M.Tech Program

Advanced Industry Integrated Programs

Jointly offered by University and LTIMindTree

Generative AI Fundamentals: Understanding Generative Adversarial Network

Knowledge partner

LTIMindtree

Implementation partner

L&T EduTech

Course Objective:

- Build and understand basic GAN architectures using PyTorch.
- Develop and tune advanced GANs like DCGANs and WGANs.
- Evaluate GAN performance and understand biases.
- Explore advancements in GANs, focusing on StyleGAN.
- Apply GANs for data augmentation and image translation.

Modules to cover….

1. Foundations of Generative Adversarial Networks (GANs)
2. Build Better Generative Adversarial Networks (GANs)
3. Apply Generative Adversarial Networks (GANs)

Foundations of Generative Adversarial Networks (GANs)

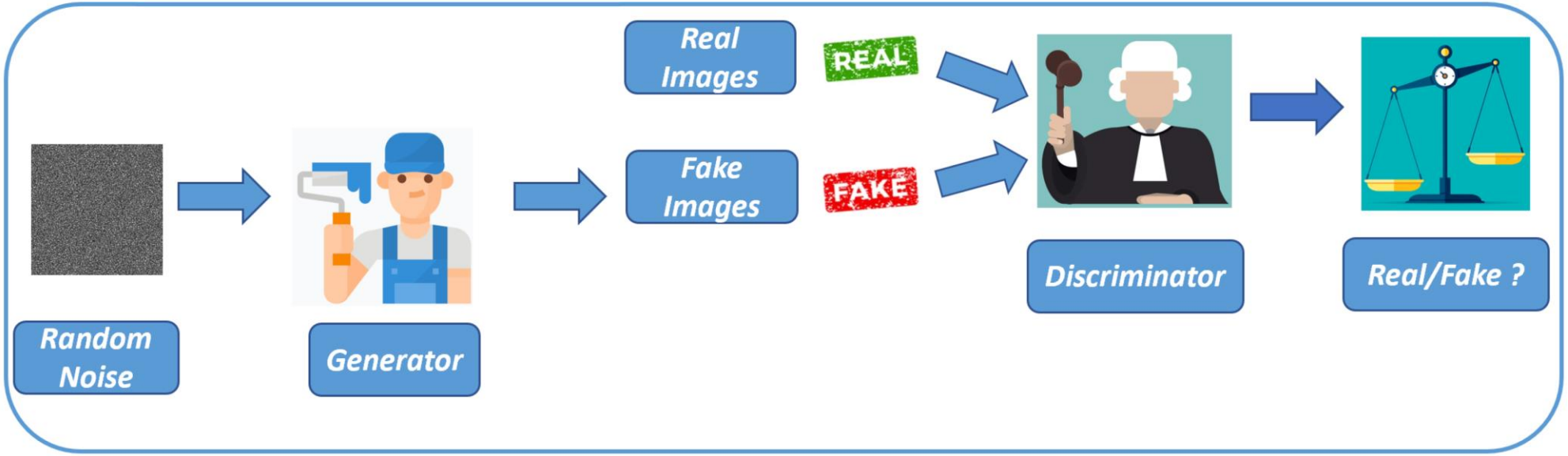
Foundations of GAN - *Learning Outcomes..*

- Understand the core concepts and components of GANs.
- Build and train a simple GAN using PyTorch.
- Construct and enhance Deep Convolutional GANs (DCGANs).
- Implement WGANs to improve training stability
- Create and control GANs for specific image generation tasks.

Foundations of GAN

Introduction to GAN

- Comprise a generator and a discriminator
- Generator creates images from random noise
- Discriminator differentiates between real and fake images
- Models compete and learn from each other (adversarial training)

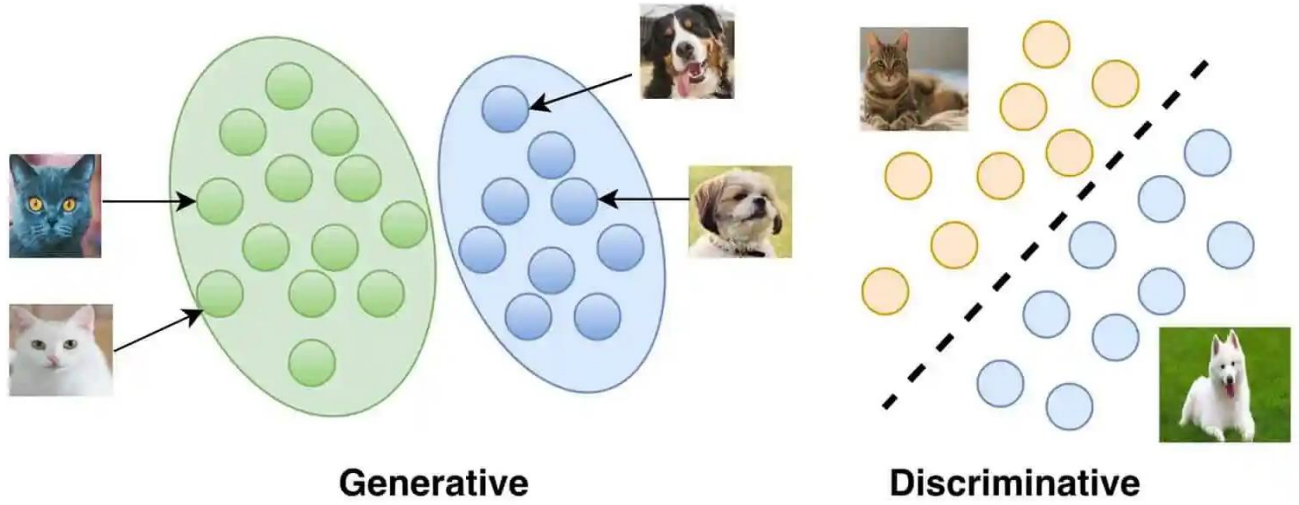


Foundations of GAN

Generative vs. Discriminative Models

Discriminative Models:

- Used for classification (e.g., dogs vs. cats)
- Model probability of class (Y) given features (X)
- Example: Classifier determining dog vs. cat based on features



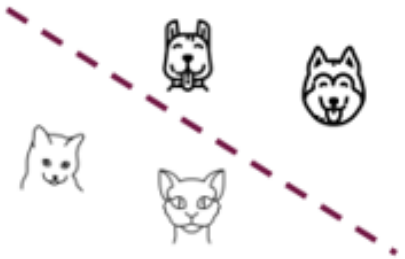
Generative Models:

- Learn to generate realistic examples of a class
- Use random noise to create diverse outputs
- Example: Generating various images of dogs

Foundations of GAN

Generative vs. Discriminative Models

Discriminative models



Features Class

$$X \rightarrow Y$$

$$P(Y|X)$$

Generative models

Noise Class Features

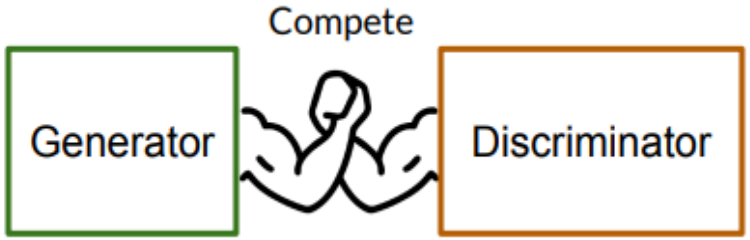
$$\xi, Y \rightarrow X$$

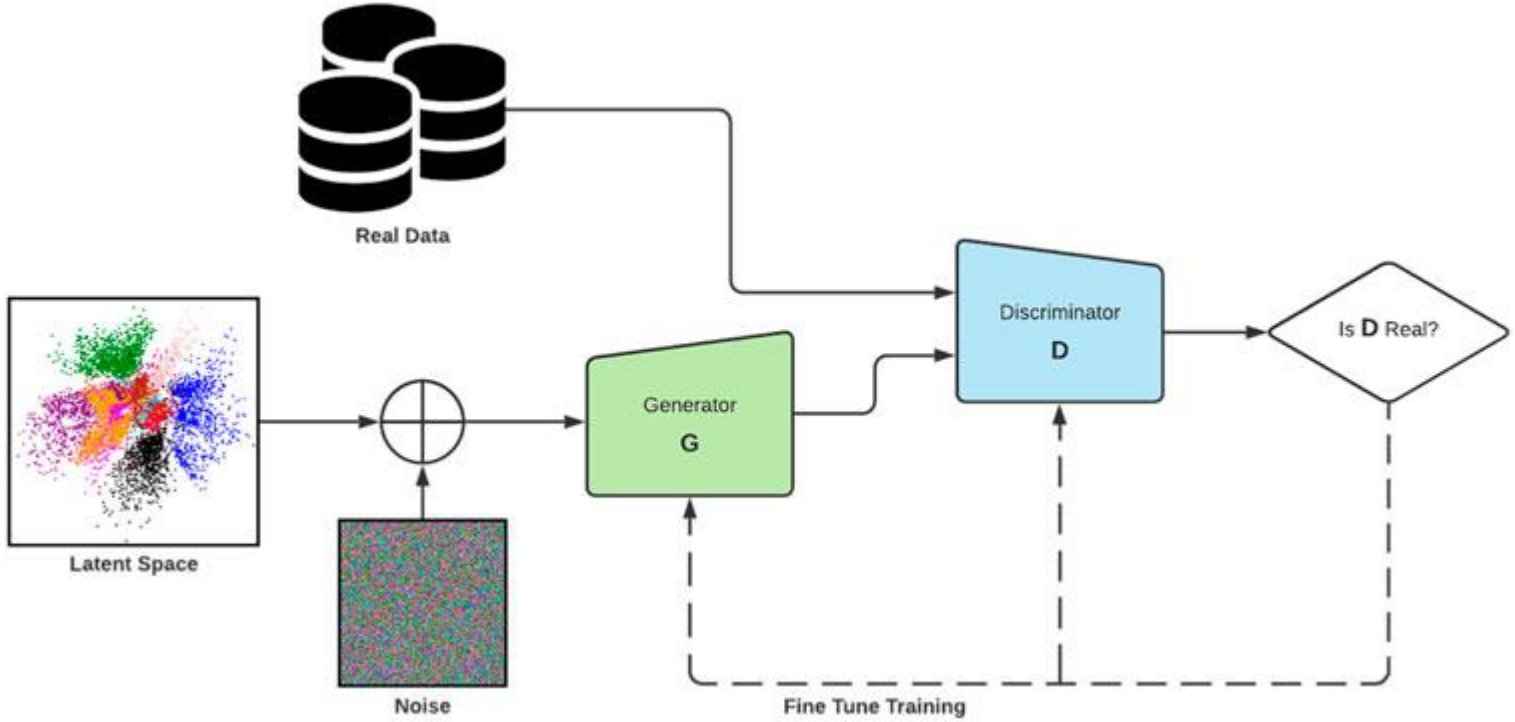
$$P(X|Y)$$

Foundations of GAN

How GANs Work

- Generator takes random noise input to produce images
- Discriminator evaluates images as real or fake
- Adversarial process improves both models over time





