Introduction to Generative Adversarial Networks GANs

Pavlos Protopapas

Photo: Pavlos
Lake Como

Outline

- Motivation for Generative Modeling
- High Level Formalism
- Architecture
- Mathematics
- Training GANS
- Deep Convolutional GANs

Outline

- **Motivation for Generative Modeling**
- High Level Formalism
- Architecture
- Mathematics
- Training GANS
- Deep Convolutional GANs

Which one is real?

A

B

They are both fake!

[Source]

Evolution of GANs

Over the last 8 years, we have been able to generate realistic artificial faces!

2014

2015

2016

2017

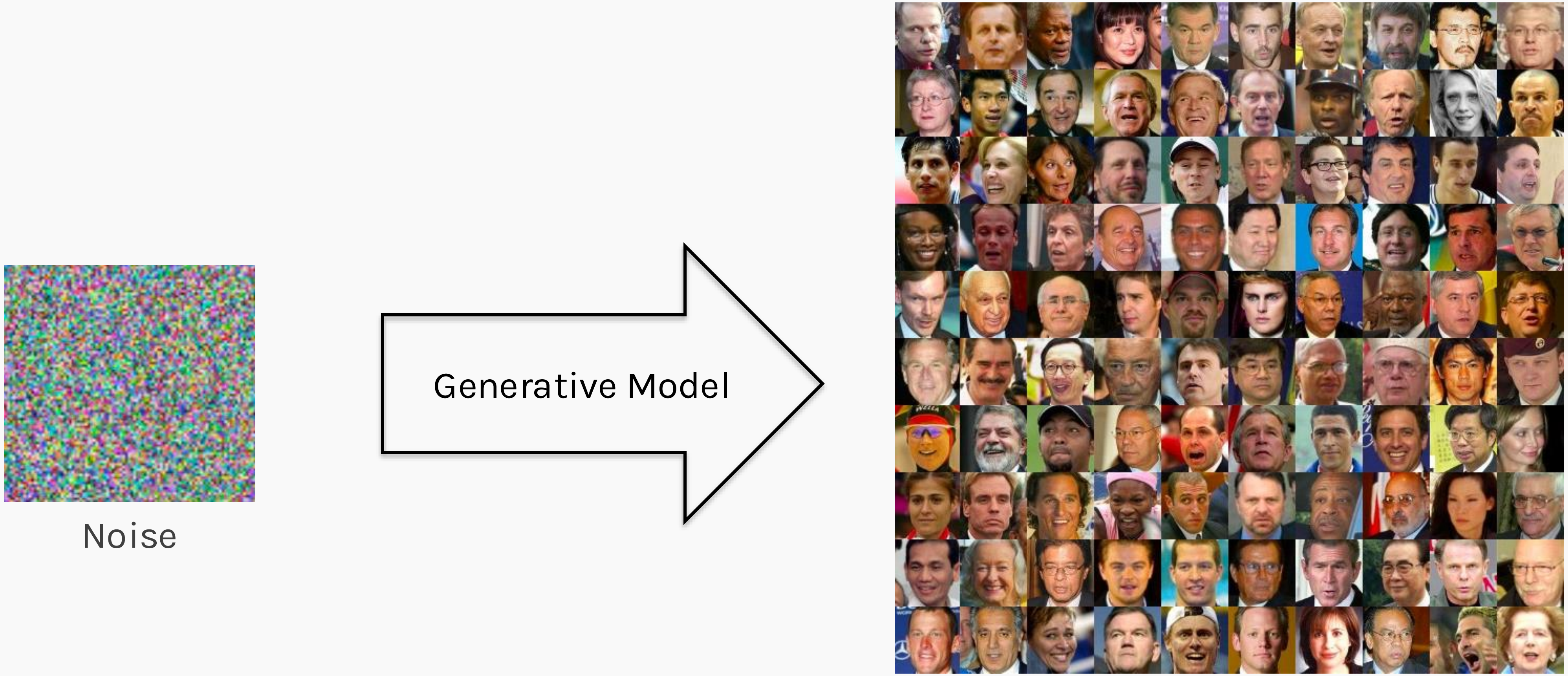
2018

2023

[Ian Goodfellow Twitter]

Generative Modeling

How was this done? Generative Modeling!



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 $\sim p_{model}(x)$

What is generative modeling?

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

Explicit sampling: Form an analytical expression for p_{model}(x).

- MCMC
- Variational methods
- Inverse transform sampling

Implicit sampling: Learn how to sample from p_{data}(x) without forming an analytical expression.

- Generative Adversarial Networks (GAN)
- Generator part of VAE

Why do we need generative modeling?

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

Homogeneous Skin Color, Pose

Diverse Skin Color, Pose and Illumination

[MIT 6.S191: Introduction to Deep Learning]

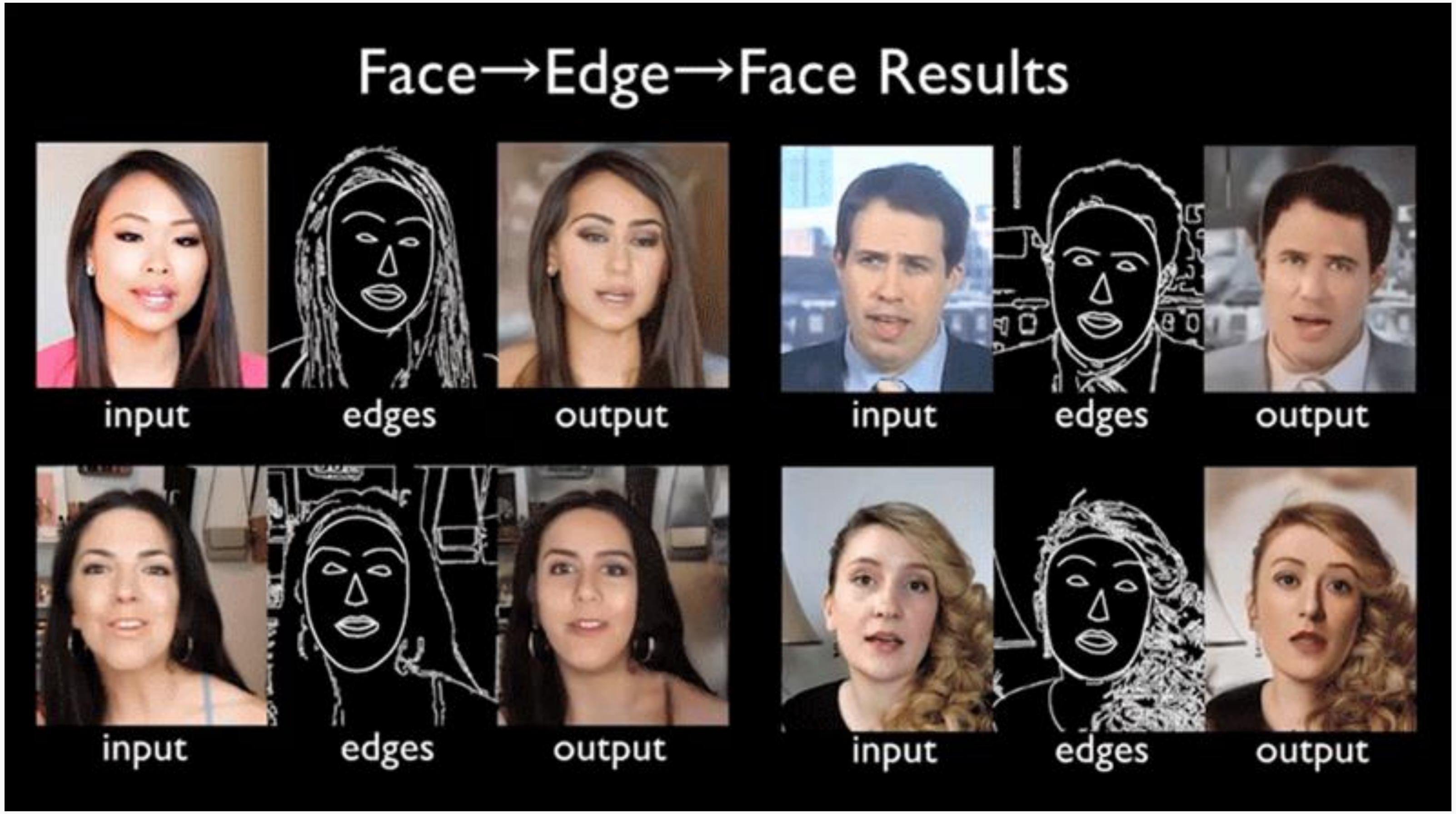
Some other applications of generative modeling

Unpaired Image-to-Image Translation using Cycle-GANs

[Zhu et al. 2017]

Some other applications of generative modeling

Video-to-Video Synthesis



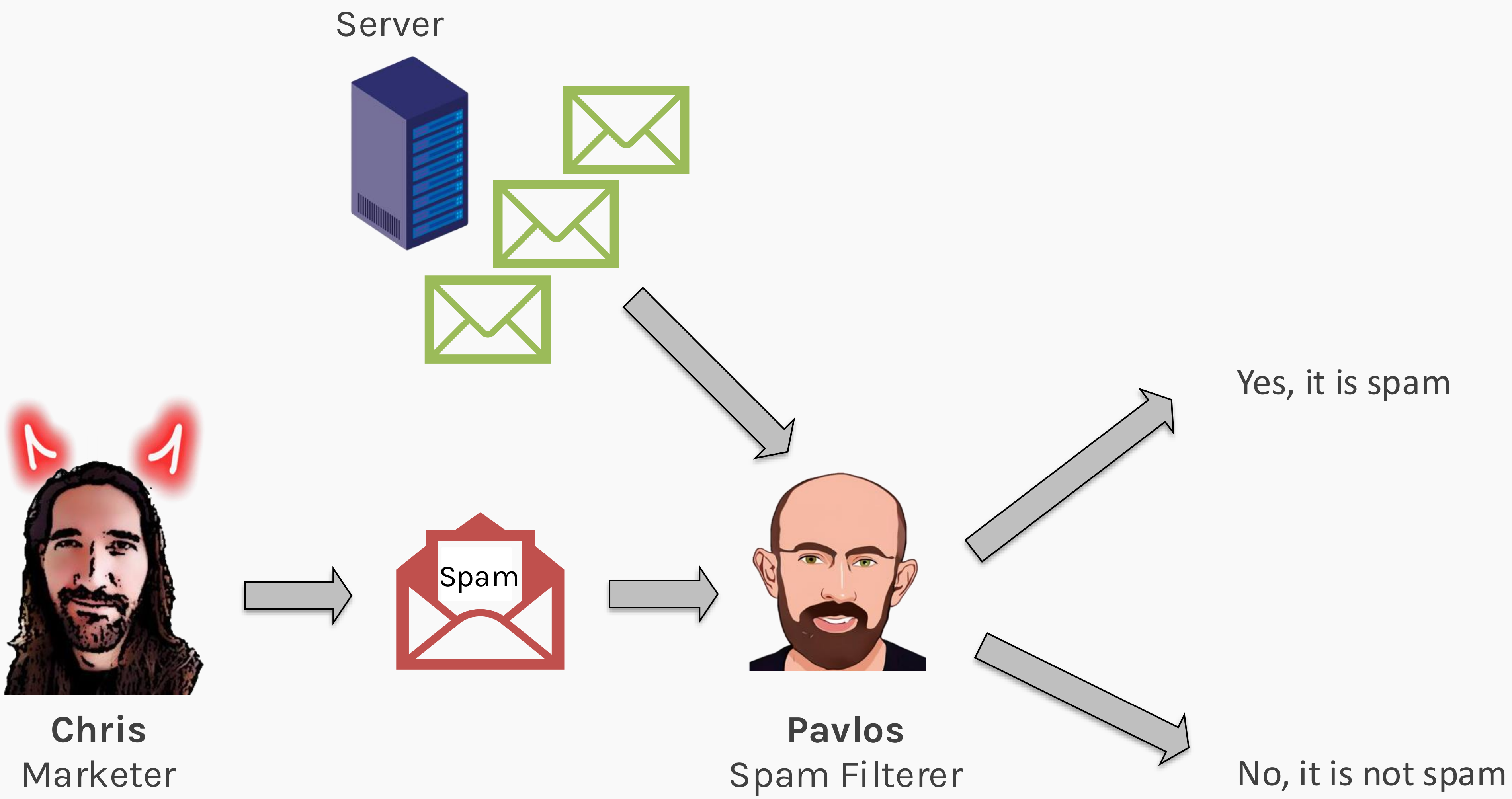
[Wang et al. 2018](#)


GANs
GANs EVERYWHERE

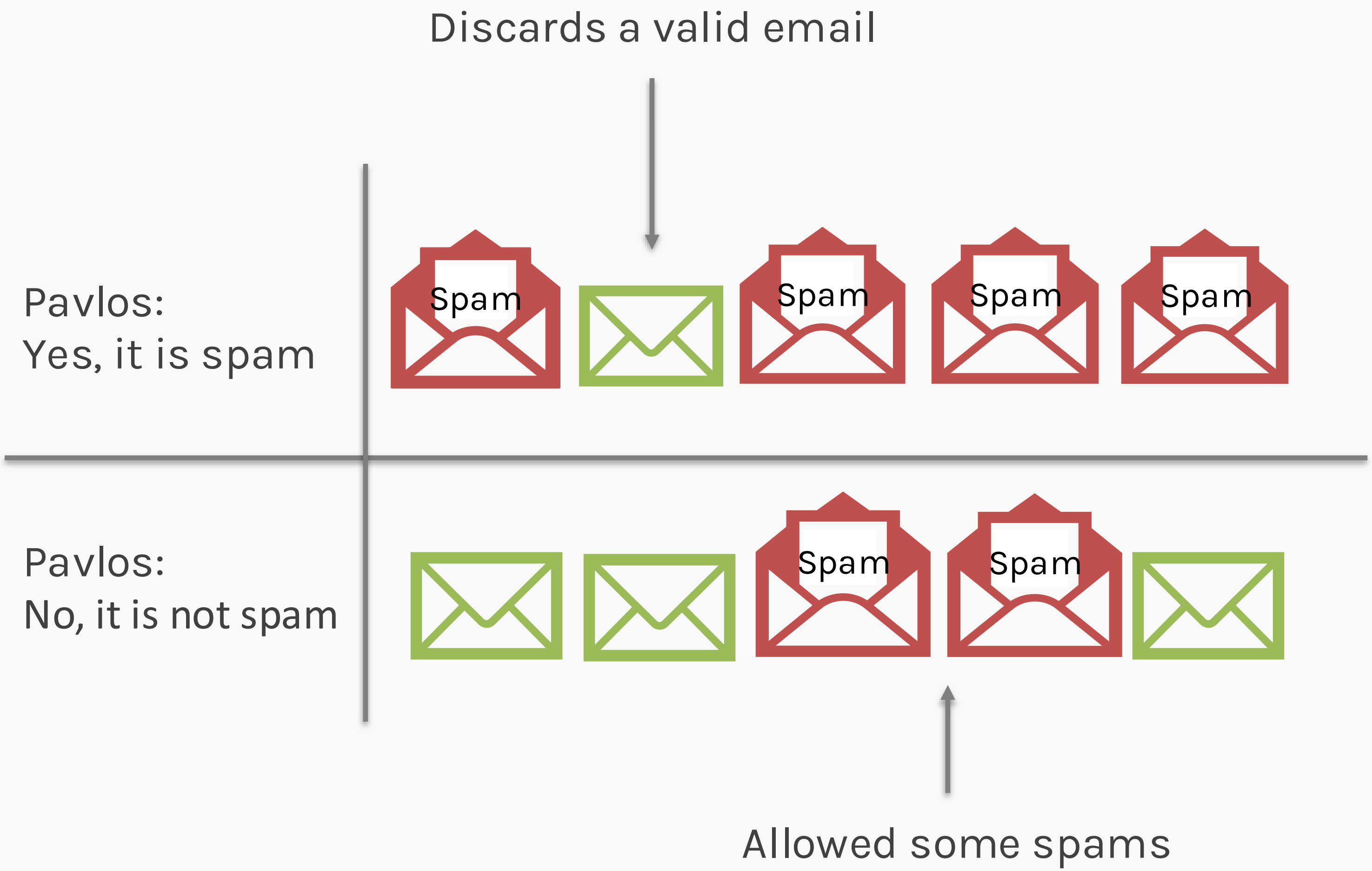
Outline

- Motivation for Generative Modeling
- **High Level Formalism**
- Architecture
- Mathematics
- Training GANS
- Deep Convolutional GANs

Generative Adversarial Networks (GANs)



Generative Adversarial Networks (GANs)

