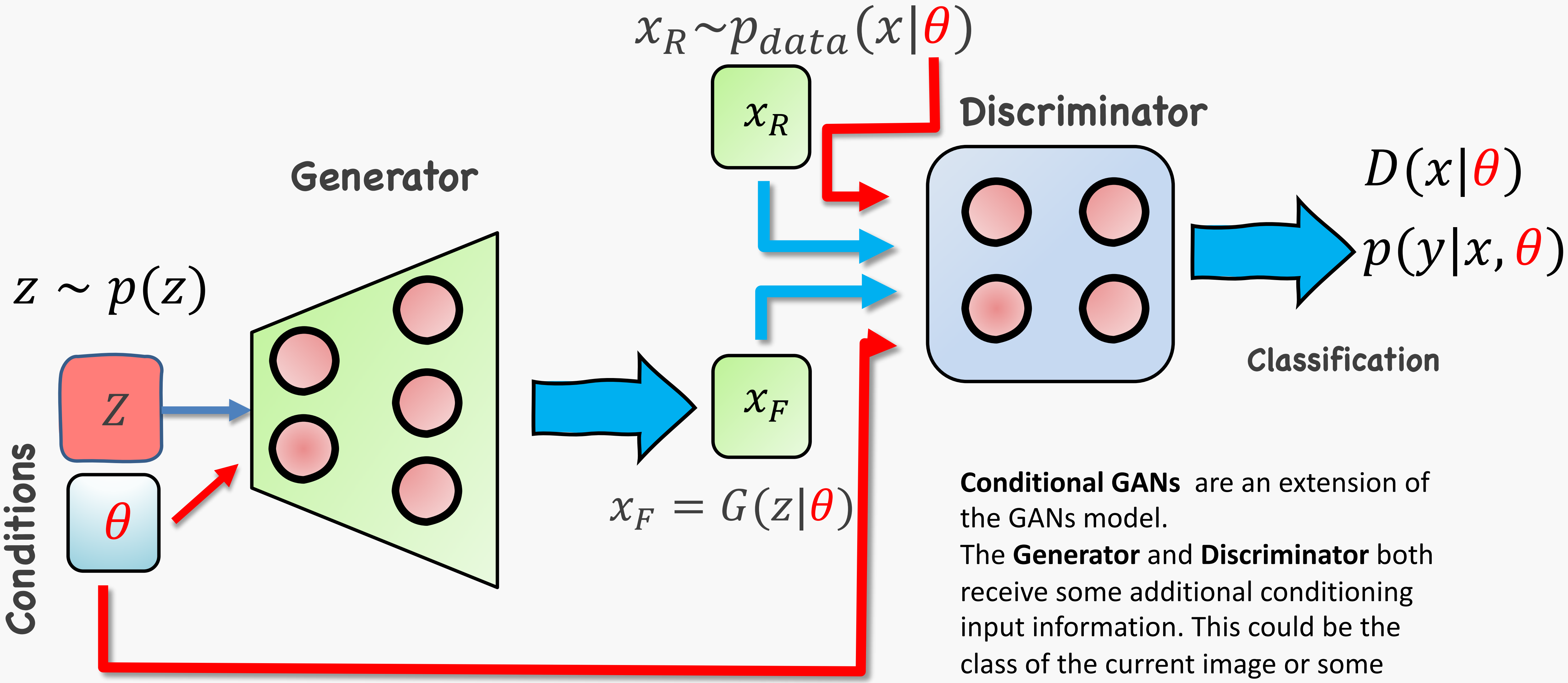
Evolution of GANs



Vanilla Generative Adversarial Nets



Conditional Generative Adversarial Nets



Conditional GANs are an extension of the GANs model.
The **Generator** and **Discriminator** both receive some additional conditioning input information. This could be the class of the current image or some other property.

https://arxiv.org/abs/1411.1784

Conditional Generative Adversarial Nets