Foundations of GAN

Industry Applications of GANs

Adobe: Next-gen Photoshop for novice artists

Google: Text and image generation

IBM: Data augmentation for improved datasets

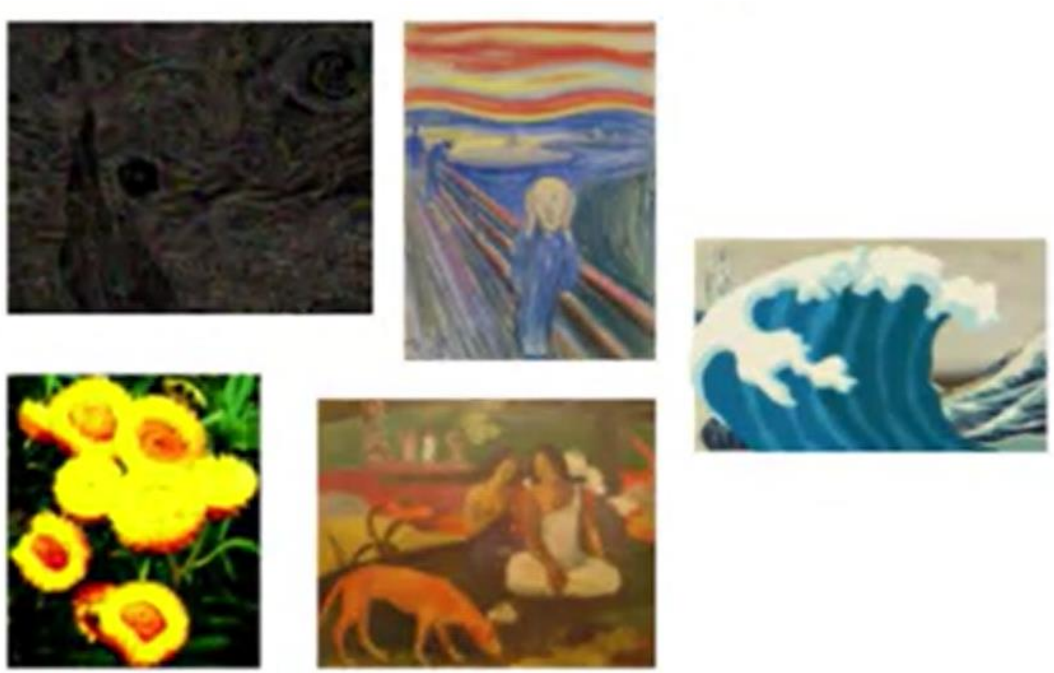
Snapchat & TikTok: Creative filters

Disney: Super-resolution for high-quality images

Foundations of GAN

Intuition Behind GANs

Generator learns to make **fakes** that look **real**

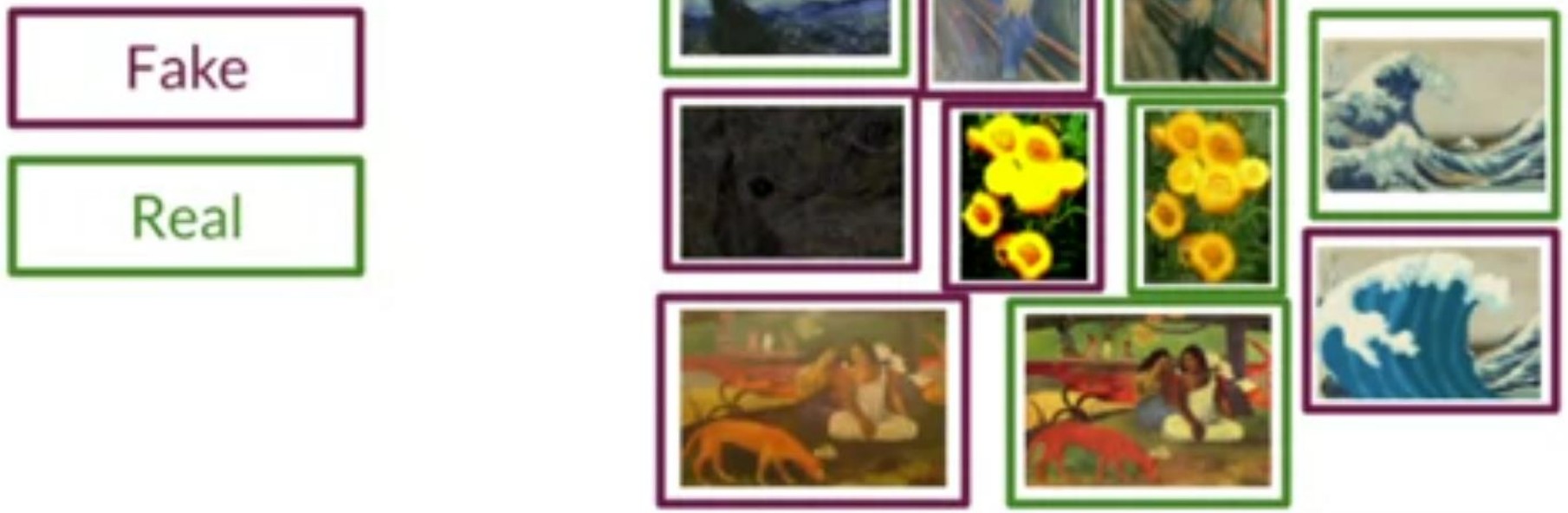


Discriminator learns to distinguish **real** from **fake**

Foundations of GAN

Intuition Behind GANs

Discriminator learns to distinguish real from fake



Foundations of GAN