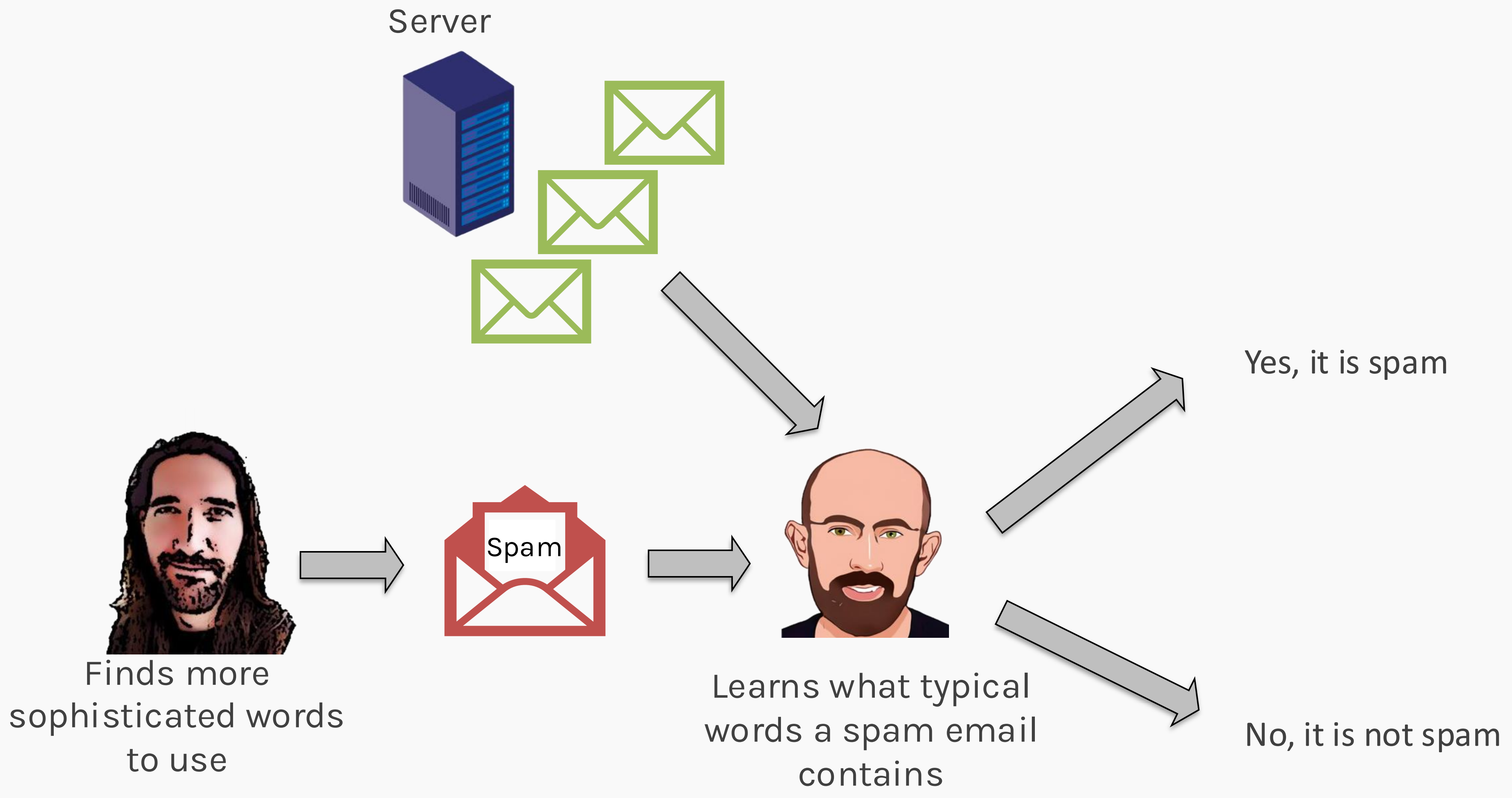


Generative Adversarial Networks (GANs)

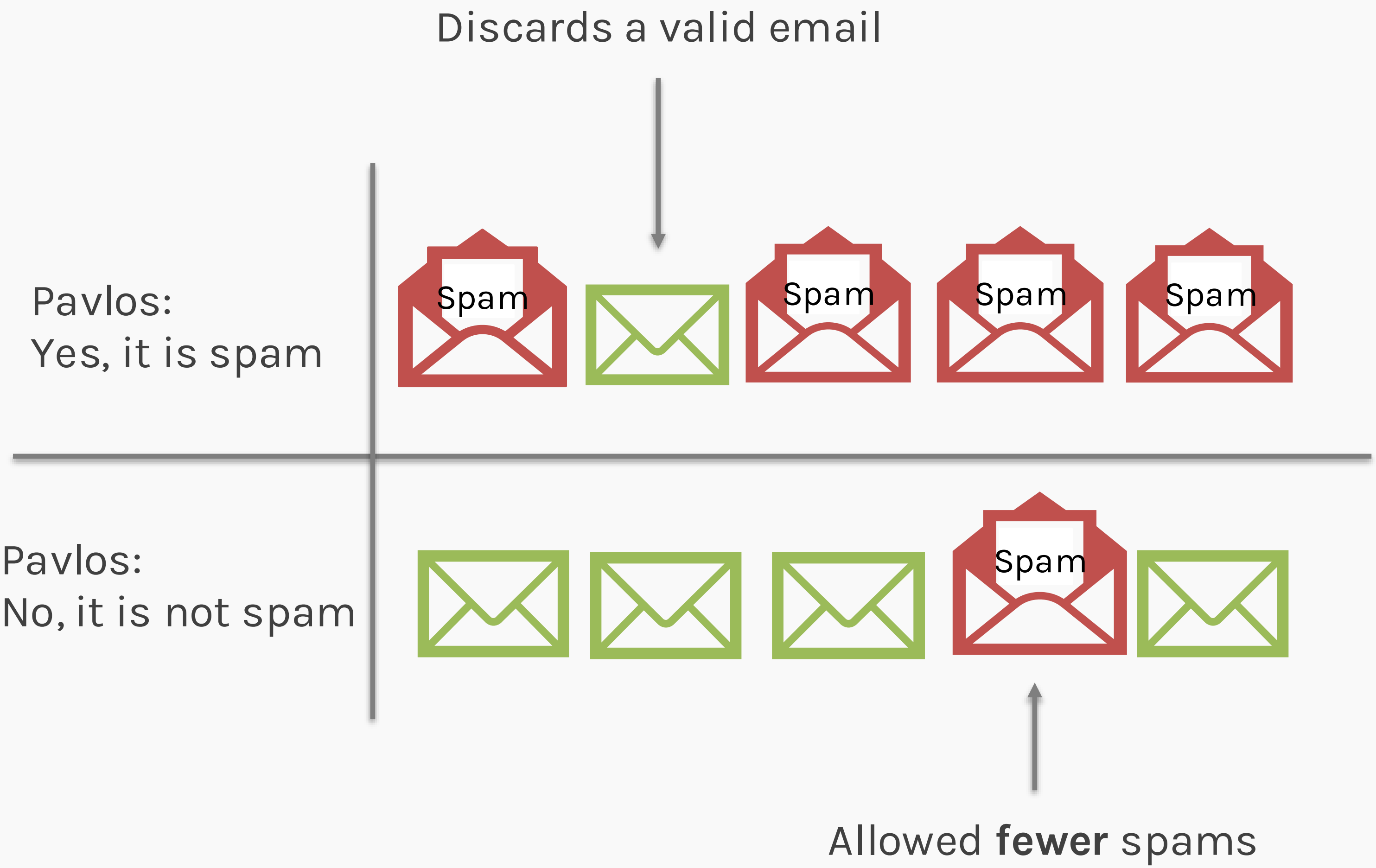
	It was actually spam	It was actually not spam
Pavlos: Yes, it is spam	Spam Spam Spam Spam	
Pavlos: No, it is not spam	Spam Spam	

Generative Adversarial Networks (GANs)

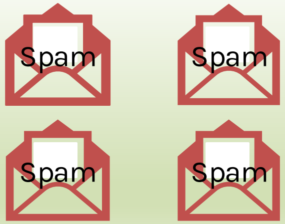
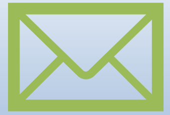
Chris and Pavlos **learn from what went wrong from their perspective**.



Generative Adversarial Networks (GANs)



Generative Adversarial Networks (GANs)

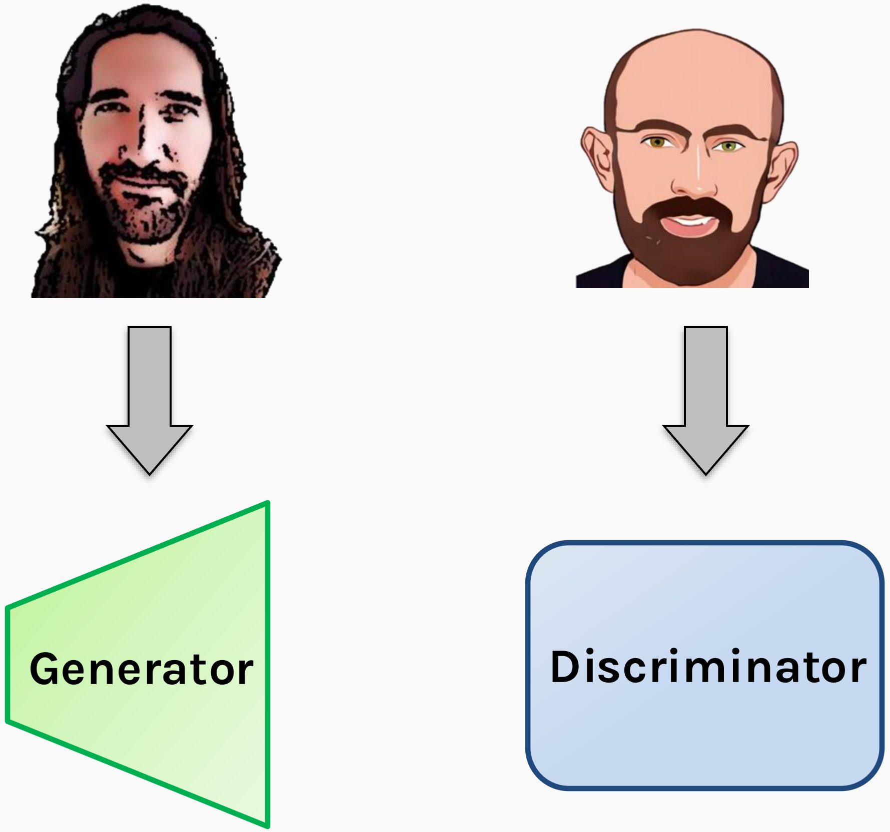
	It was actually spam	It was actually not spam
Pavlos: Yes, it is spam		
Pavlos: No, it is not spam		

Generative **Adversarial** Networks (GANs)

Adversaries: Chris and Pavlos

Becomes: Two player game between a **generator** G and a **discriminator** D.

The generator tries to fool the discriminator into thinking the spam email is real and the discriminator tries to get better at detecting if an email is spam or not.



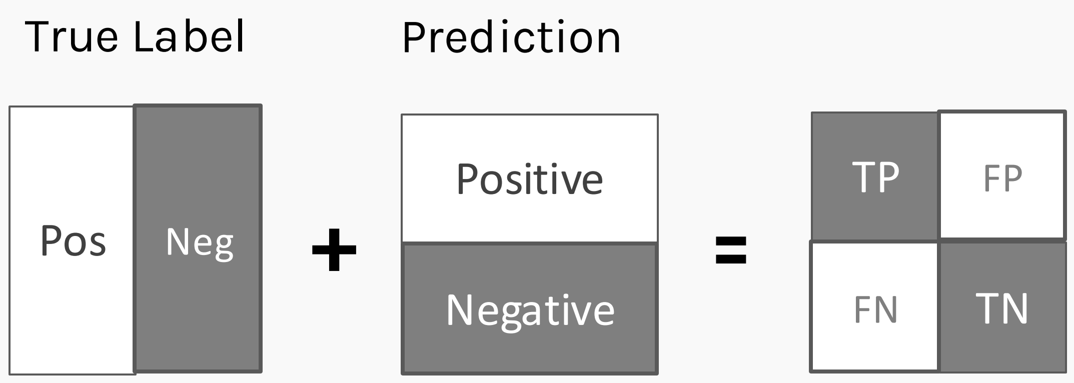
Generative **Adversarial** Networks (GANs)

Recap: Confusion Matrix

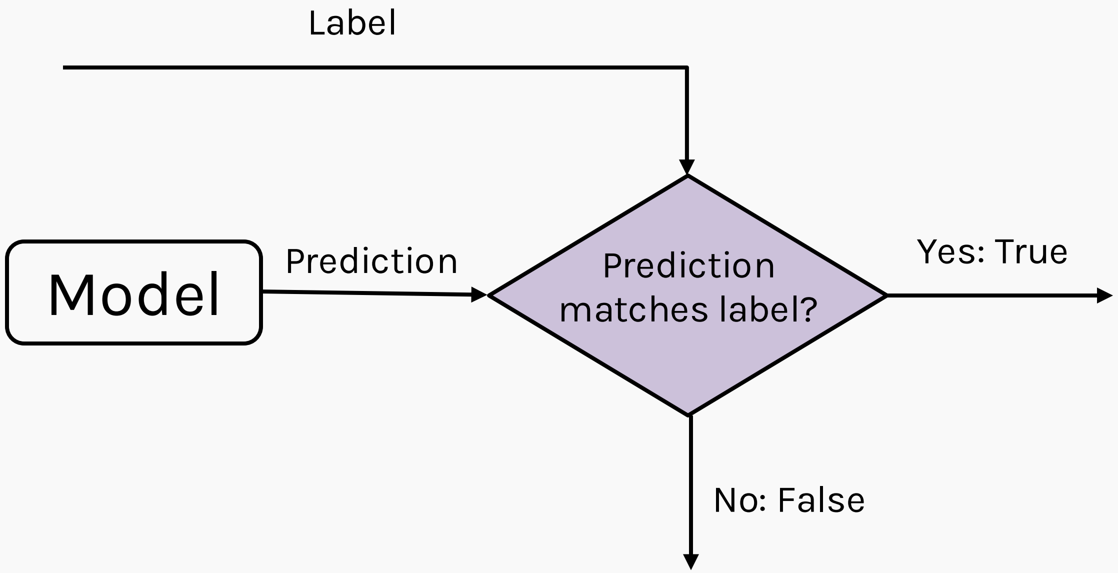
Consider the same spam dataset.

Positive: If model predicts the mail **is spam**.
Negative: If model predicts the mail is **not spam**.