Intuition Behind GANs

Discriminator learns to distinguish real from fake

5% Real

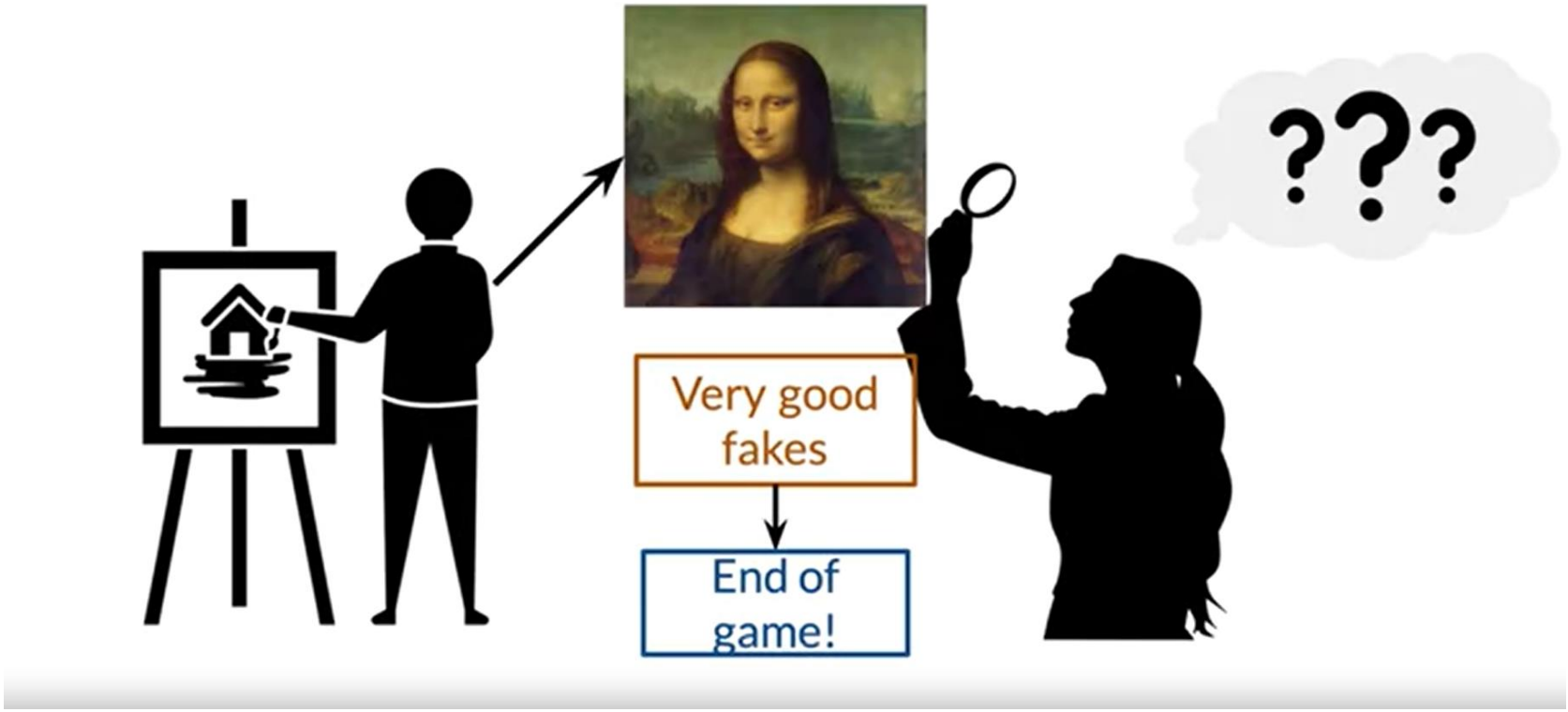
40% Real

80% Real

Foundations of GAN

Discriminator

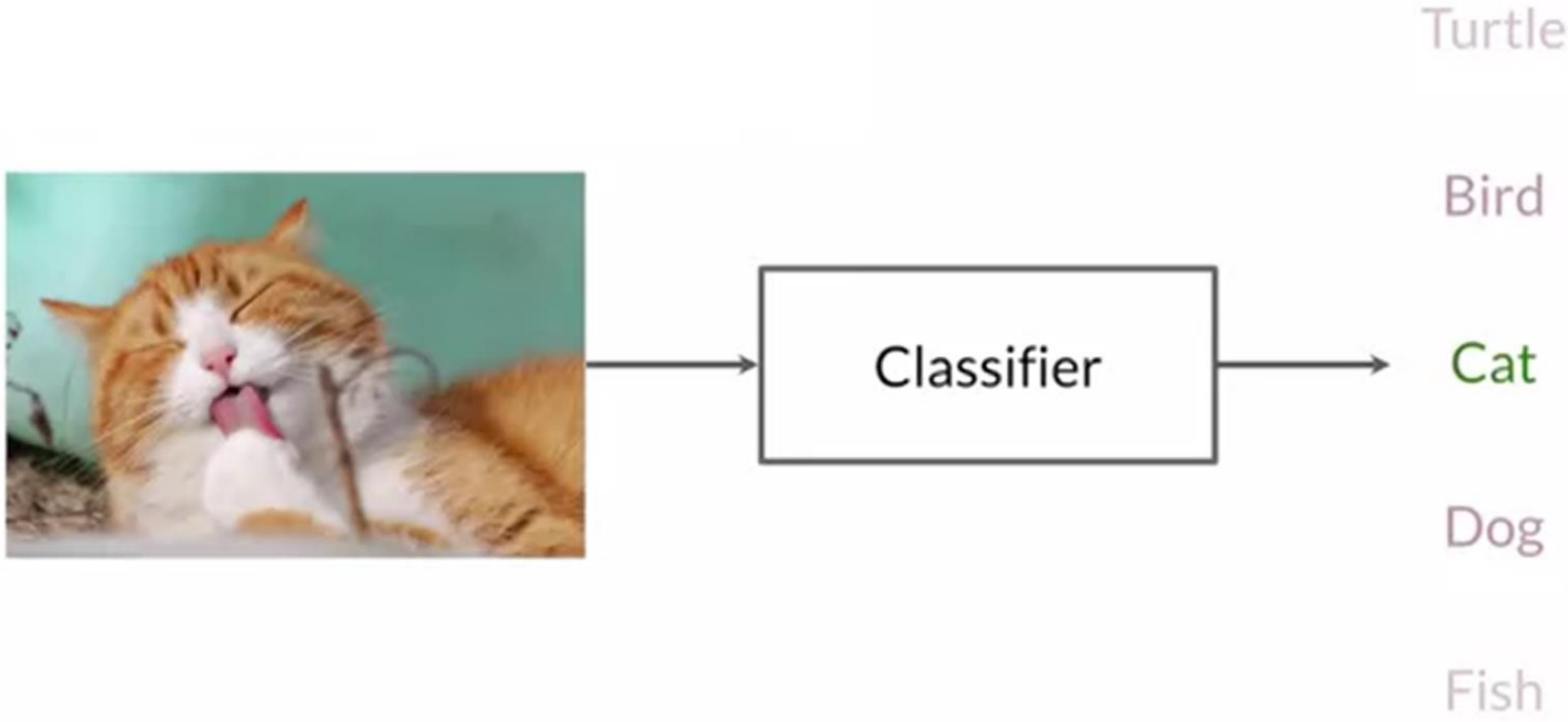
Game Start to End



Foundations of GAN

Discriminator

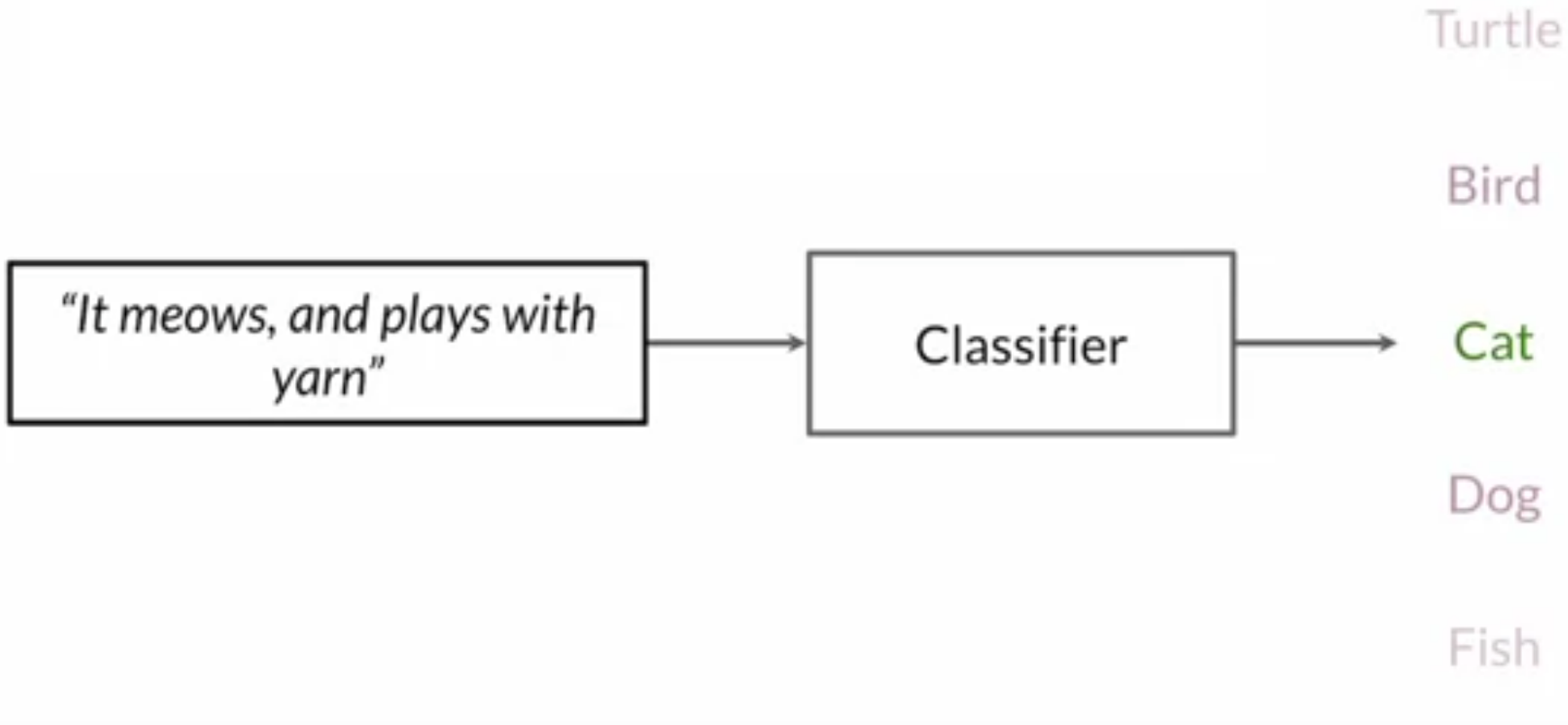
Classifier :
Distinguish between different classes



Foundations of GAN

Discriminator

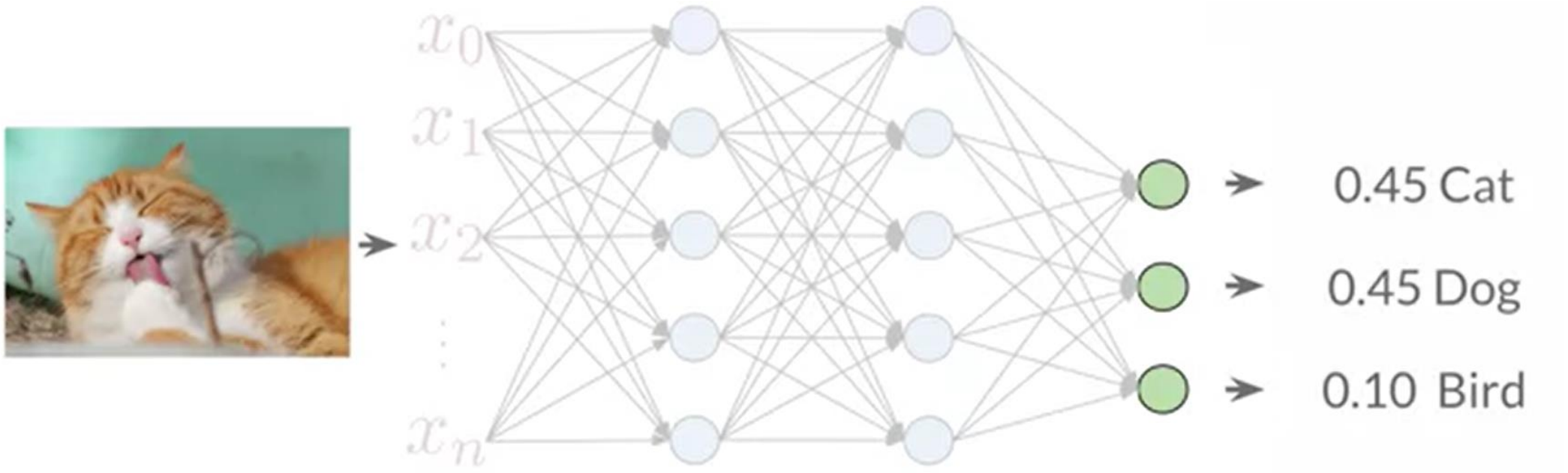
Classifier :
Distinguish between different classes



Foundations of GAN

Discriminator

Classifier :
Distinguish between different classes – Neural Networks Representation

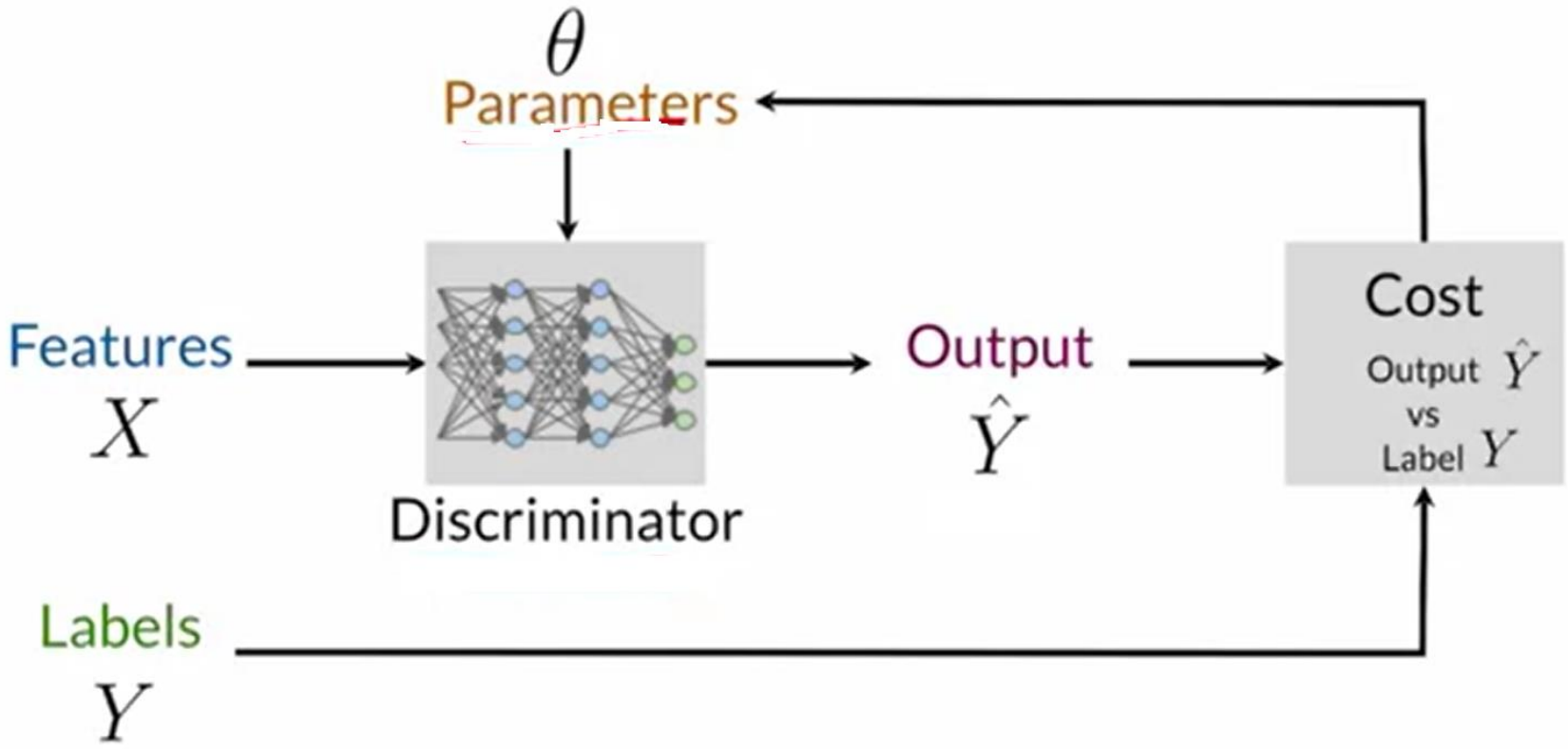


Foundations of GAN

Discriminator

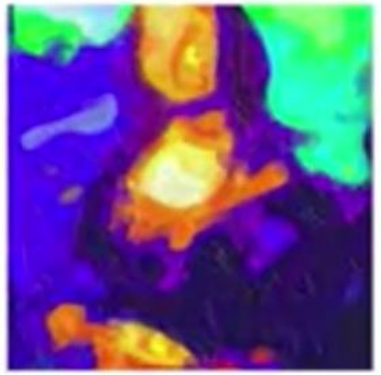
Classifier : Training

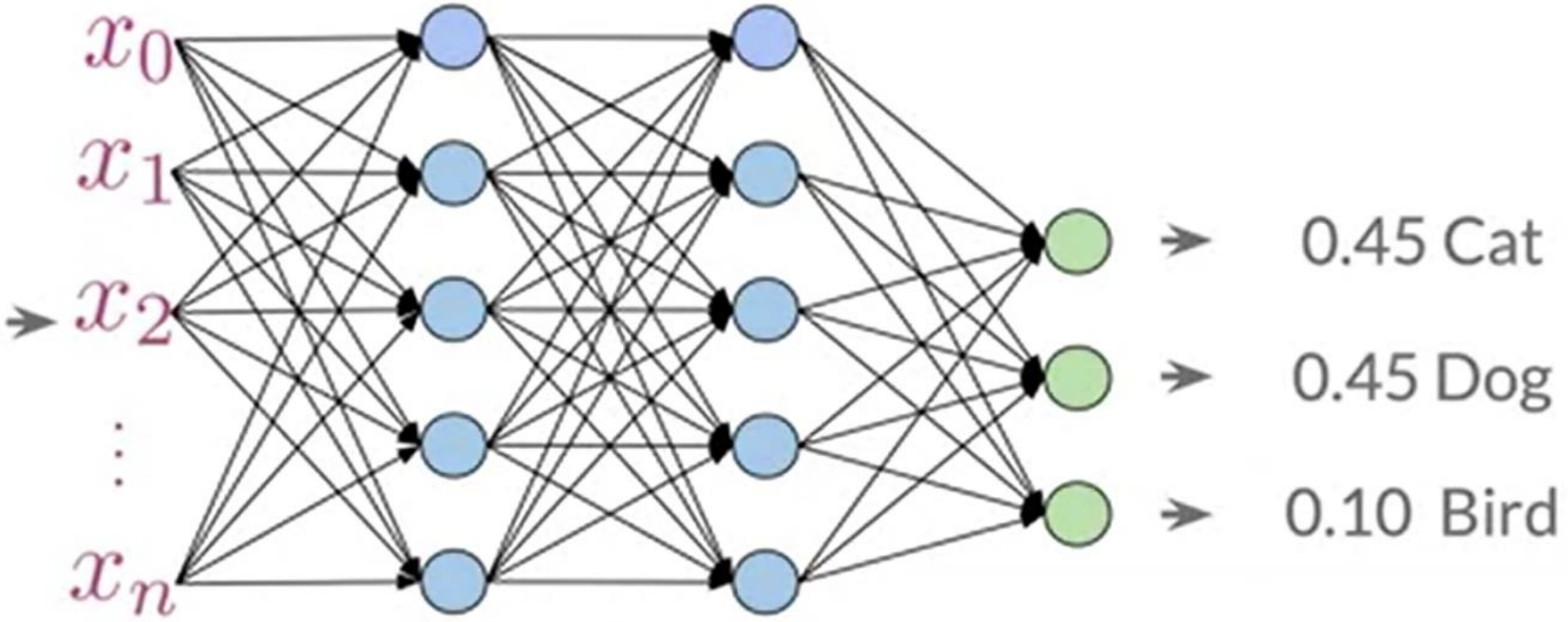
Distinguish between different classes – Neural Networks



Foundations of GAN

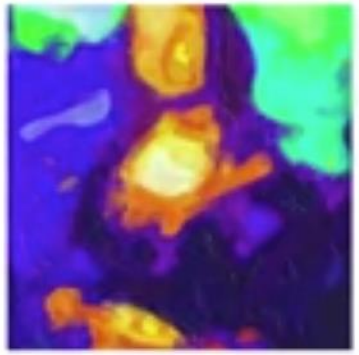
Discriminator

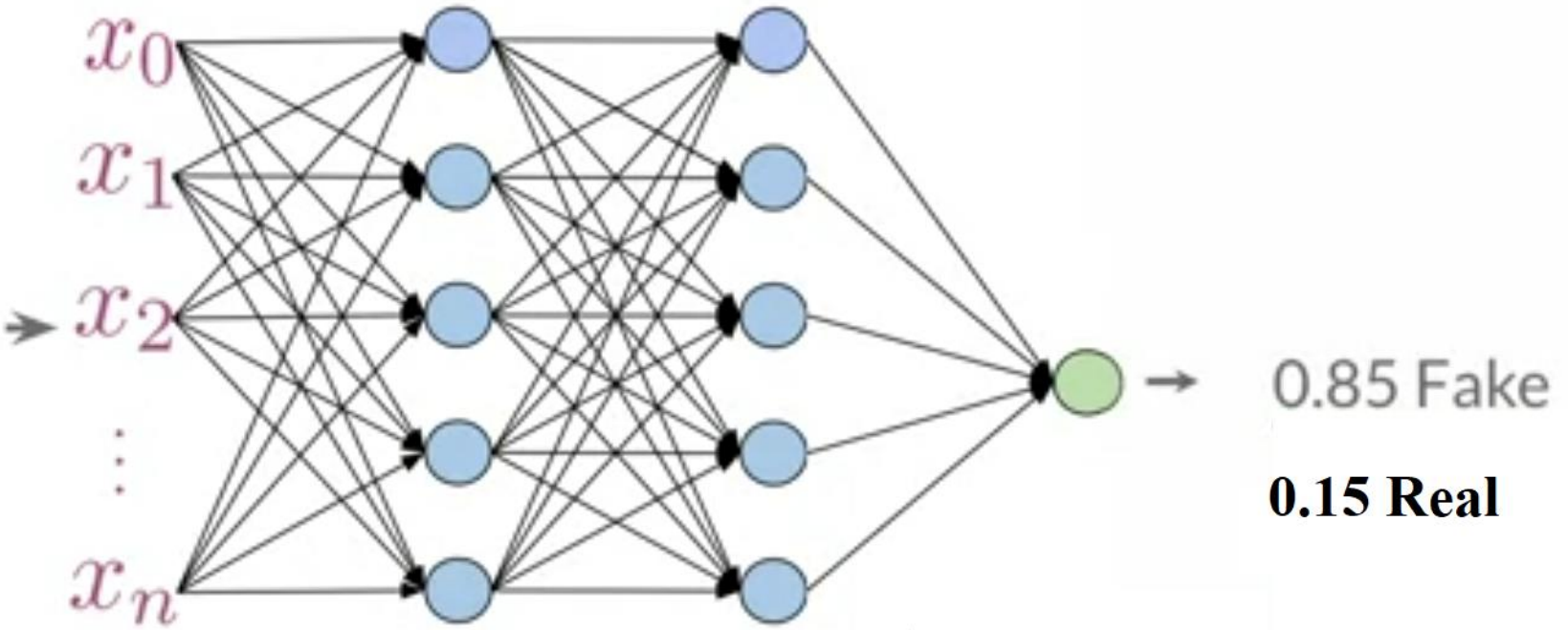




Foundations of GAN

Discriminator

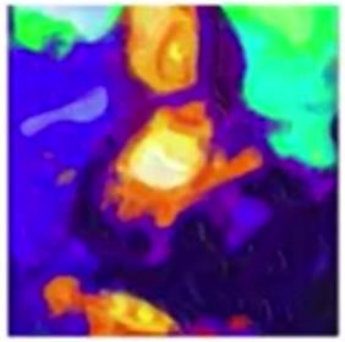




Foundations of GAN

Discriminator

- The discriminator is a classifier
- It learns the probability of class Y (real or fake) given features X
- The probabilities are the feedback for the generator

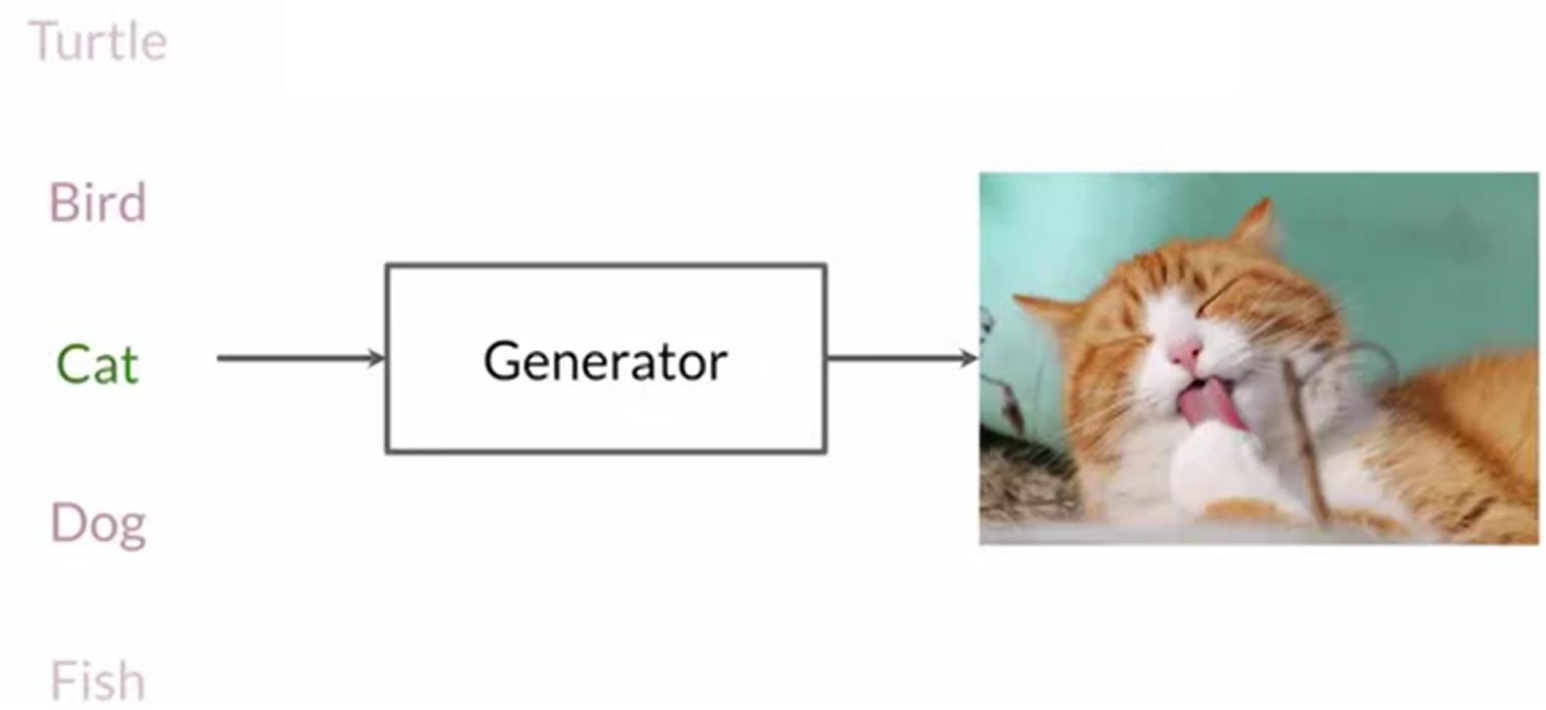
$$P(\underbrace{\text{Fake}}_{\text{Class}} \mid \underbrace{\text{  }}_{\text{Features}}) = 0.85 \longrightarrow \boxed{\text{Fake}}$$