True: If model prediction and real label **match.**
False: If model prediction and real label **do not match.**

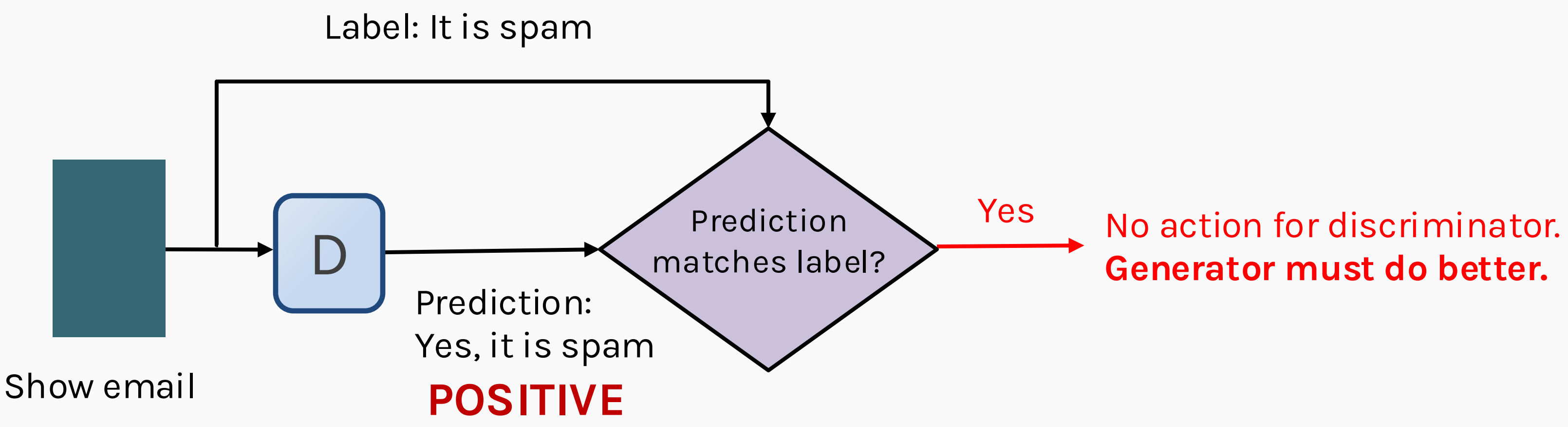


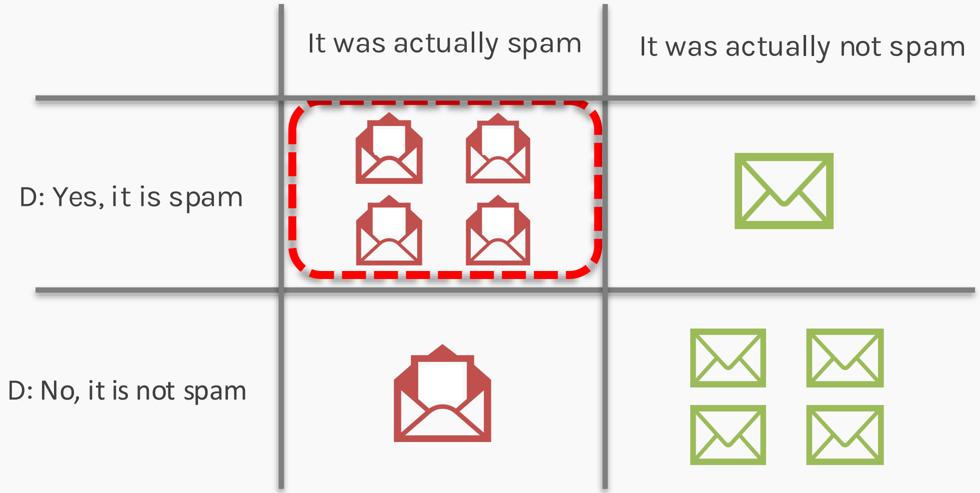
Recap: Confusion Matrix



TP	FP
FN	TN

Generative Adversarial Networks (GANs): True Positive

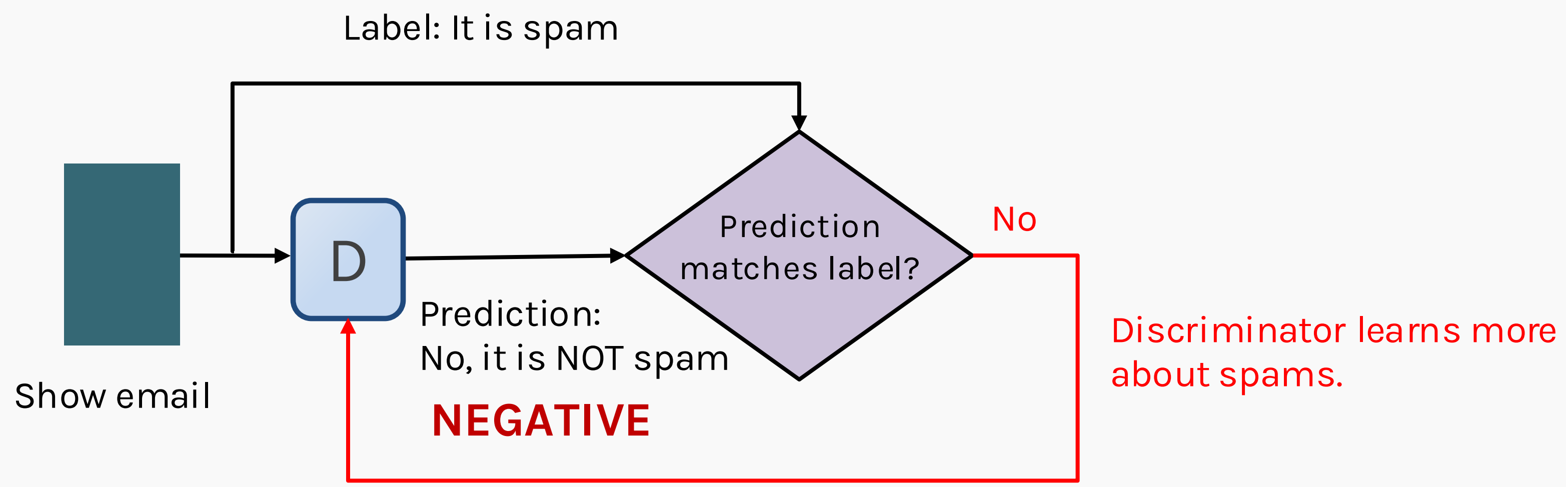


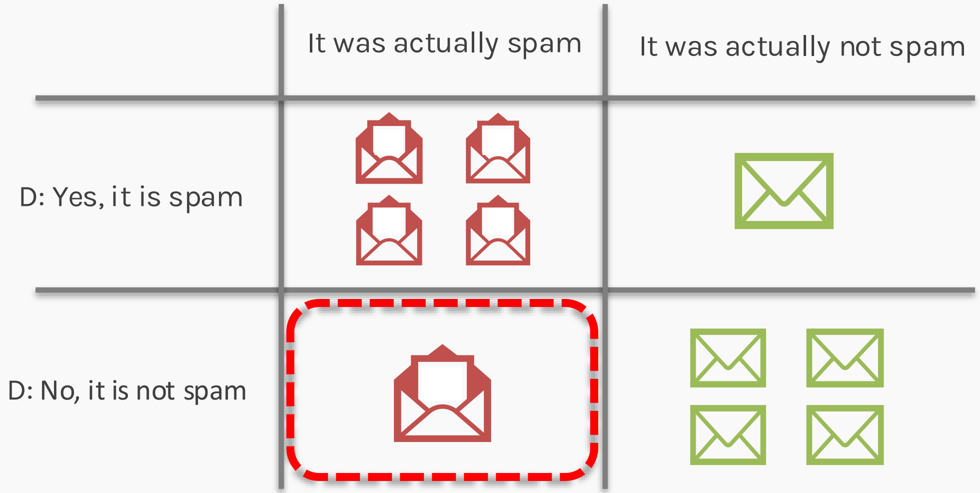


True Positive (I: Fake/D: Fake):

- 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): False Negative

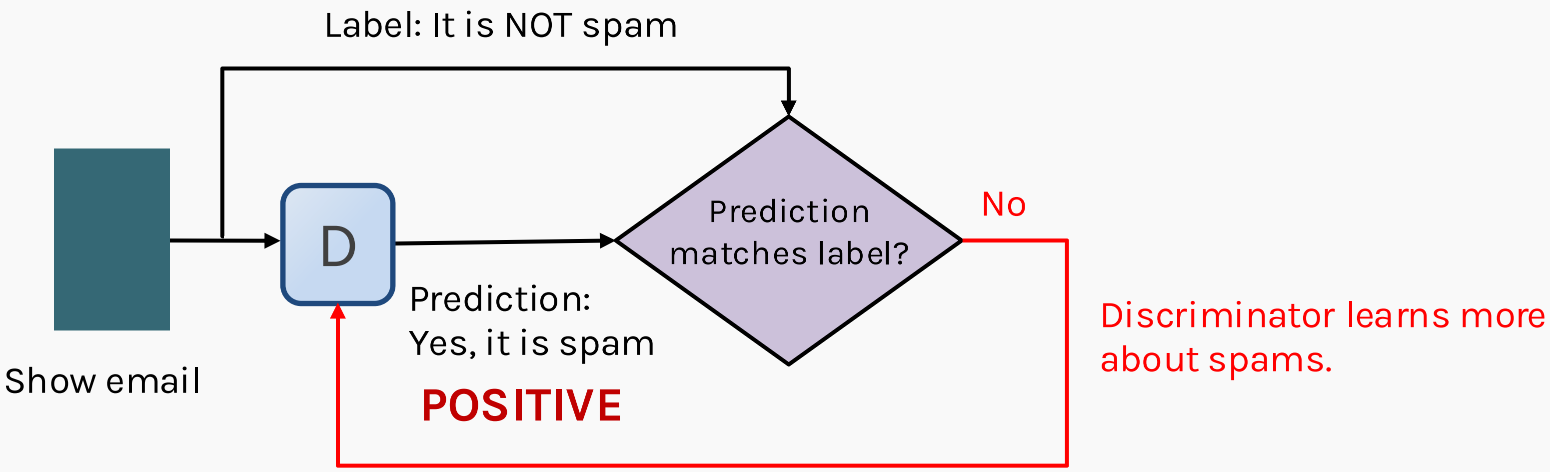


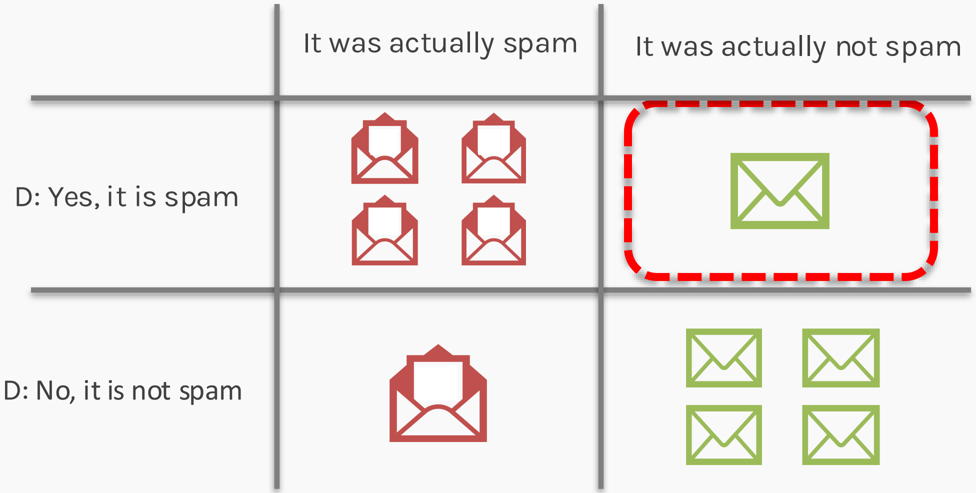


False Negative (I: Real/D: Fake):

- The discriminator sees an email and predicts it not a spam even though it is.
- The discriminator must learn more.

Generative Adversarial Networks (GANs): False Positive

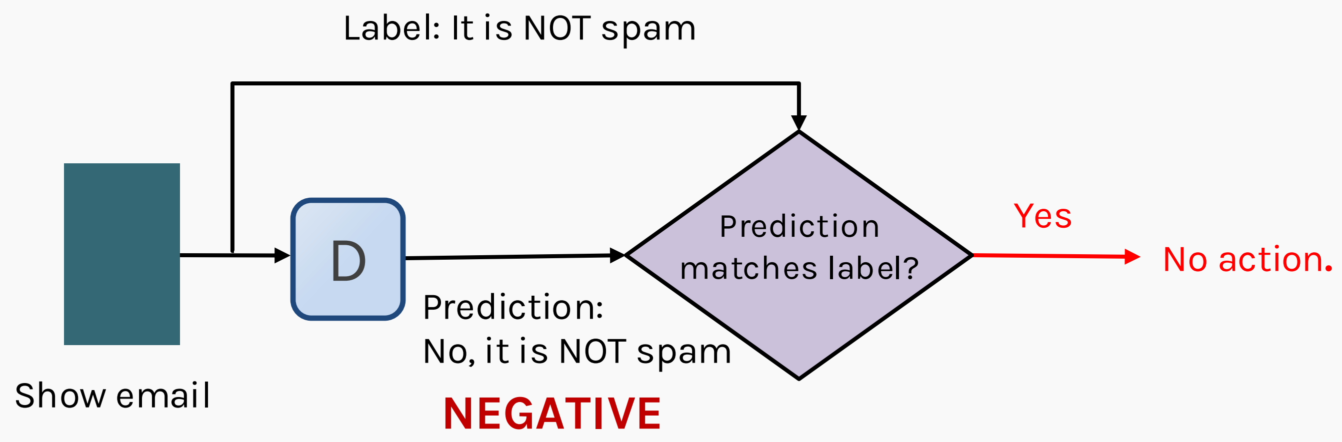


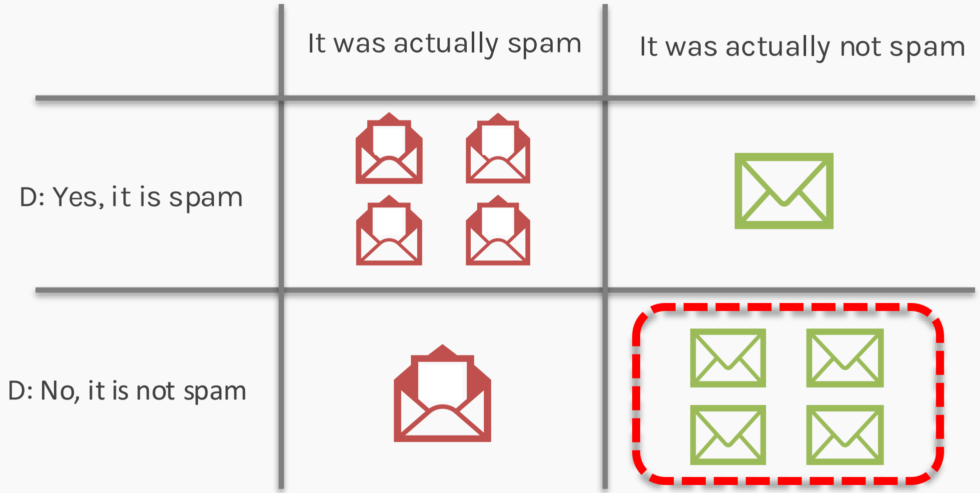


False Positive (I: Fake/D: Real):

- The discriminator sees an email and predicts it is a spam even though it is NOT.
- The discriminator must learn more.

Generative Adversarial Networks (GANs): True Negative



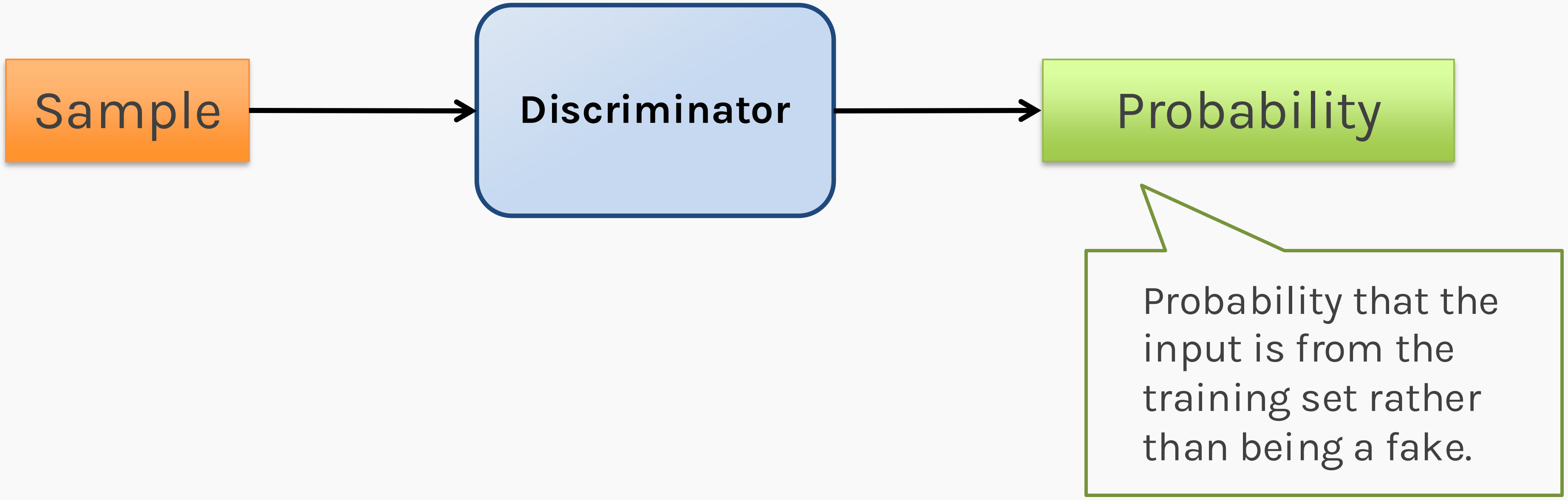


True Negative (I: Real/D: Real):
No action required by Generator or Discriminator.