Probabilities are feedback for the generator

Foundations of GAN

Generator

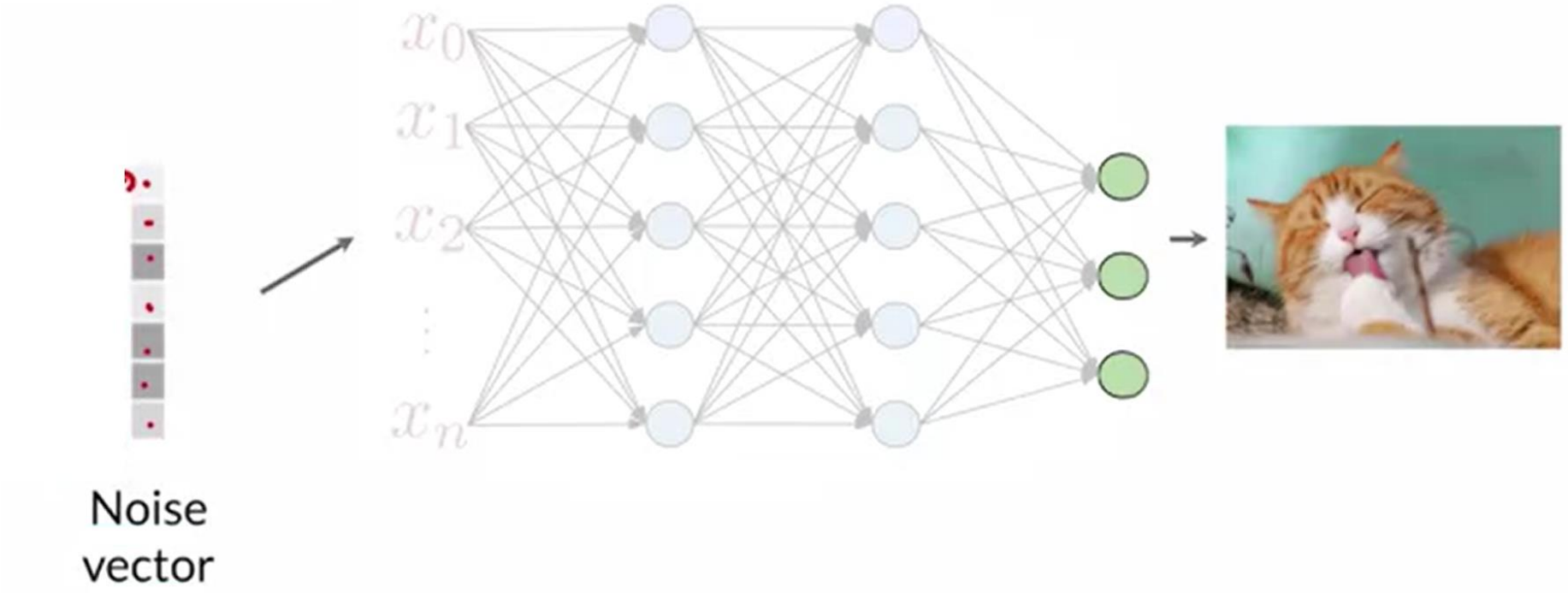
- The generator is the core component in a GAN, responsible for generating realistic examples.
- It aims to produce diverse outputs that belong to a specific class (e.g., images of cats).



Foundations of GAN

Generator – Noise Vector

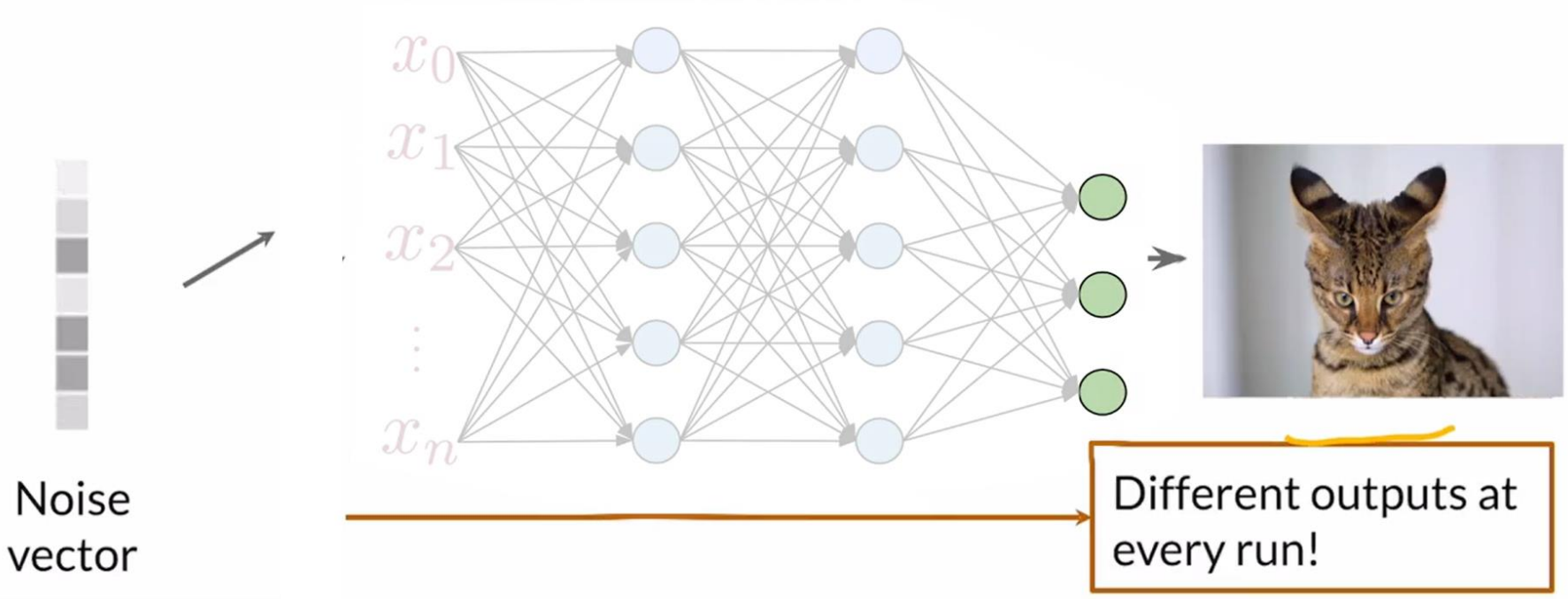
- To generate different examples each time, the generator takes in random values (noise vector) as input.
- The noise vector combined with the class label (if applicable) is processed by the generator's neural network.



Foundations of GAN

Generator – Noise Vector

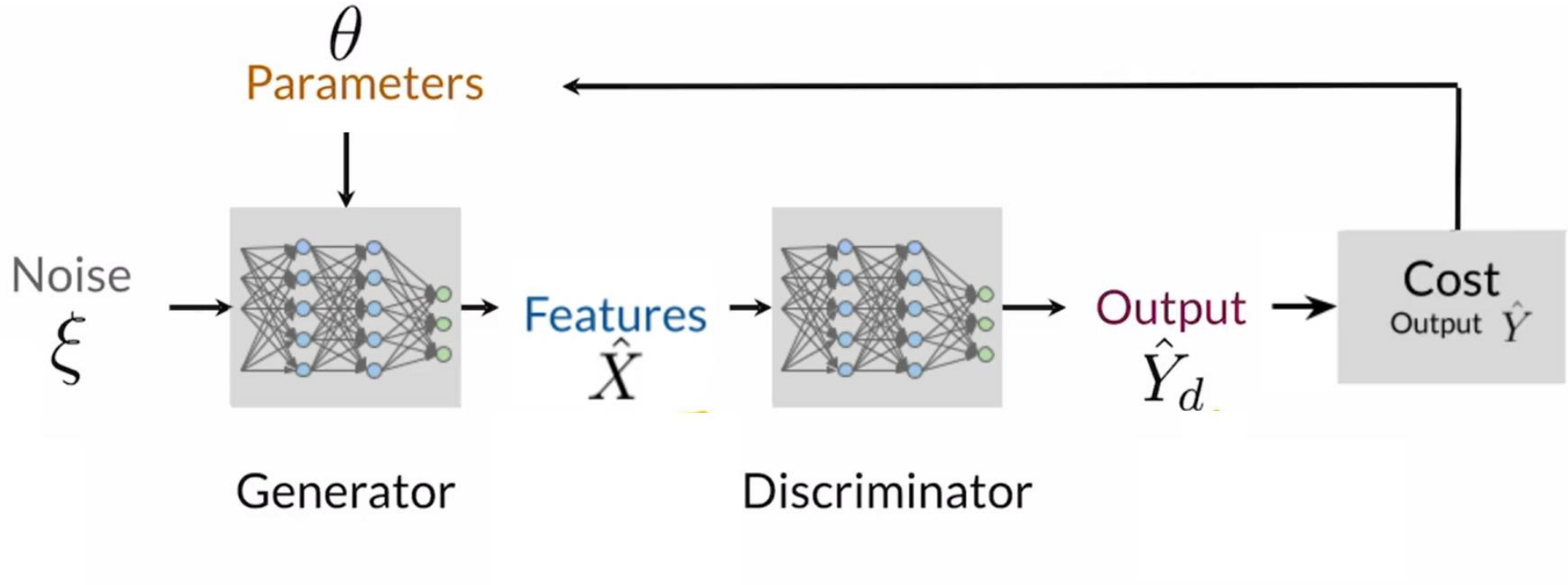
- The generator's neural network computes a series of nonlinear transformations on the input to produce an output image.
- Each pixel in the output image is determined by the network, resulting in diverse representations (e.g., different cat breeds).



Foundations of GAN

Generator – Learning

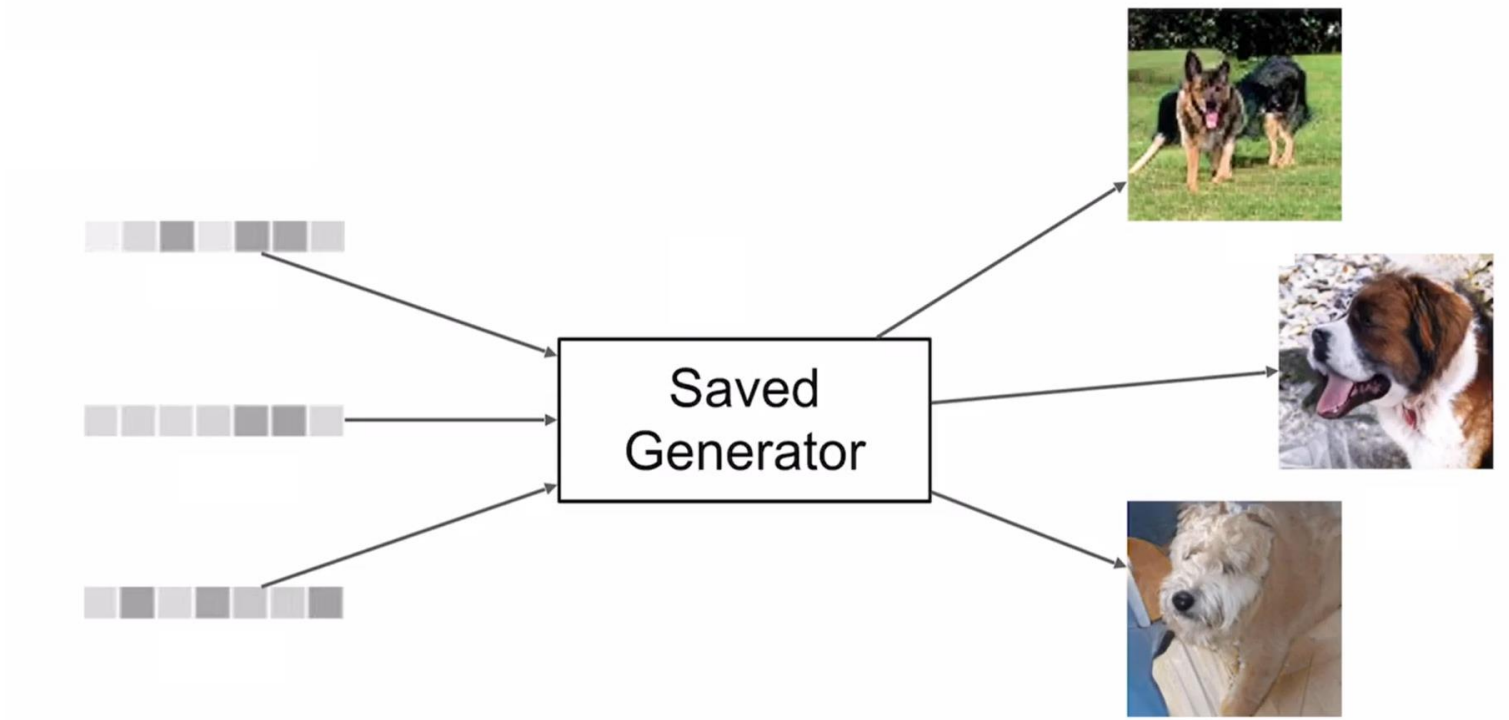
- The goal is for the generator's output to be classified as real by the discriminator.
- The cost function measures how close the generator's output is to being real, guiding the parameter updates in the generator's network



Foundations of GAN

Generator - Sampling

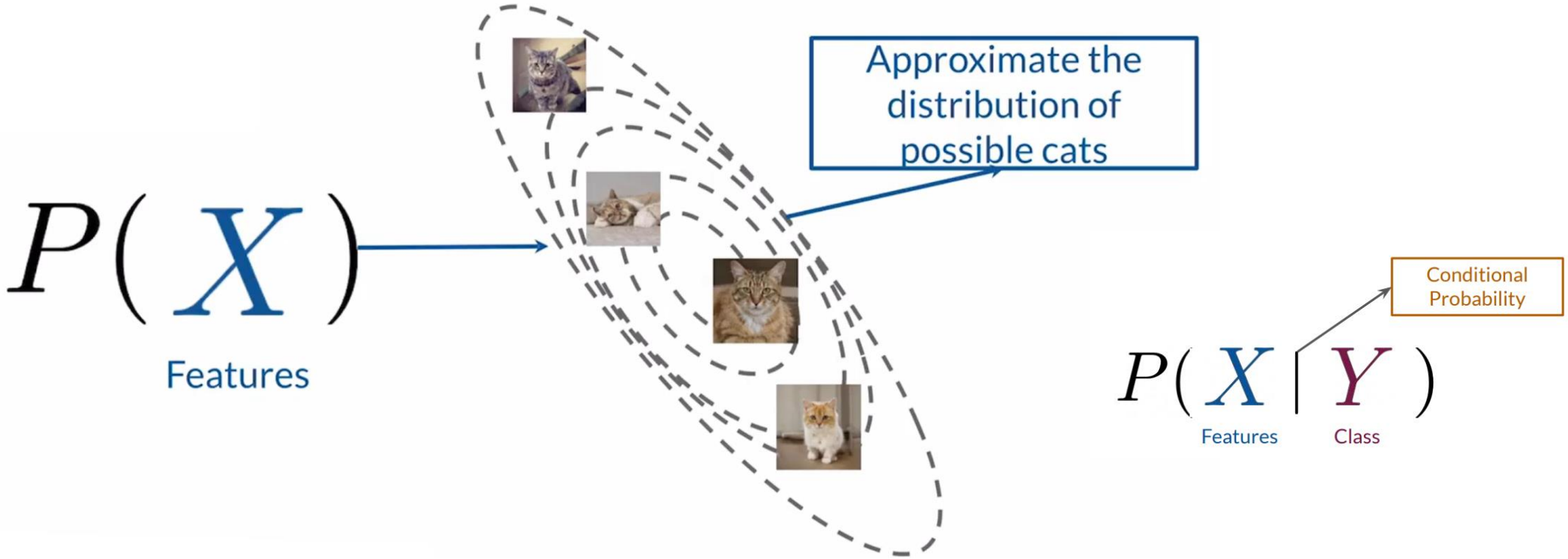
- Once the generator is well-trained, its parameters are saved.
- New noise vectors can be fed into the saved generator to produce various outputs (e.g., different dog images if trained on dogs).



Foundations of GAN

Generator - Probability Modeling

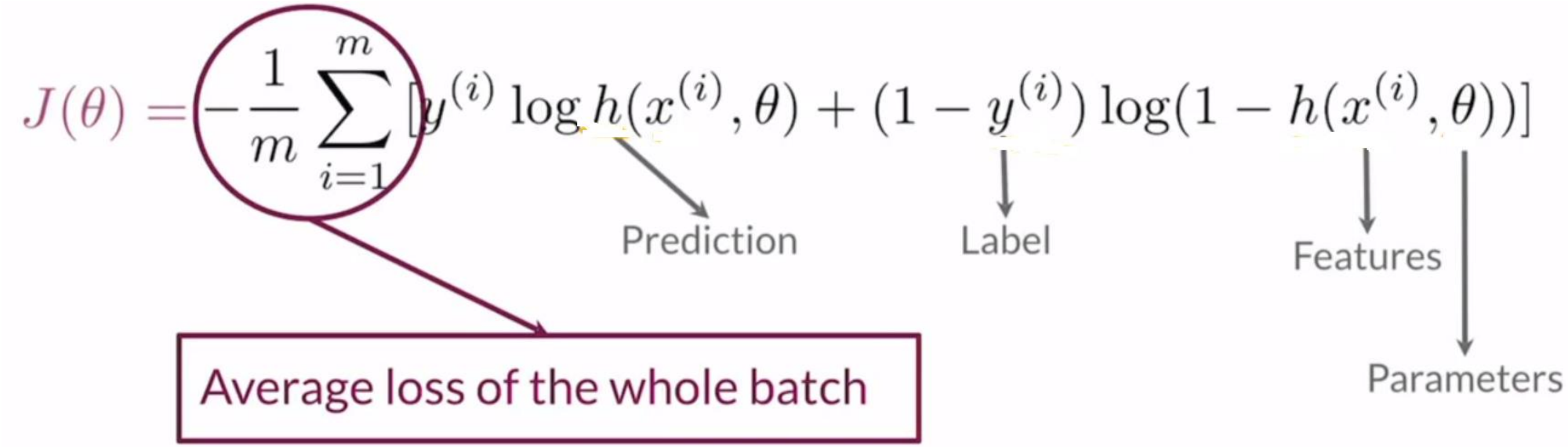
- The generator models the probability distribution of features (X) given a class (Y).
- If only one class is used (e.g., cats), it models the probability distribution of cat features without additional conditions



Foundations of GAN

BCE Cost Function -Purpose of BCE

- BCE is designed for classification tasks with two categories: real and fake.
- It is crucial for training GANs to distinguish between real and fake examples.



Foundations of GAN

BCE Cost Function -Components of the BCE Loss Function

- The loss function has two terms:One for when the true label is 1 (real)
- One for when the true label is 0 (fake)

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log h(x^{(i)},\theta) + (1-y^{(i)})\log(1-h(x^{(i)},\theta))\right]$$

$y^{(i)}$	$h(x^{(i)},\theta)$	$y^{(i)}\log h(x^{(i)},\theta)$
0	any	0
1	0.99	~0
1	~0	-inf

Relevant when the label is 1

Foundations of GAN

BCE Cost Function -Components of the BCE Loss Function

- For a true label of 1 (real), if the model's prediction is far from 1, the loss significantly increases due to the logarithm term.
- Similarly, for a true label of 0 (fake), if the prediction is far from 0, the loss also increases significantly.

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log h(x^{(i)},\theta) + (1-y^{(i)})\log(1-h(x^{(i)},\theta))\right]$$

$y^{(i)}$	$h(x^{(i)},\theta)$	$(1-y^{(i)})\log(1-h(x^{(i)},\theta))$
1	any	0
0	0.01	~0
0	~1	-inf

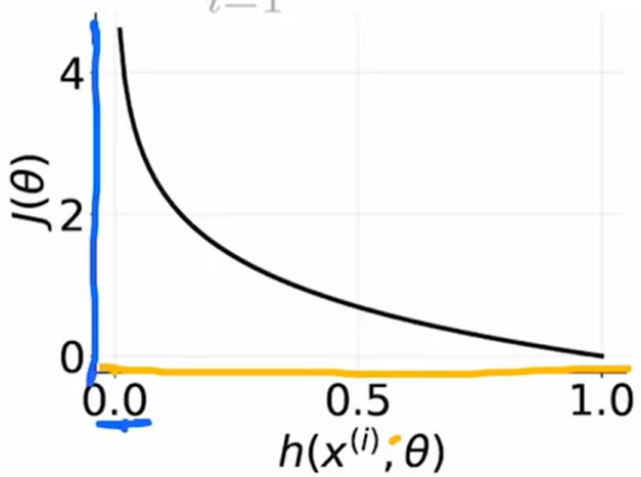
Relevant when the label is 0

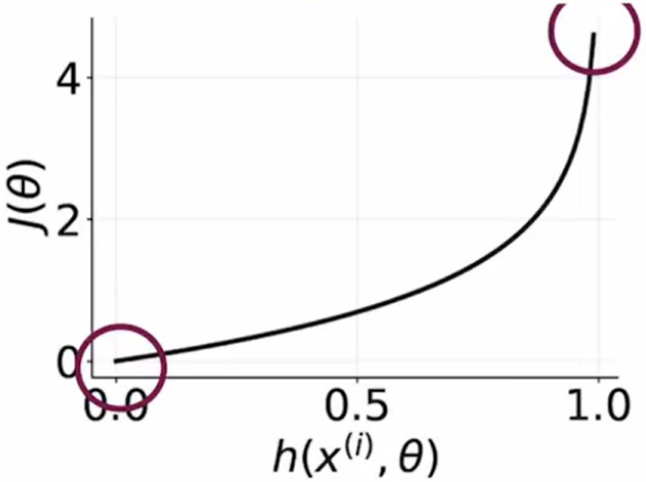
Foundations of GAN

BCE Cost Function

- Plotting the BCE loss function shows that the loss is minimal when predictions are correct and maximal when predictions are incorrect.
- For real labels (1), the loss is low when predictions are close to 1.
- For fake labels (0), the loss is low when predictions are close to 0

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log h(x^{(i)},\theta) + (1-y^{(i)})\log(1-h(x^{(i)},\theta))]$$





Foundations of GAN

BCE Cost Function -Practical Application

BCE Cost Function

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log h(x^{(i)},\theta) + (1-y^{(i)})\log(1-h(x^{(i)},\theta))\right]$$

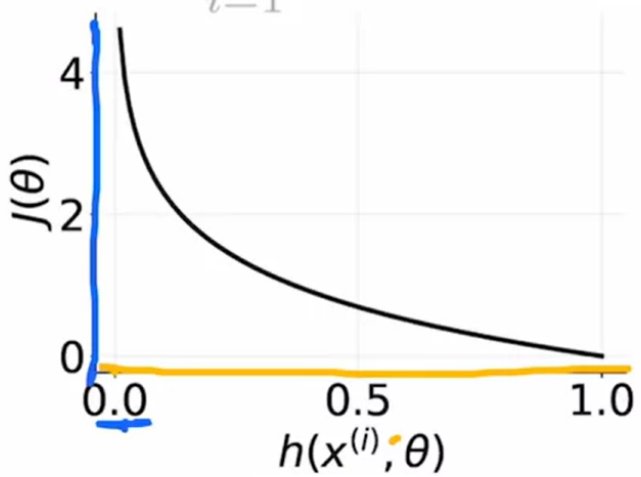
Ensures that the cost is always greater or equal to 0

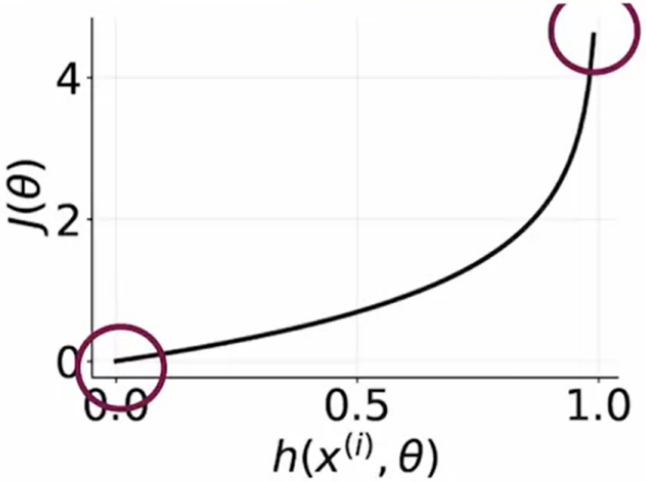
Foundations of GAN

BCE Cost Function -Practical Application

- Understanding the BCE loss function is crucial for tuning GANs to achieve better performance.
- The goal is to minimize the BCE loss by improving the generator's ability to create realistic data and the discriminator's accuracy in classifying them.