Outline

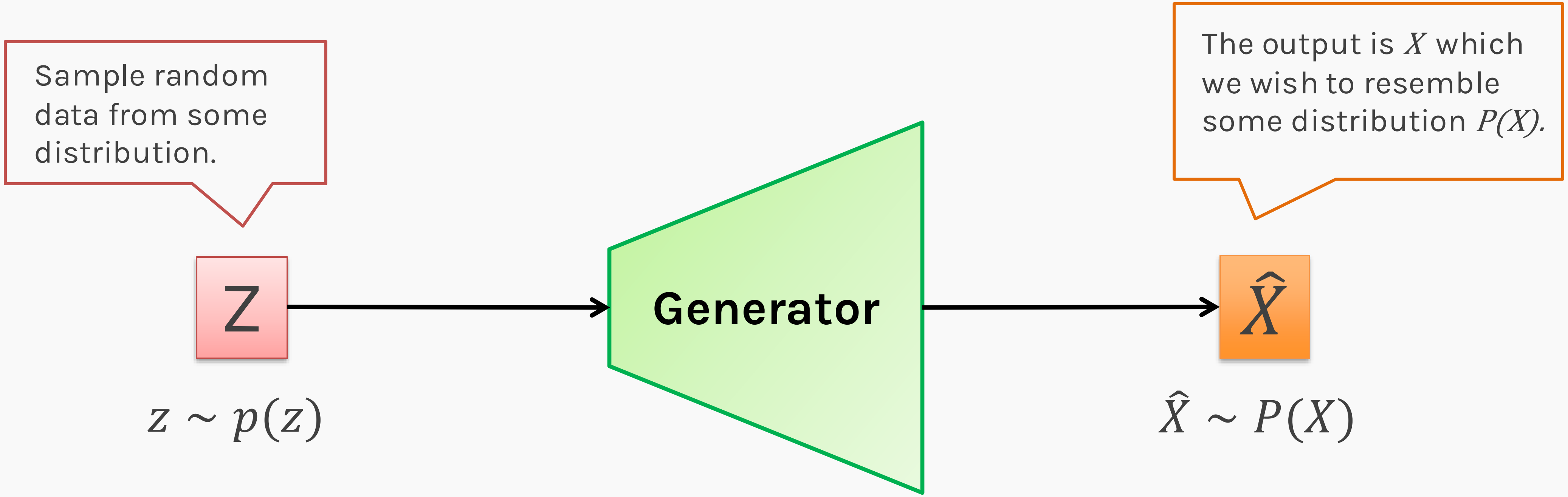
- Motivation for Generative Modeling
- High Level Formalism
- **Architecture**
- Mathematics
- Training GANS
- Deep Convolutional GANs

The Discriminator



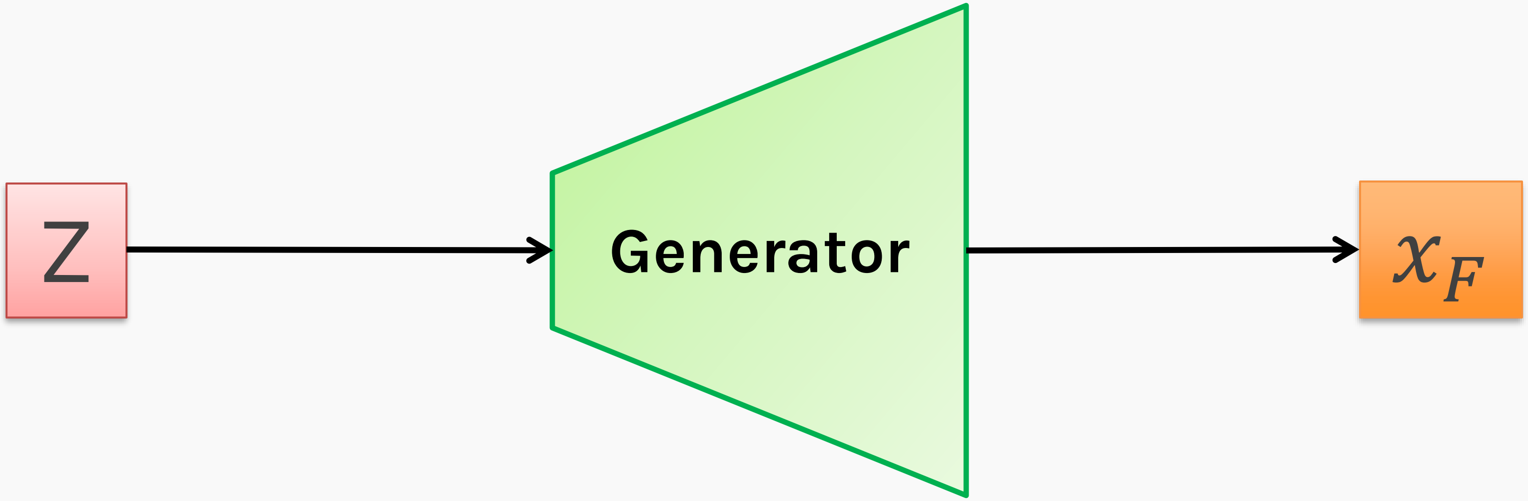
The discriminator is very simple. It is similar to any other classifier you have seen till now. There are not many restrictions on what the discriminator is.

The Generator



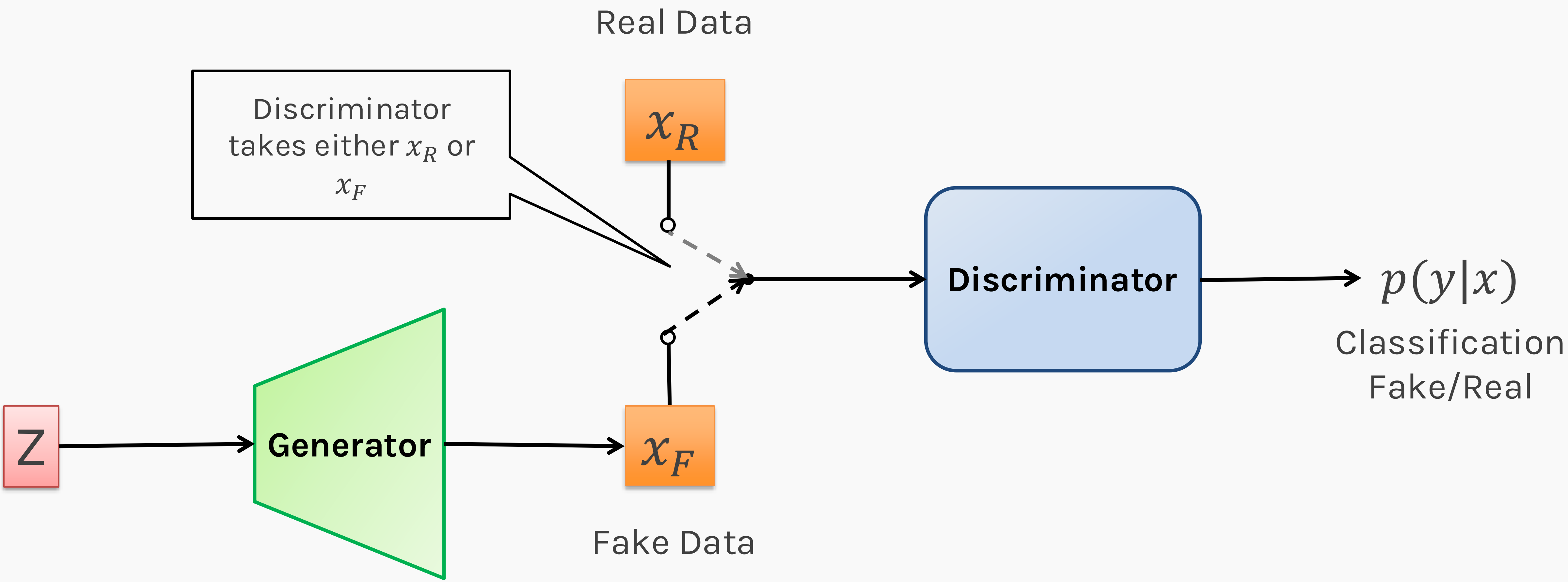
- If we build our generator to be deterministic, then the same input will always produce the same output.
- We want to generate data from a distribution. In that sense, we can think of the input values as latent variables.

The Generator



- 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.

The GAN



Real Data
Discriminator takes either x_R or x_F
x_R
Discriminator
$p(y|x)$
Classification Fake/Real
Z
Generator
x_F
Fake Data

Outline

- Motivation for Generative Modeling
- High Level Formalism
- Architecture
- **Mathematics**
- Training GANS
- Deep Convolutional GANs

Mathematics

As a binary classification problem, the loss function for GANS is the binary cross-entropy loss:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)$$

True label
Real = 1
Fake = 0

Predicted label

Mathematics

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)$$

The input to the Discriminator can be the real data or the fake data generated by the Generator. Splitting the loss function into two sums, we have:

$$\mathcal{L} = \quad -\frac{1}{N_{Real}}\sum_{i=1}^{N_{Real}} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)$$

$$-\frac{1}{N_{Fake}}\sum_{i=1}^{N_{Fake}} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)$$

Mathematics

True label - Real
$y_i = 1$

$$\mathcal{L} = -\frac{1}{N_{Real}}\sum_{i=1}^{N_{Real}} \mathbf{1}\,\log(\hat{y}_i) + (1 - y_i)\log(1-\hat{y}_i)$$

$$-\frac{1}{N_{Fake}}\sum_{i=1}^{N_{Fake}} y_i\log(\hat{y}_i) + \mathbf{1}\,\log(1-\hat{y}_i)$$

True label - Fake
$y_i = 0$
$1 - y_i = 1$

Mathematics

True label - Real

$y_i = 1$

$1 - y_i = 0$

$$\mathcal{L} = -\frac{1}{N_{Real}} \sum_{i=1}^{N_{Real}} \mathbf{1} \log(\hat{y}_i) + \cancel{(1 - y_i) \log(1 - \hat{y}_i)}$$

$$-\frac{1}{N_{Fake}} \sum_{i=1}^{N_{Fake}} \cancel{y_i \log(\hat{y}_i)} + 1 \log(1 - \hat{y}_i)$$

True label - Fake

$y_i = 0$

Mathematics

$$\mathcal{L} = \quad -\frac{1}{N_{Real}} \sum_{i=1}^{N_{Real}} \log(\hat{y}_i)$$

$$-\frac{1}{N_{Fake}} \sum_{i=1}^{N_{Fake}} \log(1 - \hat{y}_i)$$

Mathematics

$$\mathcal{L} = -\frac{1}{N_{Real}} \sum_{i=1}^{N_{Real}} \log(\hat{y}_i) - \frac{1}{N_{Fake}} \sum_{i=1}^{N_{Fake}} \log(1 - \hat{y}_i)$$

Rewriting in terms of discriminator's, **D,** output:

$$\mathcal{L} = -\frac{1}{N_{Real}} \sum_{i=1}^{N_{Real}} \log(\boldsymbol{D}(\boldsymbol{x}_i^R)) - \frac{1}{N_{Fake}} \sum_{i=1}^{N_{Fake}} \log(1 - \boldsymbol{D}(\boldsymbol{x}_i^F))$$

Mathematics

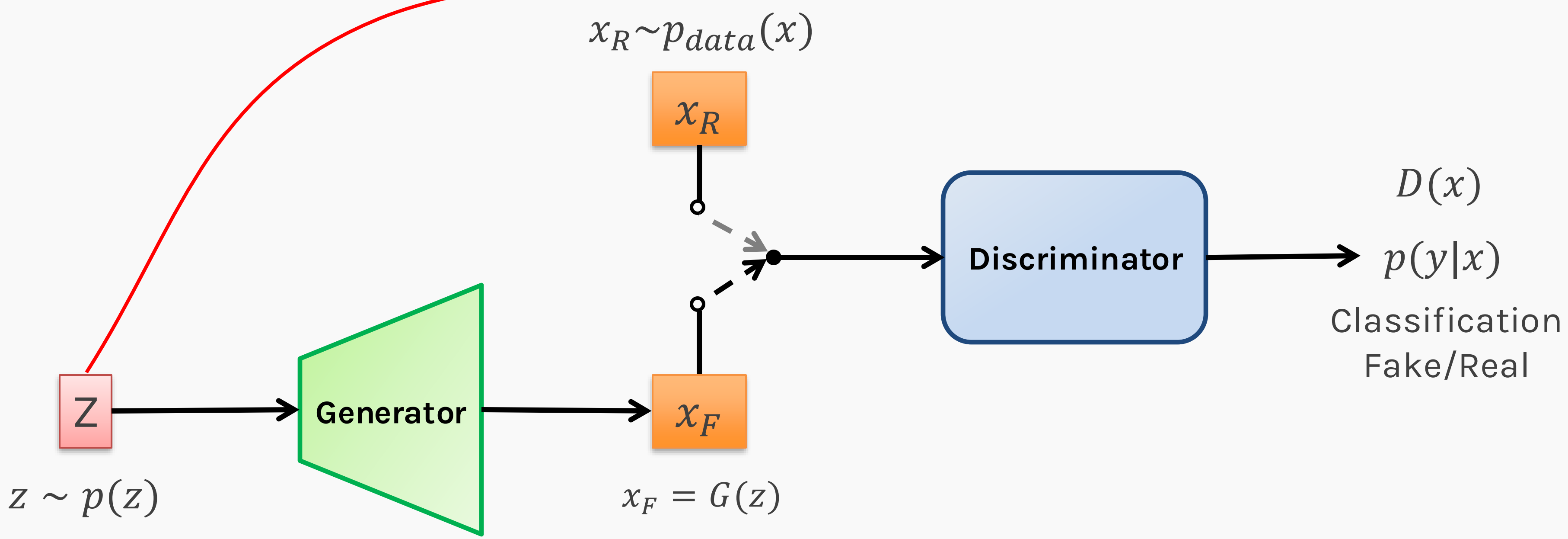
$$\mathcal{L} = -\frac{1}{N_{Real}}\sum_{i=1}^{N_{Real}} \log(D(x_i^R)) - \frac{1}{N_{Fake}}\sum_{i=1}^{N_{Fake}} \log(1 - D(x_i^F))$$

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

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

Mathematics

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



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

Mathematics

$$\mathcal{L} = -\mathbb{E}_{x\sim \mathrm{p}_{\mathrm{data}}(x)}\log(D(x)) - \mathbb{E}_{\mathrm{z}\sim \mathrm{p}_{\mathrm{Z}}(\mathrm{z})}\log(1 - D(G(z)))$$

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

$$\min_{G}\max_{D} V(G, D) = \min_{G}\max_{D} \mathbb{E}_{x\sim \mathrm{p}_{\mathrm{data}}(x)}\log(D(x)) + \mathbb{E}_{\mathrm{z}\sim \mathrm{p}_{\mathrm{Z}}(\mathrm{z})}\log(1 - D(G(z)))$$

Mathematics

Minimax value function

Discriminator’s prediction on real data

Discriminator’s prediction on fake data

$$\boxed{\min_G \max_D V(G,D)} = \min_G \max_D \mathbb{E}_{\boxed{x\sim \mathrm{p}_{\mathrm{data}}(x)}} \log\boxed{\left(D(x)\right)} + \mathbb{E}_{\boxed{\mathrm{z}\sim \mathrm{p}_{\mathrm{Z}}(\mathrm{z})}} \log(1 - \boxed{D\left(\boxed{G(z)}\right)})$$

Sample real data

Sample random noise

Generator’s output: fake data

Mathematics

Minimax value function

Discriminator: Maximize

$$\underbrace{\min_G \max_D V(G,D)}_{\text{Minimax value function}} = \min_G \max_D \overbrace{\mathbb{E}_{x\sim \mathrm{p}_{\mathrm{data}}(x)} \log\big(D(x)\big) + \underbrace{\mathbb{E}_{\mathrm{z}\sim \mathrm{p}_{\mathrm{Z}}(\mathrm{z})} \log(1 - D\big(G(z)\big))}_{\text{Generator: Minimize}}}^{\text{Discriminator: Maximize}}$$

Generator: Minimize

Mathematics

Minimax value function

$$\boxed{\min_G \max_D V(G,D)} = \min_G \max_D \overbrace{\mathbb{E}_{x\sim p_{\text{data}}(x)} \log\big(D(x)\big) + \mathbb{E}_{z\sim p_Z(z)} \log\big(1 - D\big(G(z)\big)\big)}^{\text{Discriminator: Maximize}}$$

$$\underbrace{\mathbb{E}_{z\sim p_Z(z)} \log\big(D\big(G(z)\big)\big)}_{\text{Generator: Maximize}}$$

Outline

- Motivation for Generative Modeling
- High Level Formalism
- Architecture
- Mathematics
- **Training GANS**
- Deep Convolutional GANs

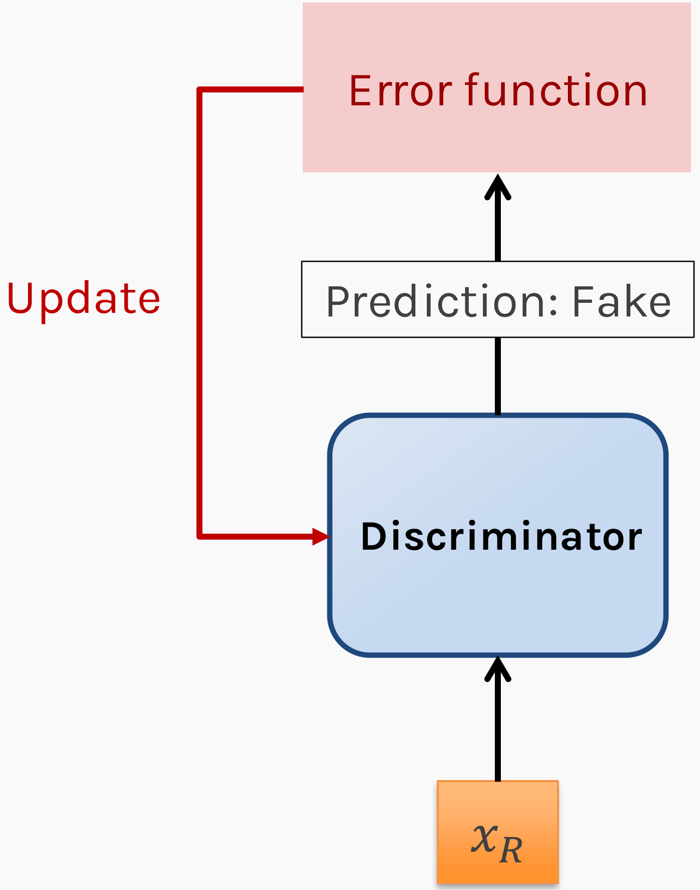
Training GANs

Let us now formalize the training you saw previously in the spam example.