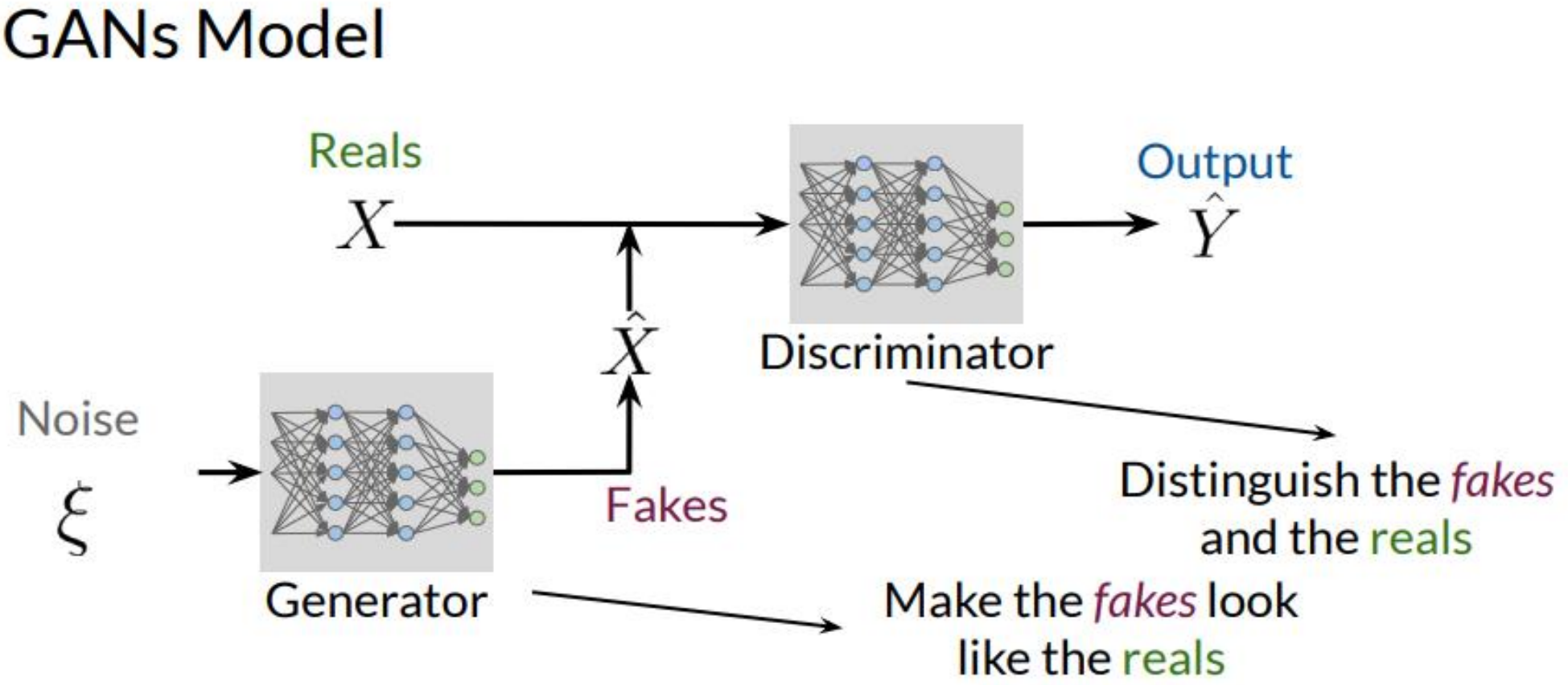
$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log h(x^{(i)},\theta) + (1-y^{(i)})\log(1-h(x^{(i)},\theta))]$$



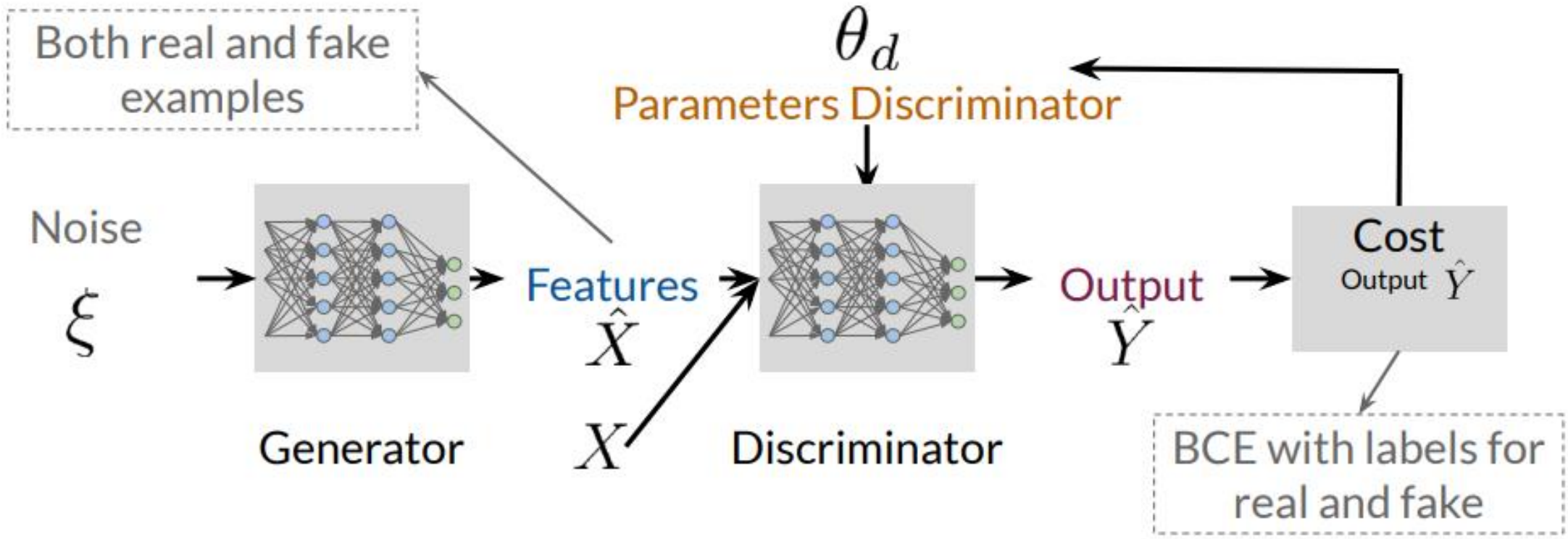


How to Train GANs

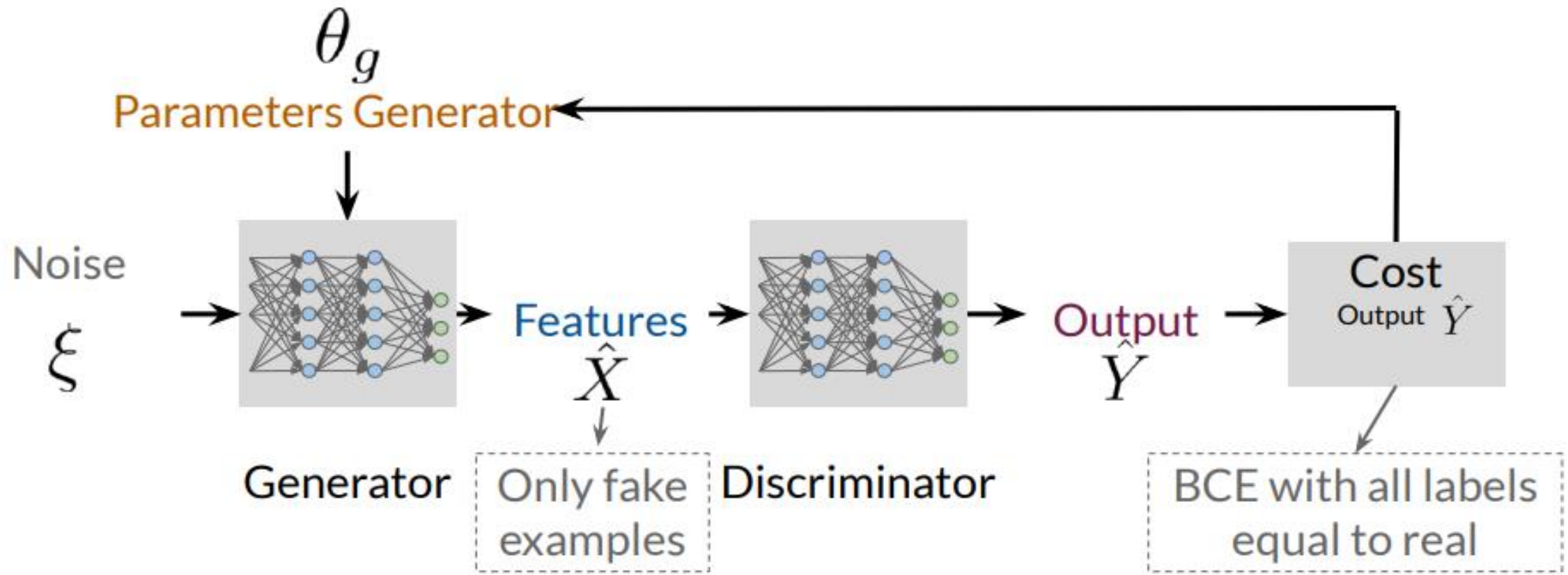
Putting it all together



- **Training GANs : Discriminator**



- **Training GANs : Generator**



Foundations of GAN

DEMO : Build your very own GAN using PyTorch

•Implementing a GAN using PyTorch , with a focus on generating synthetic hand-written digits from the MNIST dataset.