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

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

- The discriminator incorrectly classifies a real as a fake.
- The Generator is not involved in this step at all.
- The error drives a backpropagation step through the discriminator so that it will get better at recognizing reals.



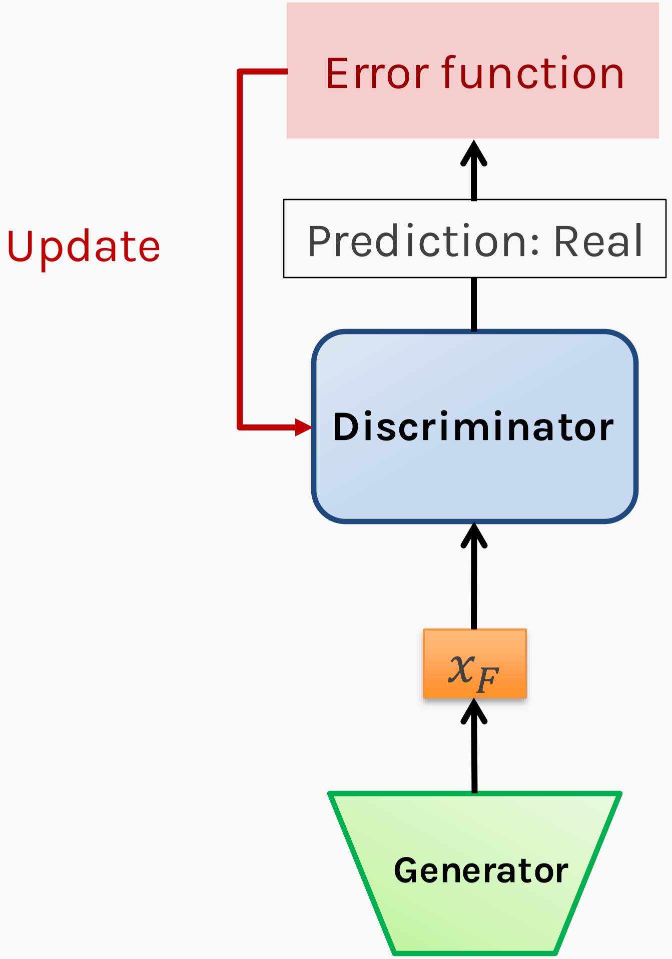
Training GANs

Let us now formalize the training you saw previously in the spam example.

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

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

- The discriminator incorrectly classifies a fake that is generated by the generator as real.
- The error drives a backpropagation step through the discriminator so that it will get better at recognizing reals.



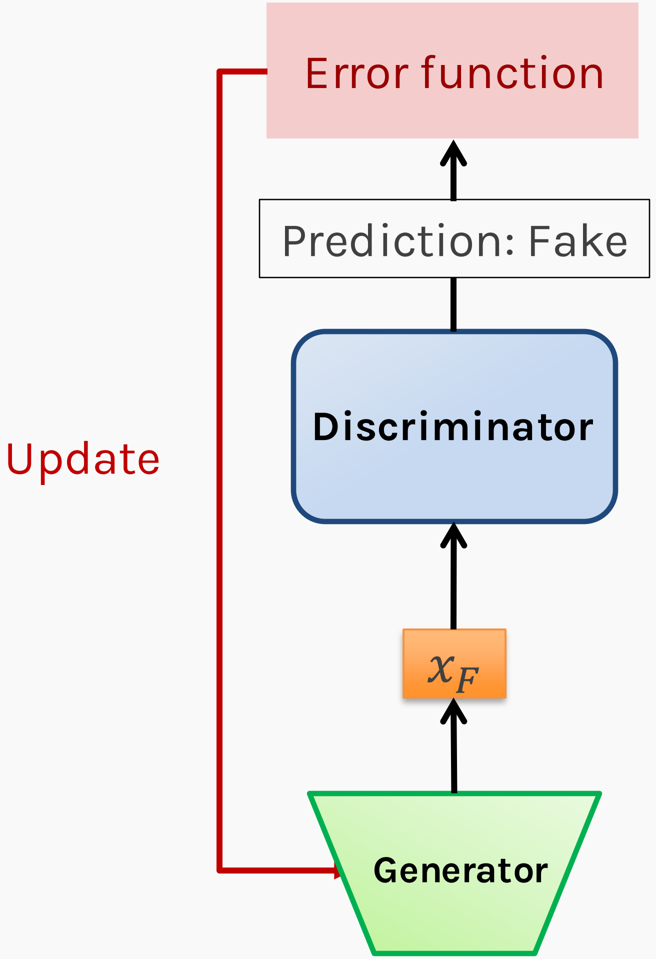
Training GANs

Let us now formalize the training you saw previously in the spam example.

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

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

- The discriminator correctly classifies a fake that is generated by the generator. Meaning the generator is caught.
- The error drives a backpropagation step through the discriminator (which is frozen) to the generator, updating its weights, so that it will get better at fooling the discriminator.

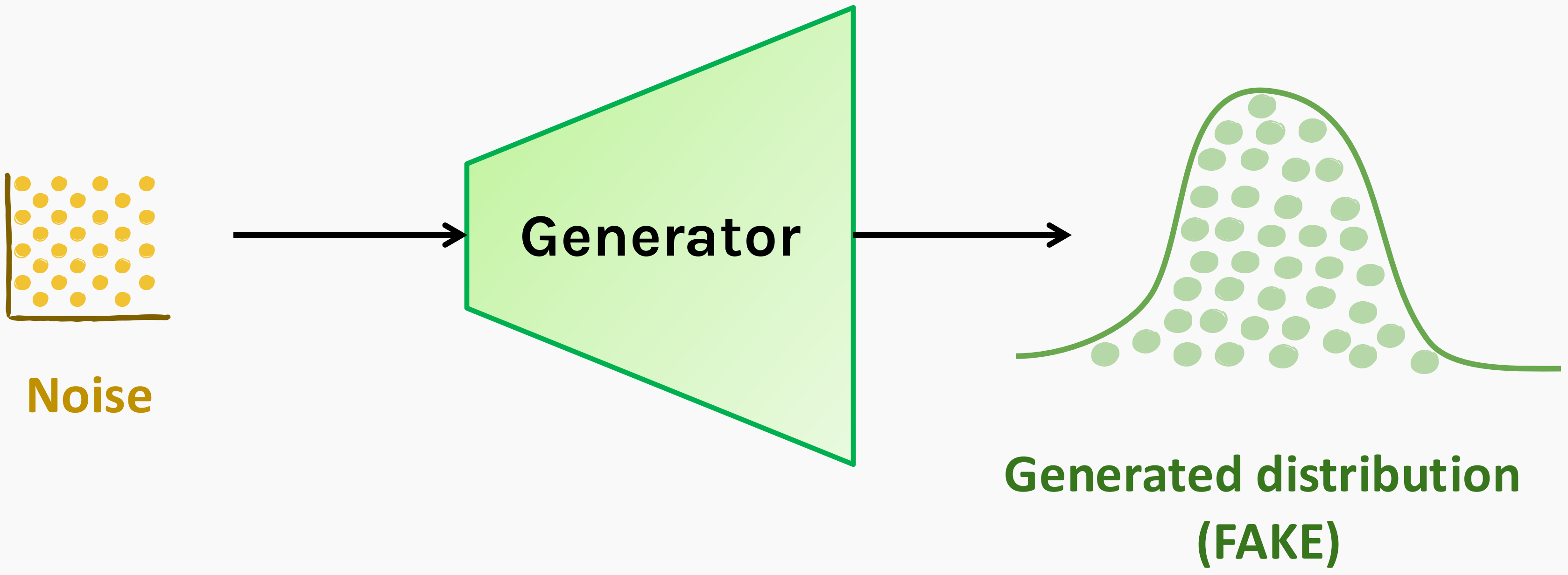


Training GANs

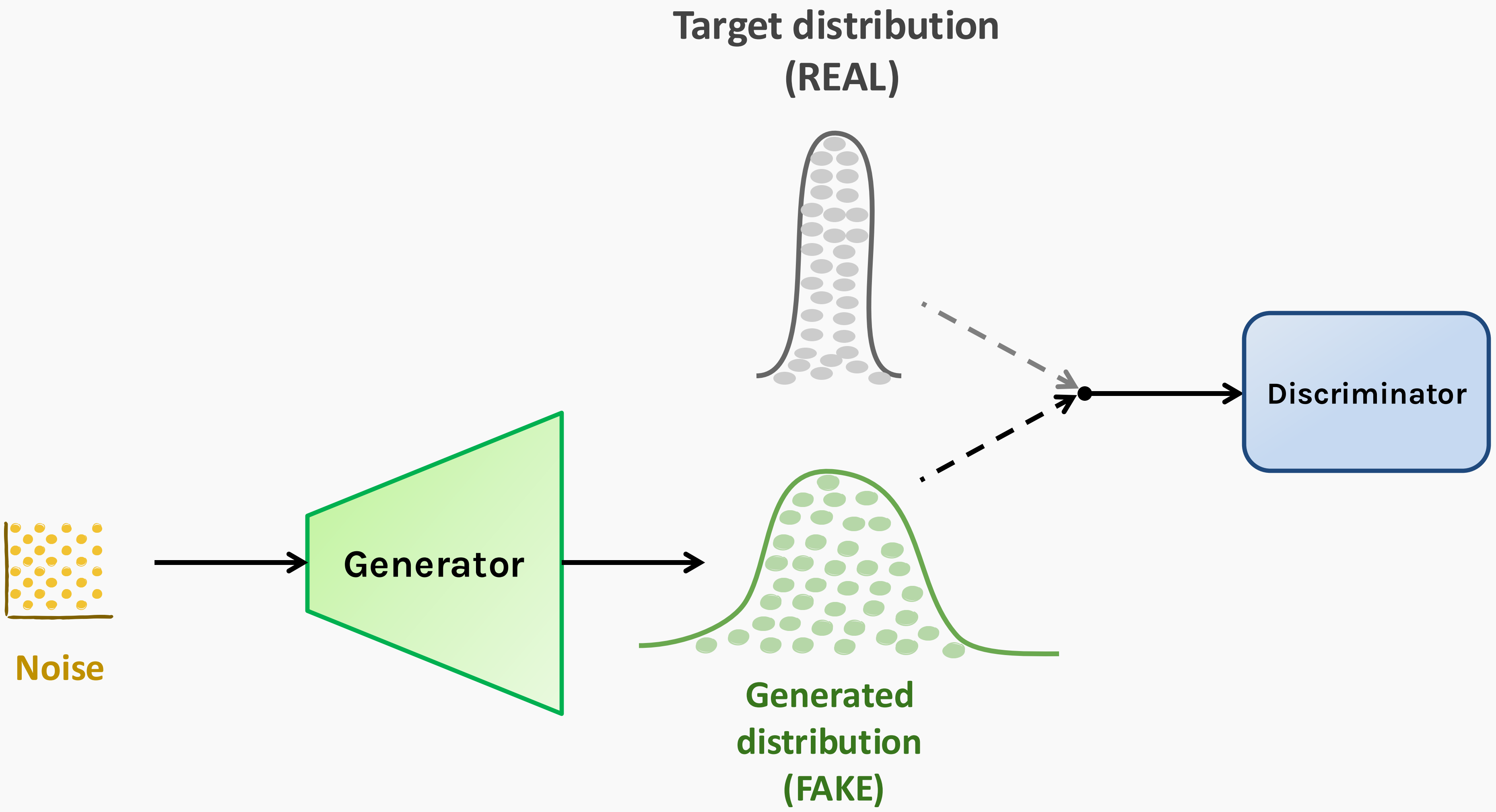
The process – known as **Learning Round** - accomplishes 3 jobs:

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.

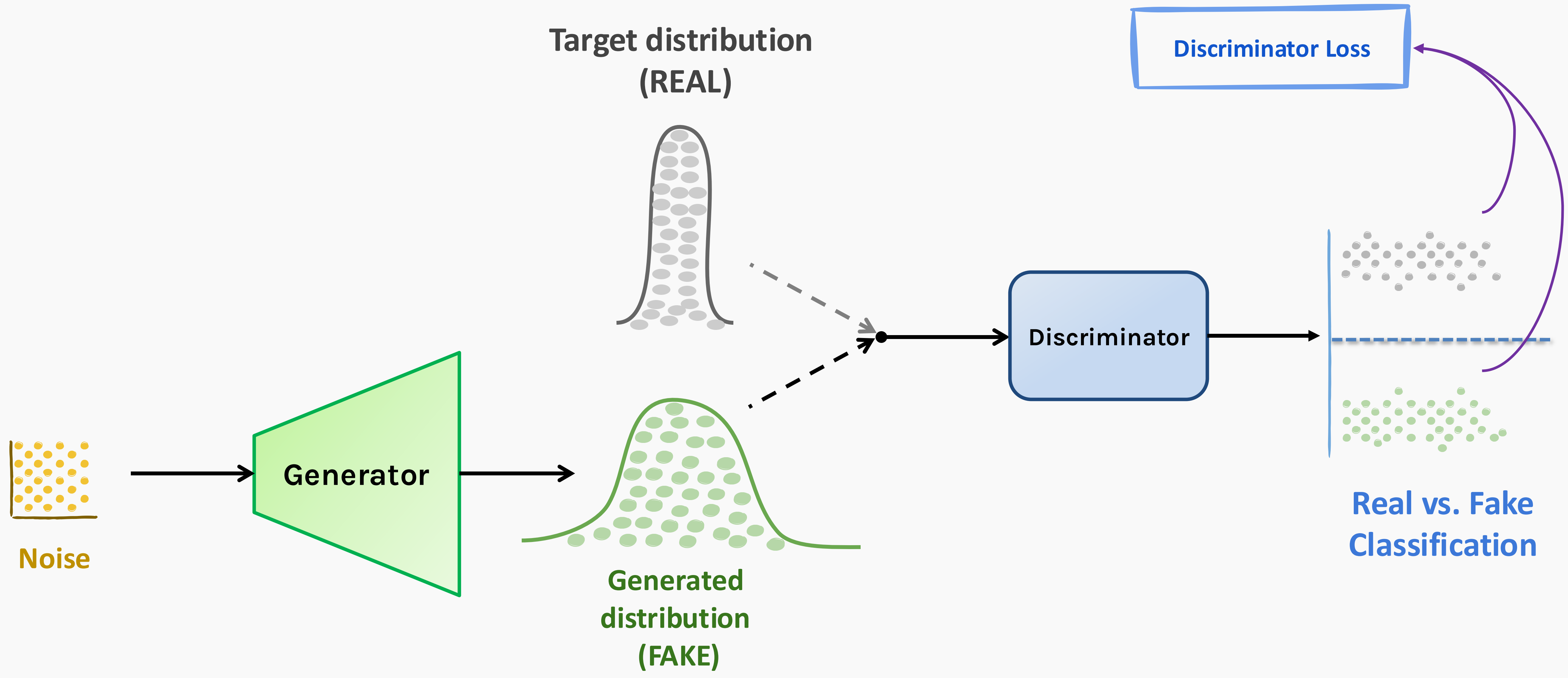
Training GANs



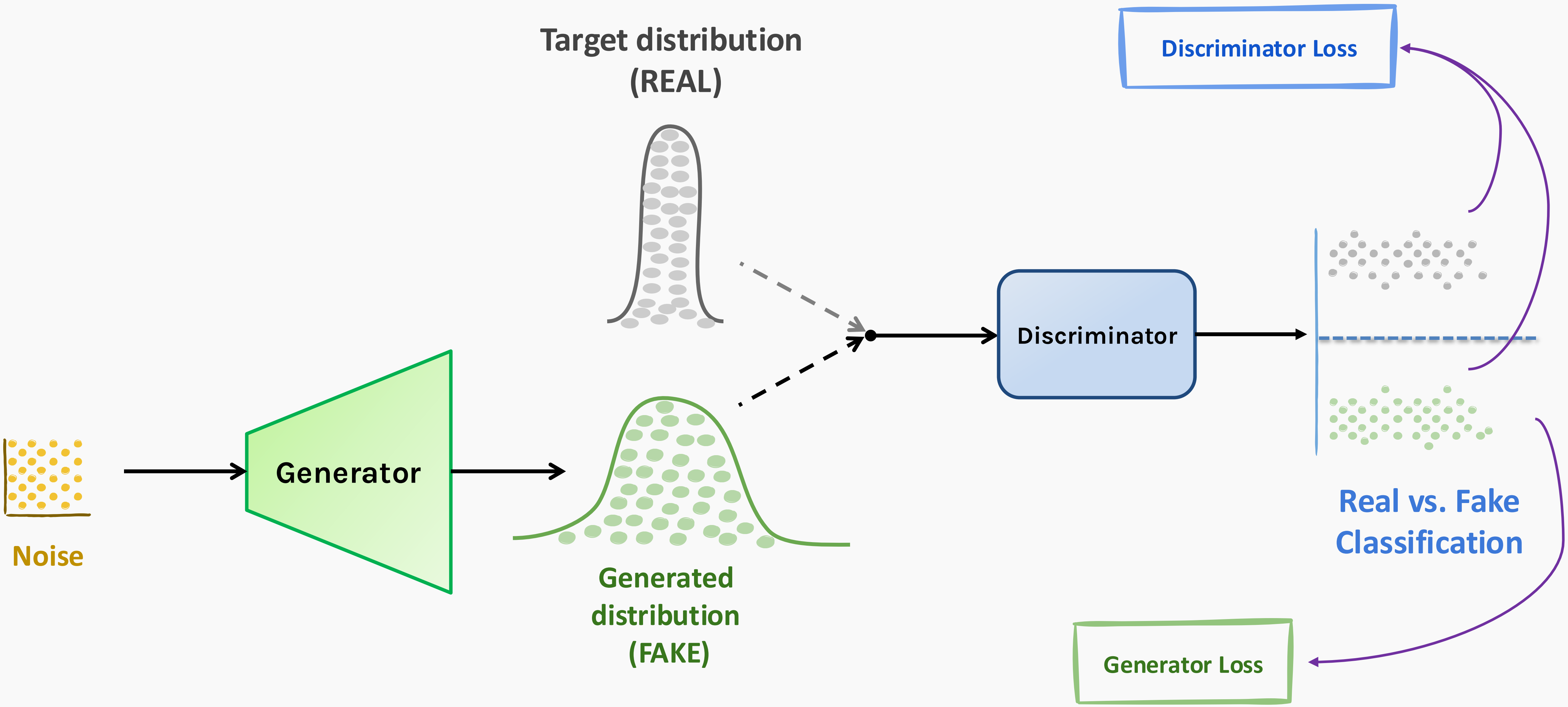
Training GANs: Forward Pass



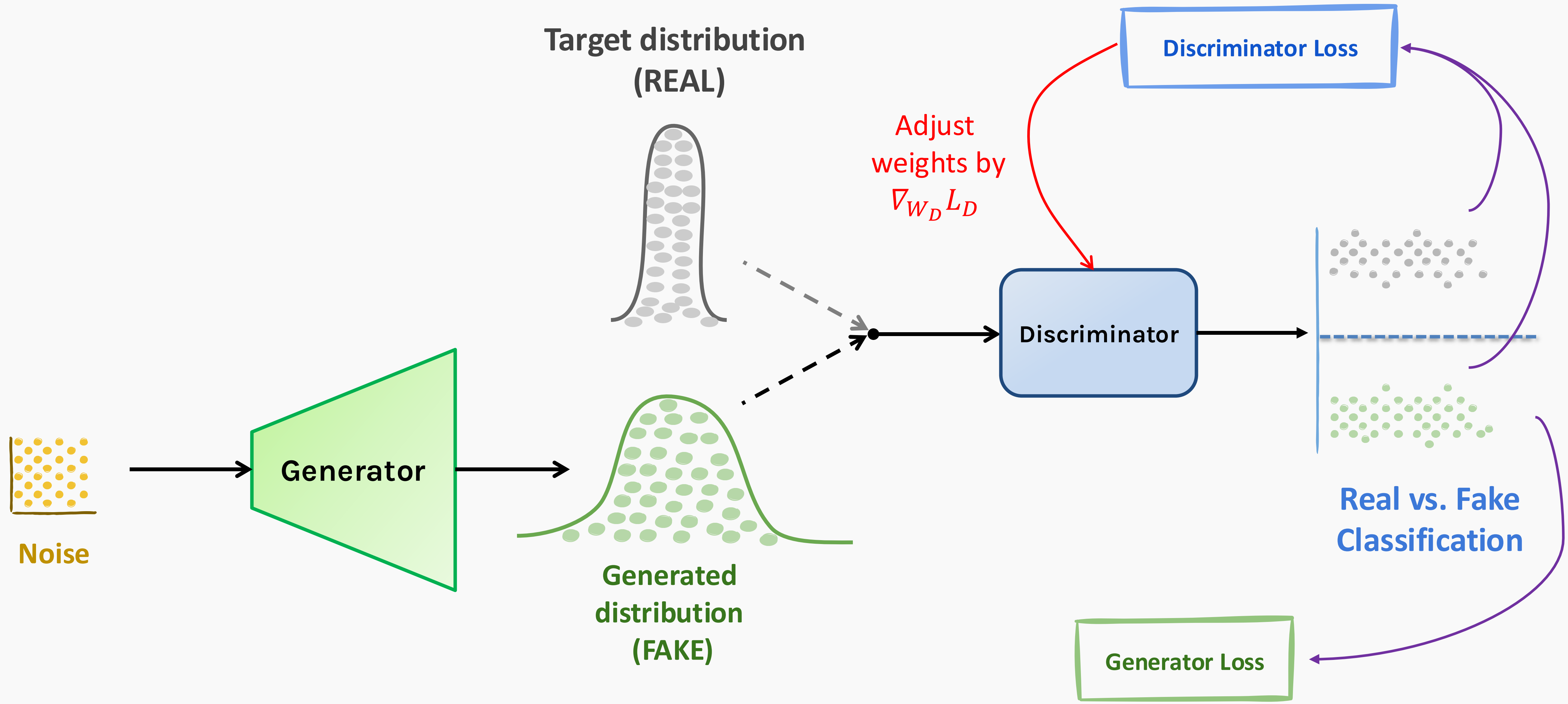
Training GANs: Forward Pass



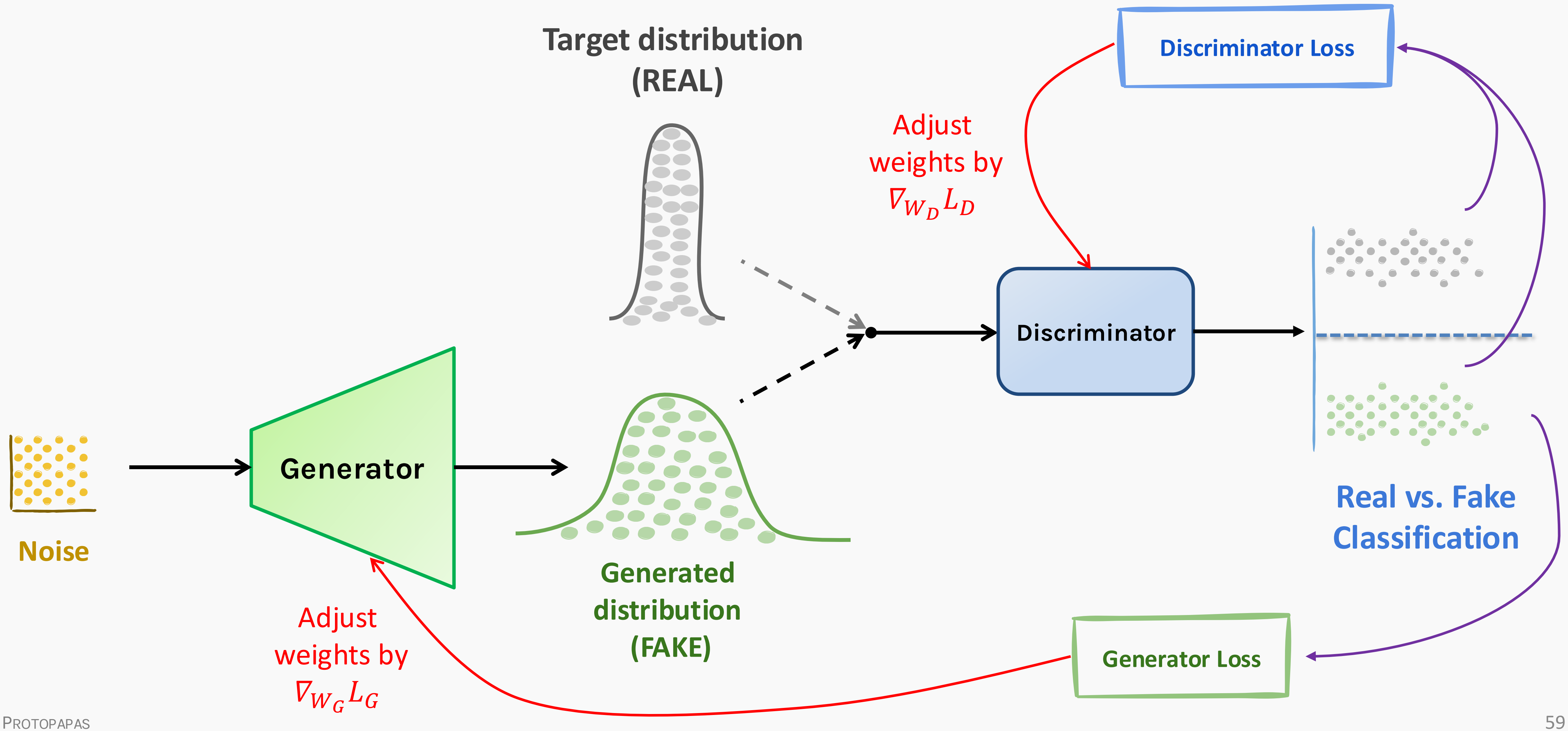
Training GANs: **Forward Pass**



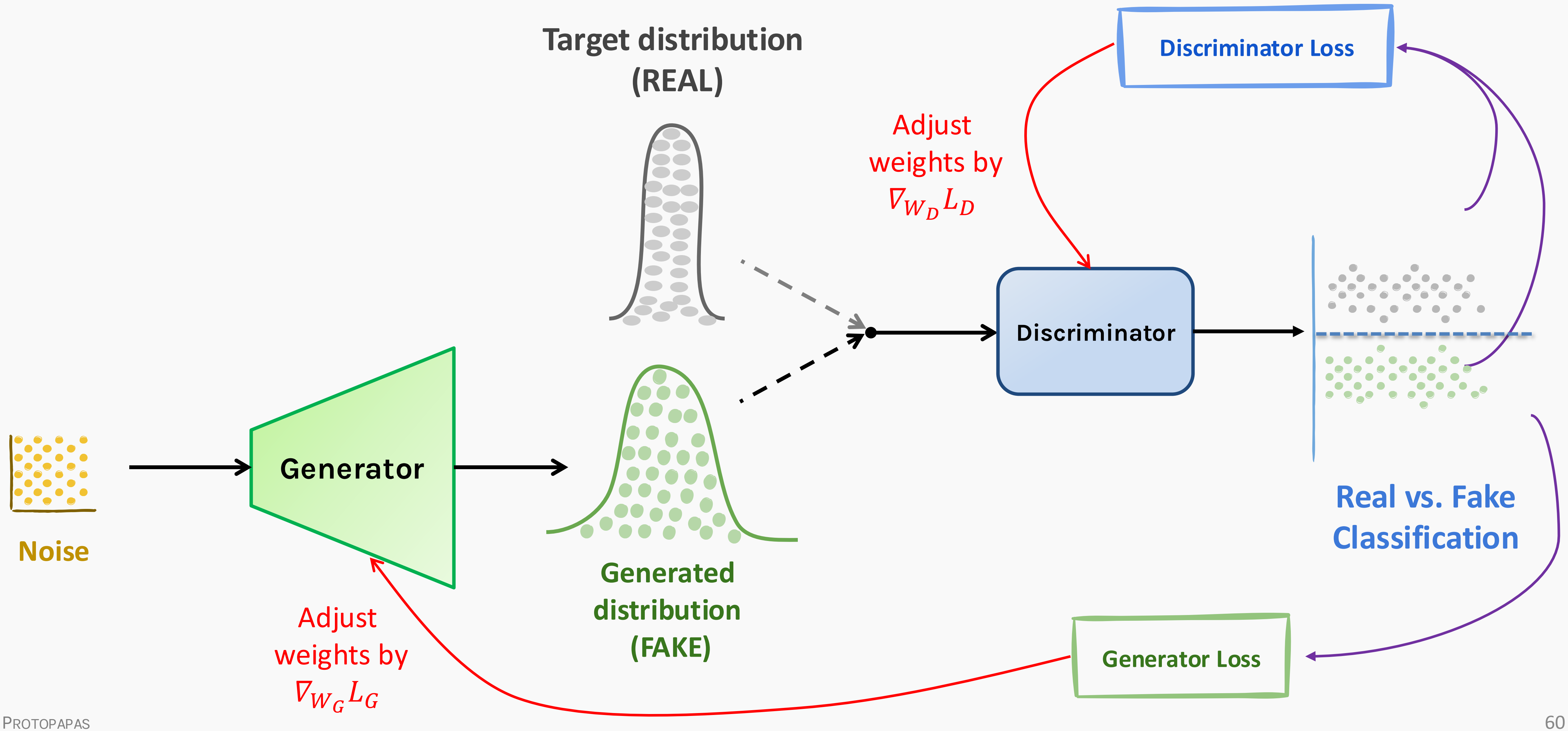
Training GANs: Backward Pass



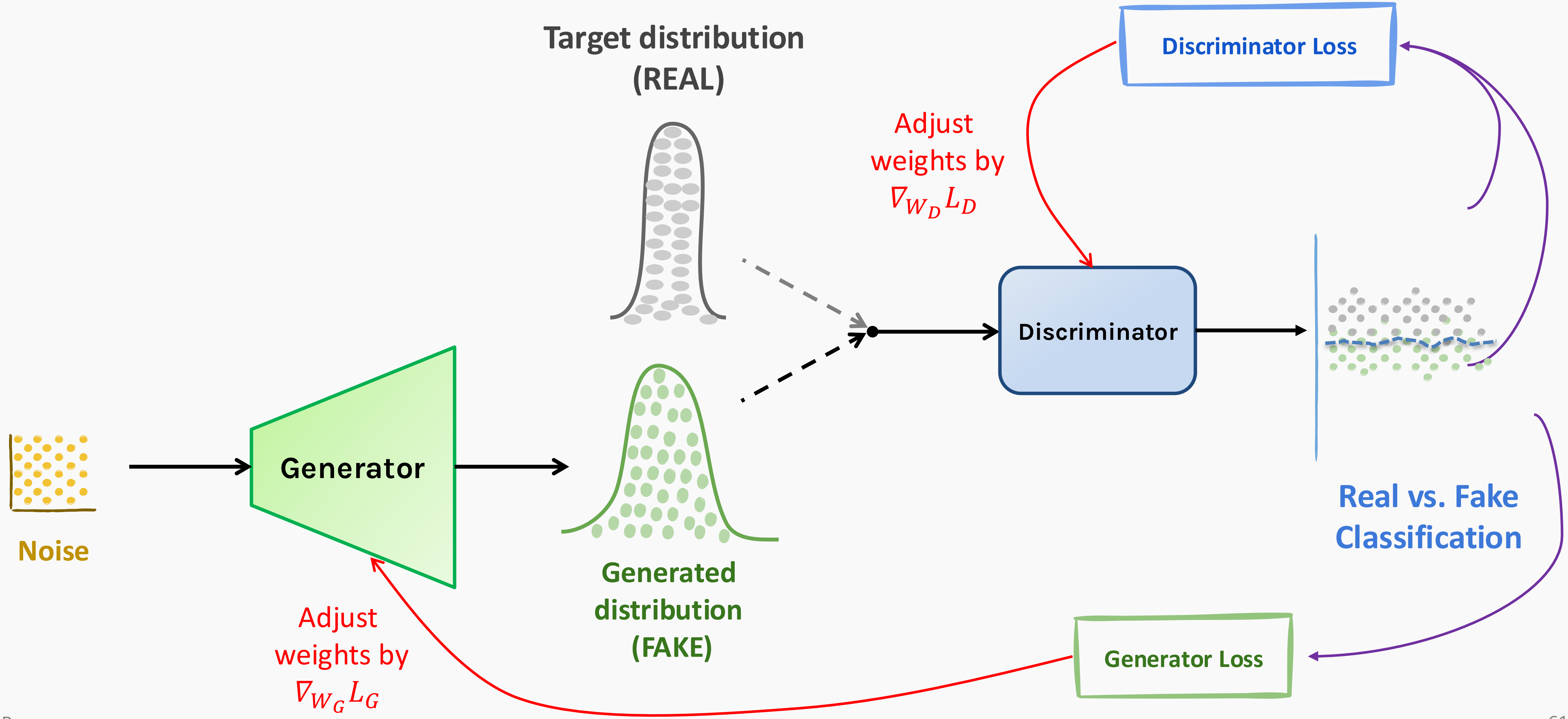
Training GANs: **Backward Pass**



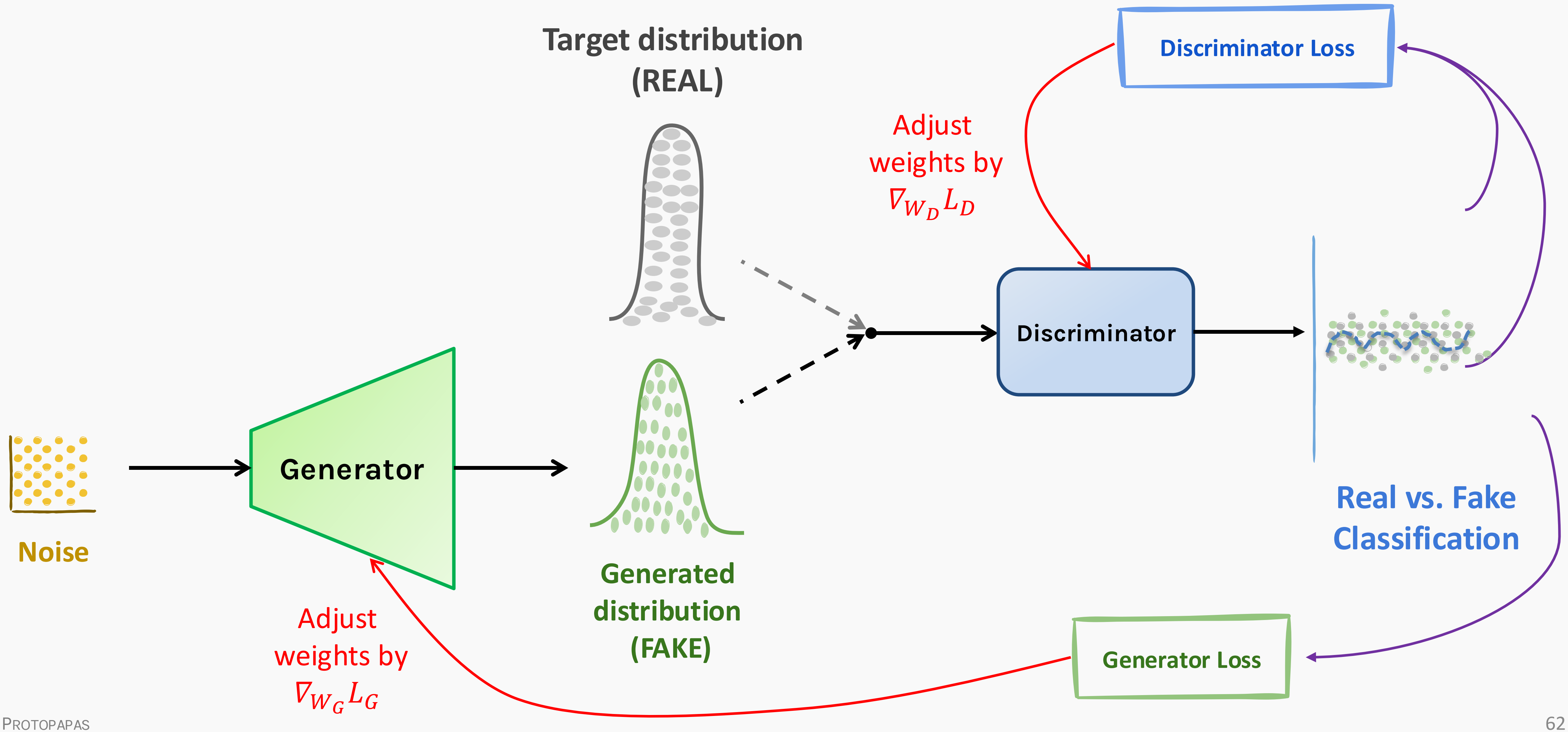
Training GANs: Forward/Backward Pass



Training GANs: Forward/Backward Pass



Training GANs: Forward/Backward Pass



Training GANs - Vanilla

For number of training iterations **do:**

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

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

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

$$\nabla_{W_g} \frac{1}{m} \sum_{i=1}^{m} \left[\log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right) \right]$$

Training GANs - Vanilla

For number of training iterations **do:**

For k steps **do:**

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

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

End **for**

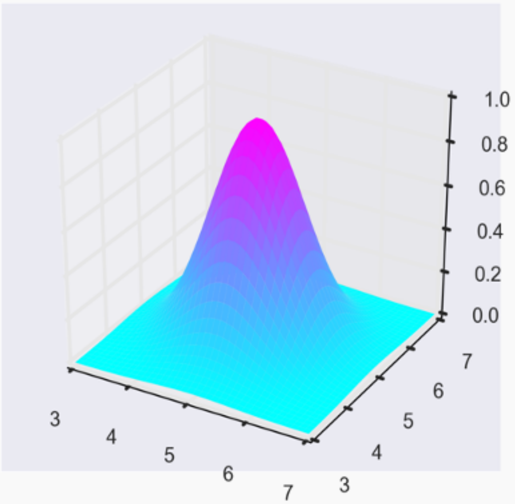
- Sample minibatch of m noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
- Update the generator by descending its stochastic gradient:

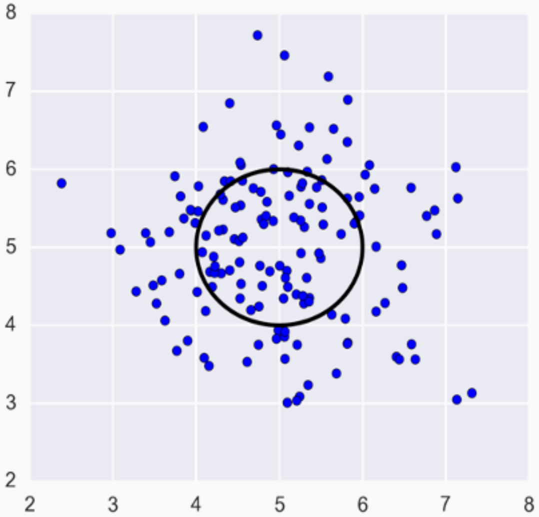
$$\nabla_{W_g} \frac{1}{m} \sum_{i=1}^{m} \left[\log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right) \right]$$

Building GANS: Fully Connected Case

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

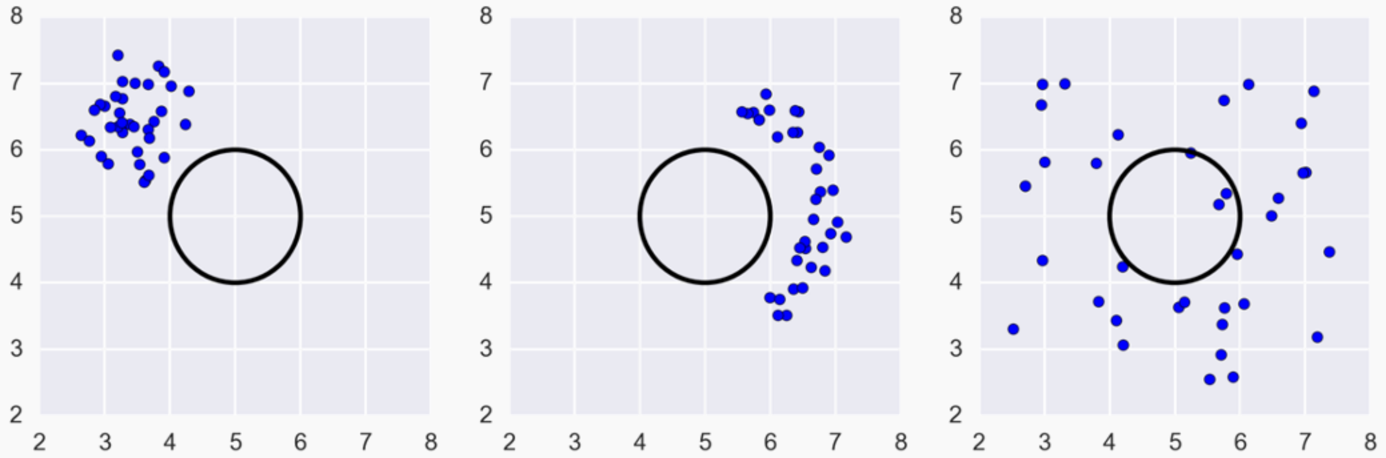




Building GANS: Fully Connected Case

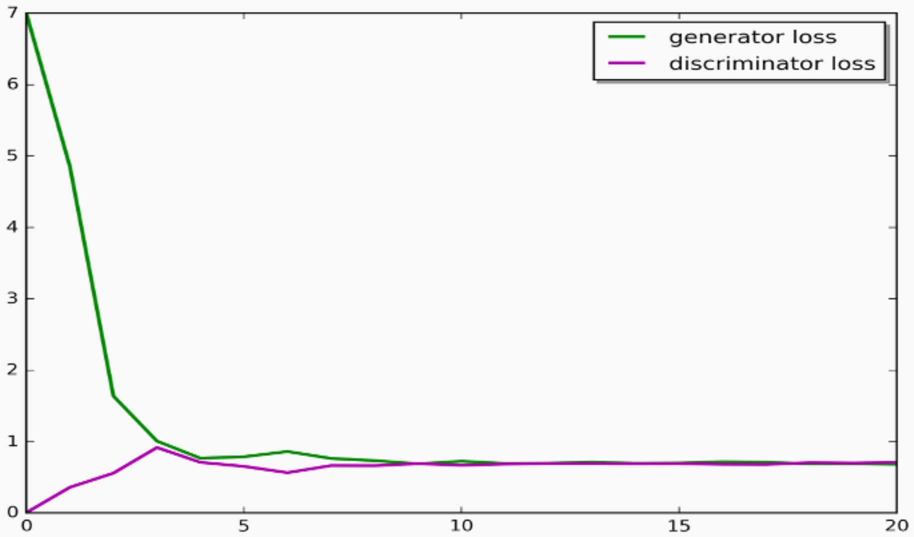
Train the Networks based on their ability to generate/discriminate batches of points drawn from the distribution.

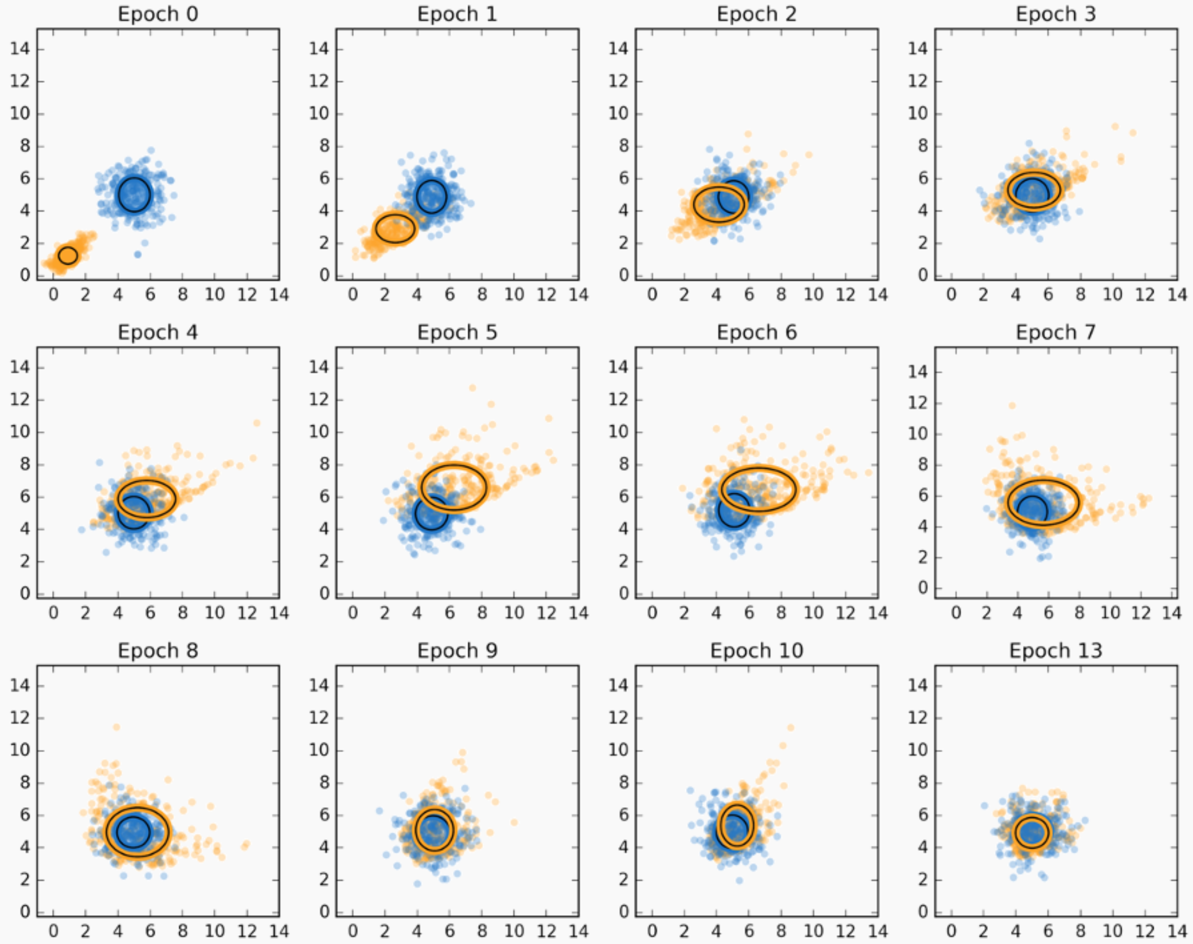
Are these batches of points drawn from the right distribution?



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).



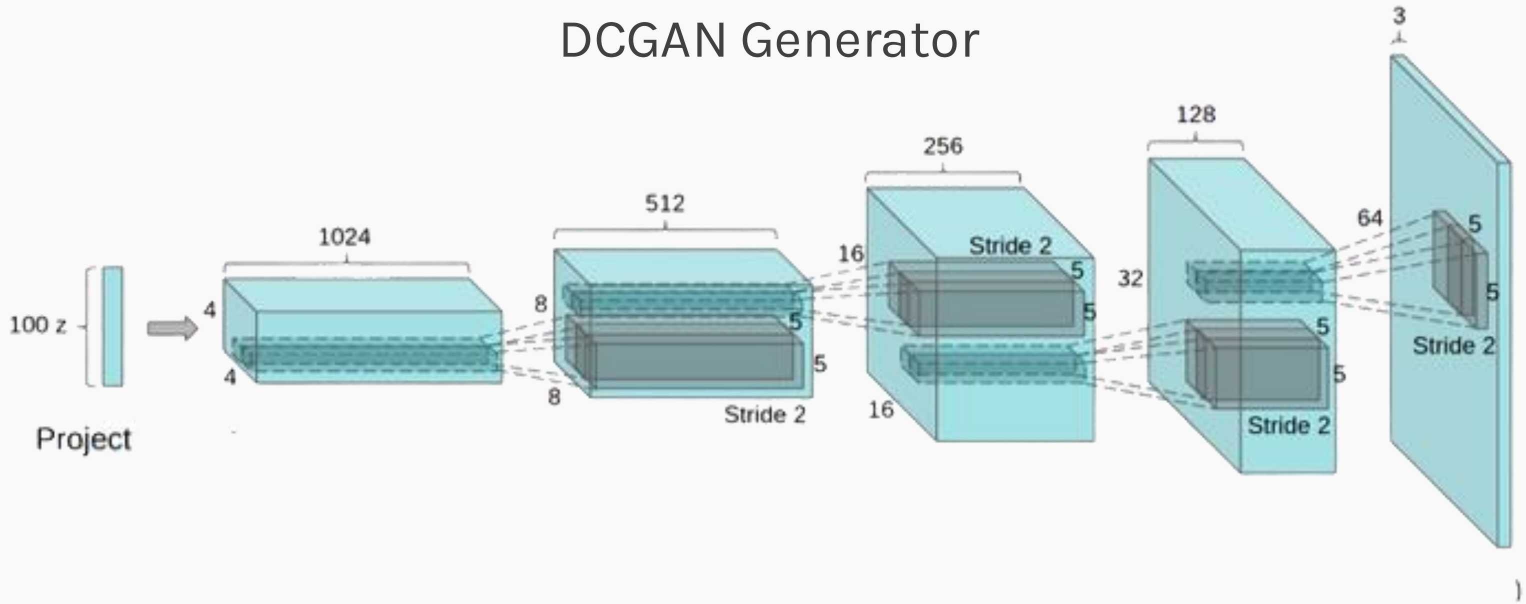


Outline

- Motivation for Generative Modeling
- High Level Formalism
- Architecture
- Mathematics
- Training GANS
- **Deep Convolutional GANs**

Deep Convolutional GAN: DCGAN - Alex Radford et al. 2016

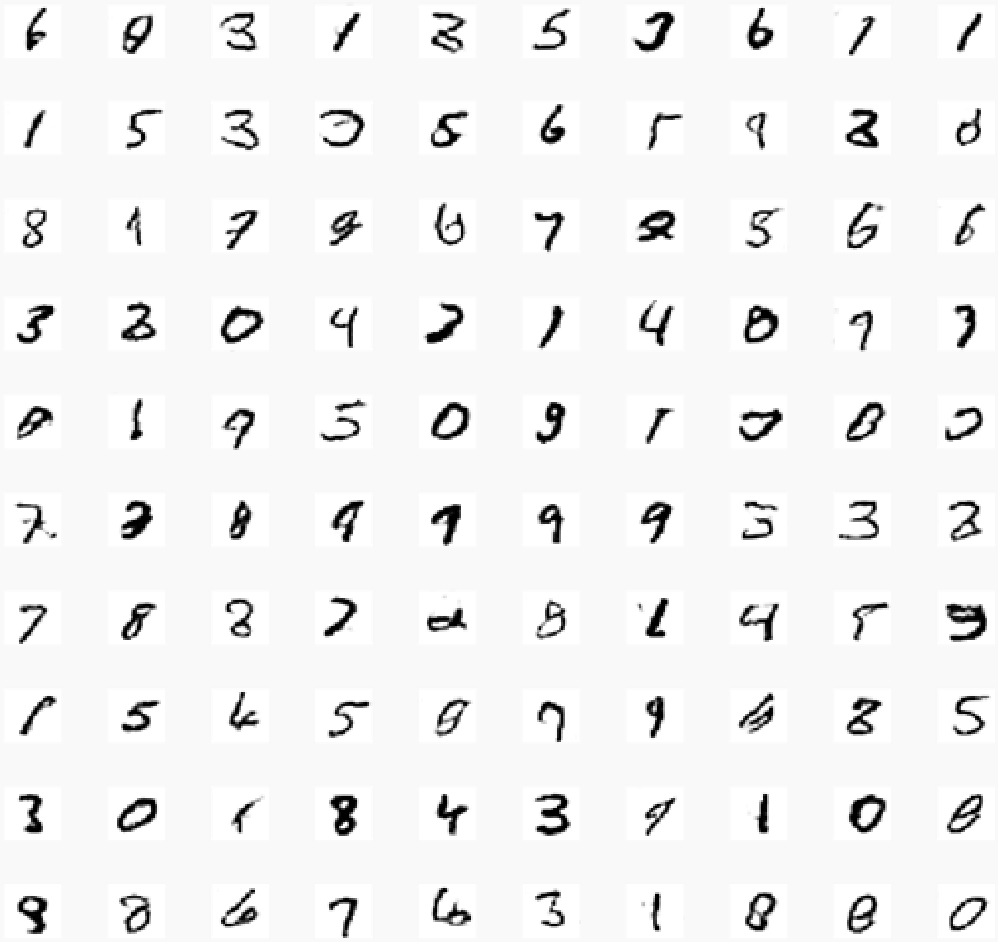
DCGAN Generator



- Eliminate fully connected layers.
- Replace all max pooling with convolutional stride.
- Use **transposed convolution** for upsampling or simple upsampling.
- Use Batch normalization.

[Source]

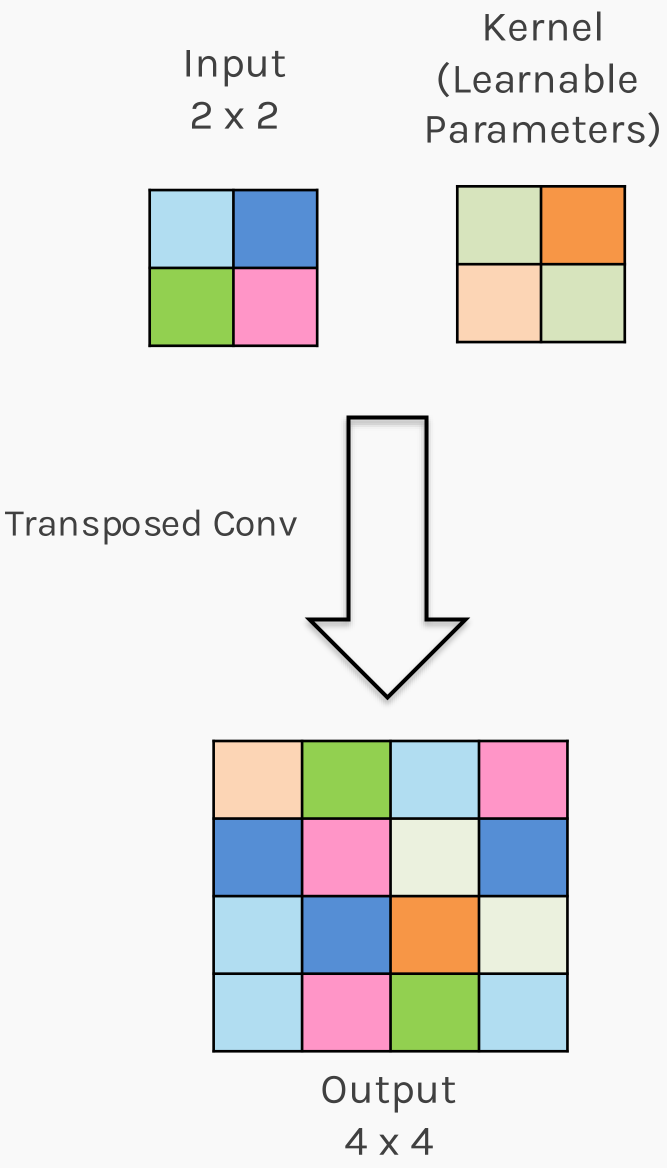
DCGAN on MNIST



Optional: Transposed Convolution

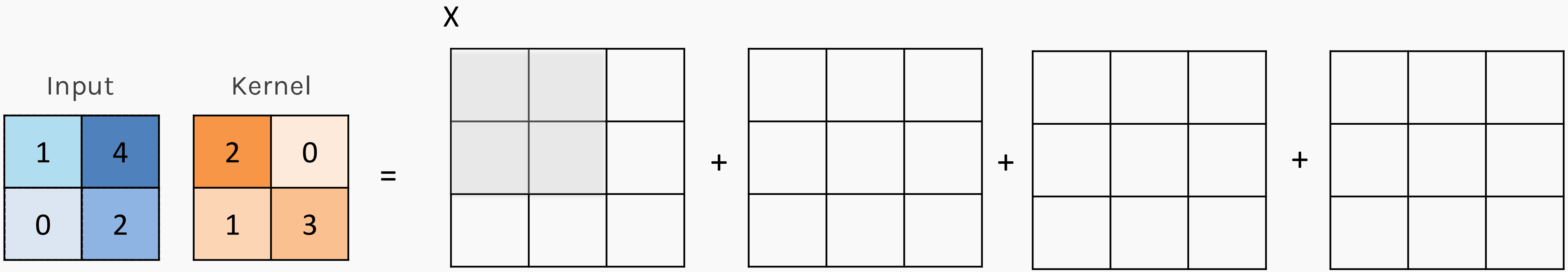
What is transposed convolution?

Transposed Convolution is used to upsample an image by **learning kernel parameters** unlike other up sampling techniques such as nearest neighbor or bilinear interpolation.



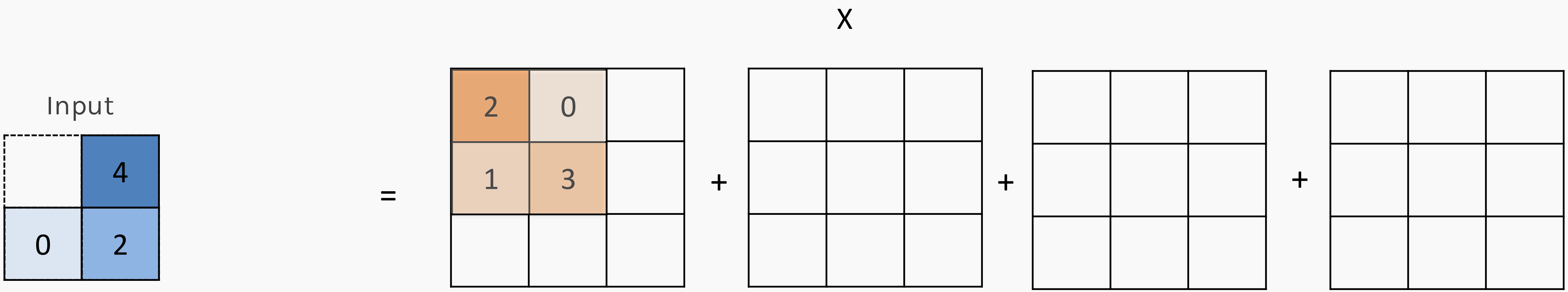
What is transposed convolution?

Assume we have a 2x2 input that needs to be up-sampled to a 3x3 output using transposed convolution with stride 1.



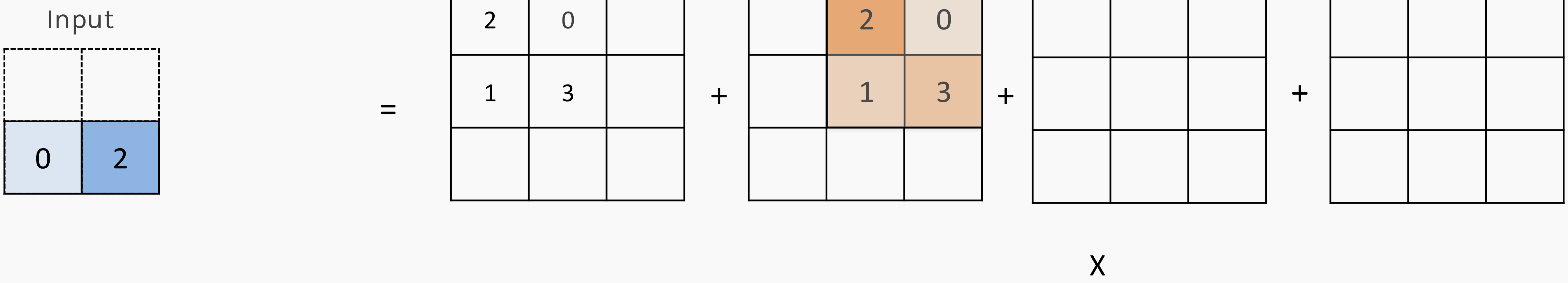
What is transposed convolution?

Assume we have a 2x2 input that needs to be up-sampled to a 3x3 output using transposed convolution with stride 1.



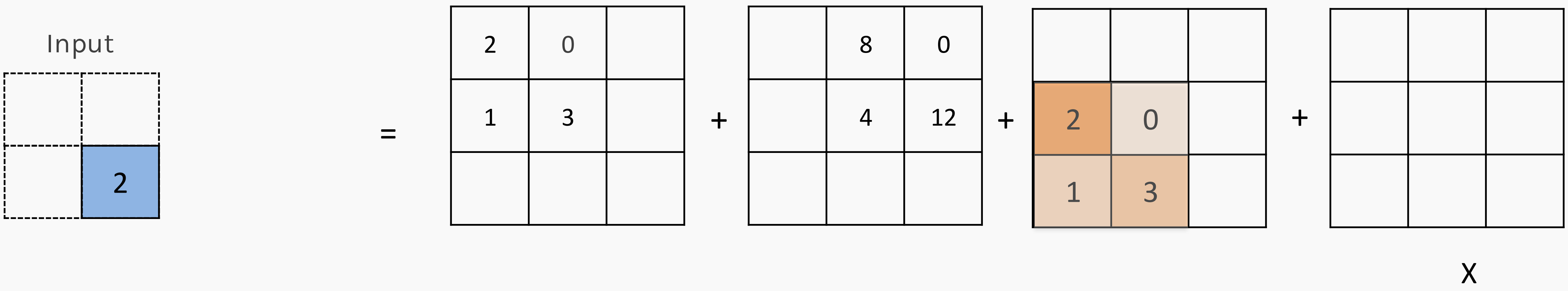
What is transposed convolution?

Assume we have a 2x2 input that needs to be up-sampled to a 3x3 output using transposed convolution with stride 1.



What is transposed convolution?

Assume we have a 2x2 input that needs to be up-sampled to a 3x3 output using transposed convolution with stride 1.



What is transposed convolution?

Assume we have a 2x2 input that needs to be up-sampled to a 3x3 output using transposed convolution with stride 1.

Input

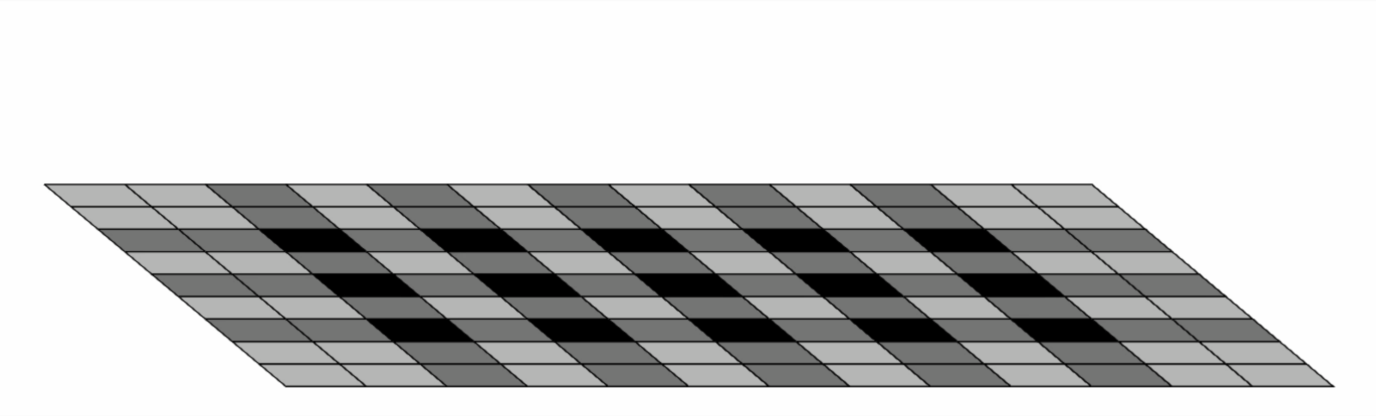
$$\begin{bmatrix} \; & \; \\ \; & \; \end{bmatrix} \qquad = \begin{bmatrix} 2 & 0 & \\ 1 & 3 & \\ & & \end{bmatrix} + \begin{bmatrix} & 8 & 0 \\ & 4 & 12 \\ & & \end{bmatrix} + \begin{bmatrix} & & \\ 0 & 0 & \\ 0 & 0 & \end{bmatrix} + \begin{bmatrix} & & \\ & 4 & 0 \\ & 2 & 6 \end{bmatrix}$$

What is transposed convolution?

$$
\begin{array}{|c|c|c|}\hline 2 & 0 & \\ \hline 1 & 3 & \\ \hline & & \\ \hline\end{array}
+
\begin{array}{|c|c|c|}\hline & 8 & 0 \\ \hline & 4 & 12 \\ \hline & & \\ \hline\end{array}
+
\begin{array}{|c|c|c|}\hline & & \\ \hline 0 & 0 & \\ \hline 0 & 0 & \\ \hline\end{array}
+
\begin{array}{|c|c|c|}\hline & & \\ \hline & 4 & 0 \\ \hline & 2 & 6 \\ \hline\end{array}
=
\begin{array}{|c|c|c|}\hline 2 & 8 & 0 \\ \hline 1 & 11 & 12 \\ \hline 0 & 2 & 6 \\ \hline\end{array}
$$

Checkerboard Artifact

Transposed Convolution can easily have “uneven overlap,” putting more emphasis in some places than others.



[Source]

There are few ways we can avoid this issue:

1. Choose a kernel size that is divisible by your stride, avoiding the overlap issue.
2. Separate out up-sampling to a higher resolution from convolution to compute features.

Thank you