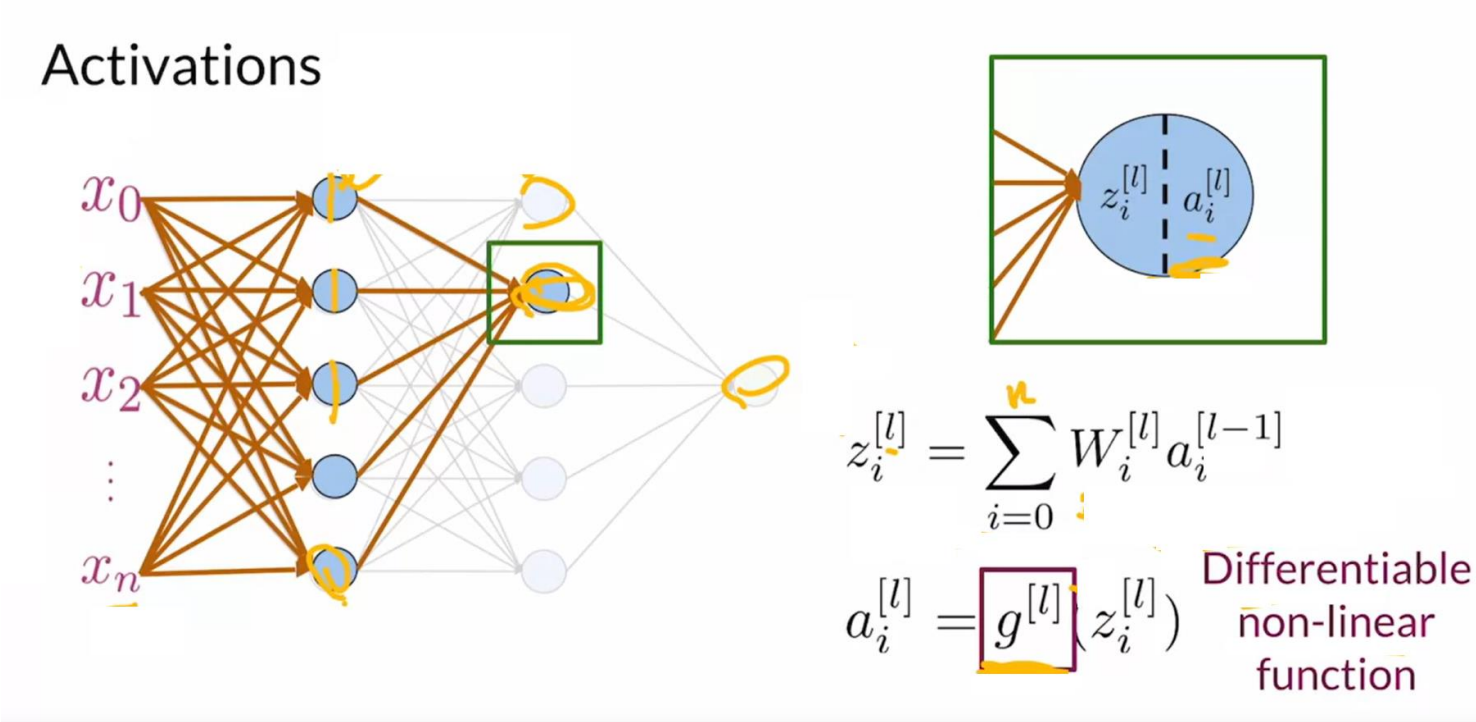
PRACTICAL
DEMONSTRATION

Foundations of GAN

Deep Convolutional GAN - Activation Functions

- **Activation functions** are the mathematical engines that power complex computations in neural networks, including GANs.

- They process information and introduce non-linearity, enabling networks to perform tasks like classification.



Foundations of GAN

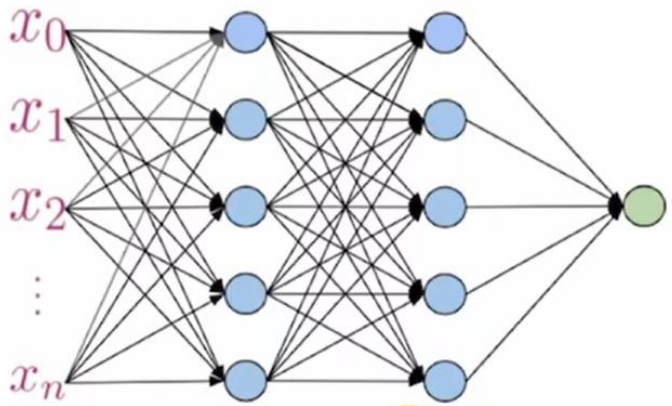
Deep Convolutional GAN - Activation Functions

Activations

$$a_i^{[l]} = g^{[l]}(z_i^{[l]})$$

Differentiable non-linear function

1. Differentiable for backpropagation
2. Non-linear to compute complex features, **if not**:



$\equiv$

$$WX + b$$

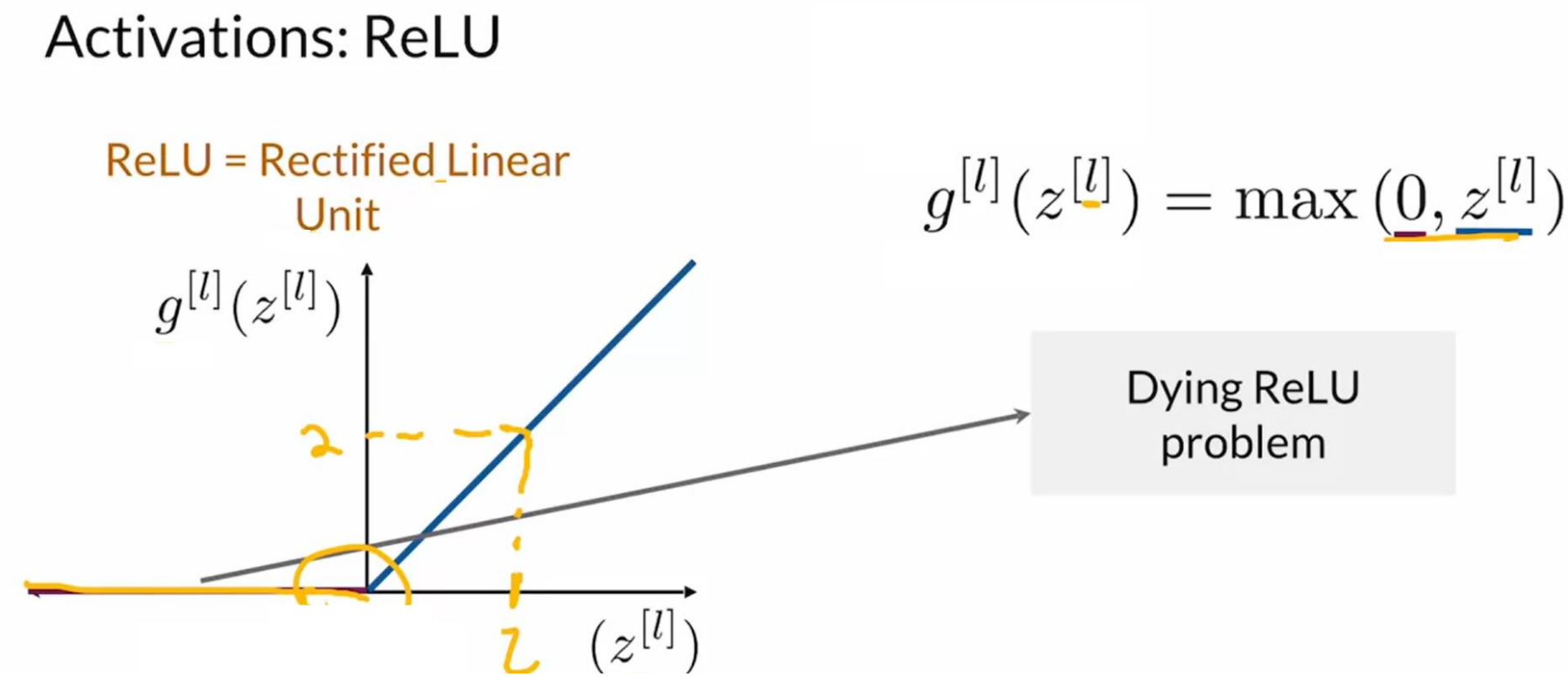
Linear regression

Foundations of GAN

Deep Convolutional GAN - Activation Functions

Common activations and their structure

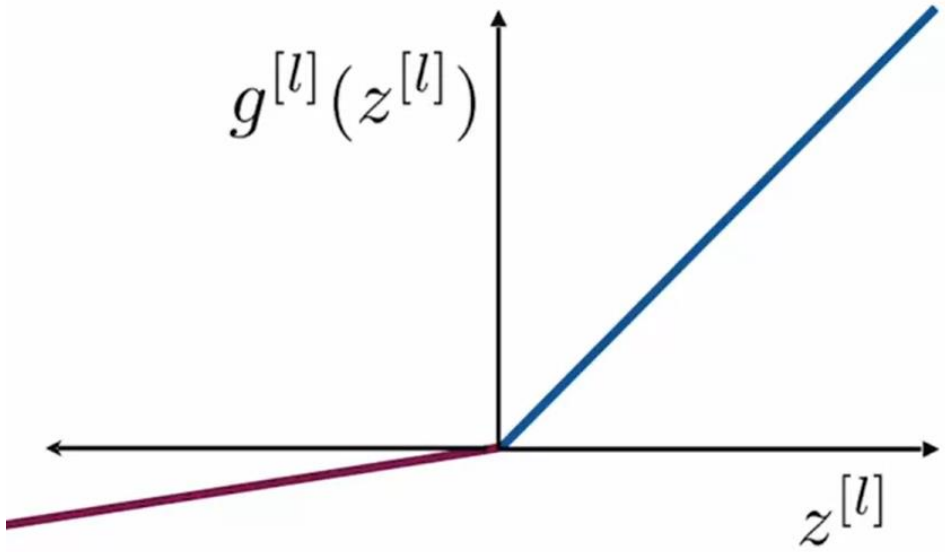
- ReLU
- Leaky ReLU
- Sigmoid
- Tanh



Foundations of GAN

Deep Convolutional GAN - Activation Functions

Activations: Leaky ReLU



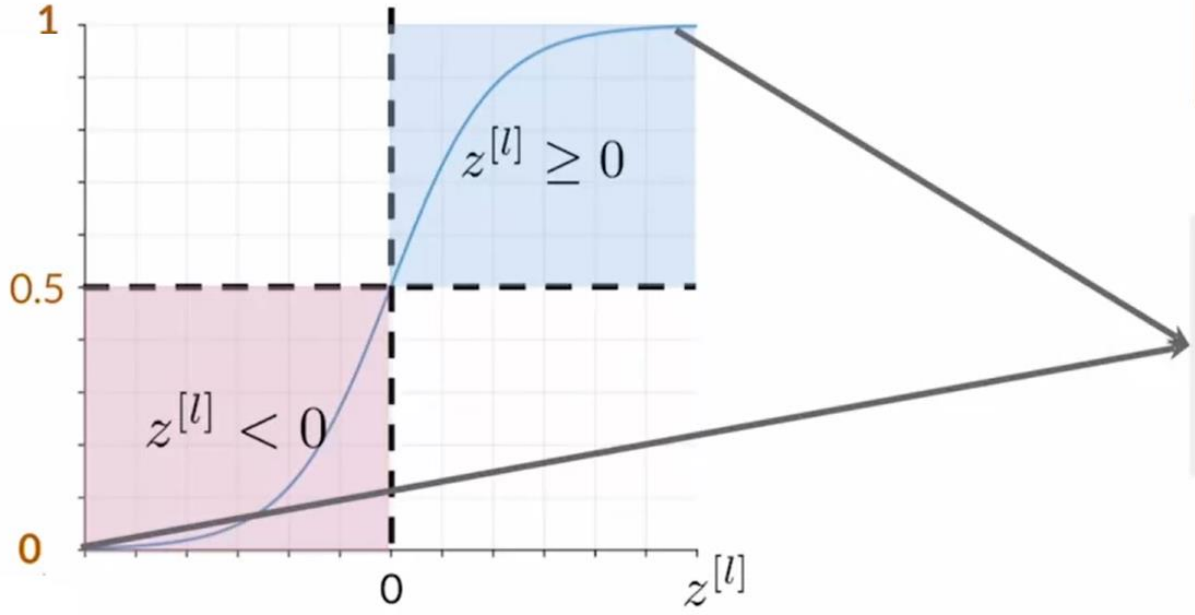
$$g^{[l]}(z^{[l]}) = max(az^{[l]}, z^{[l]})$$

Foundations of GAN

Deep Convolutional GAN - Activation Functions

Activations: Sigmoid

$$g^{[l]}(z^{[l]}) = \frac{1}{1 + e^{-z^{[l]}}}$$



Values between 0 and 1

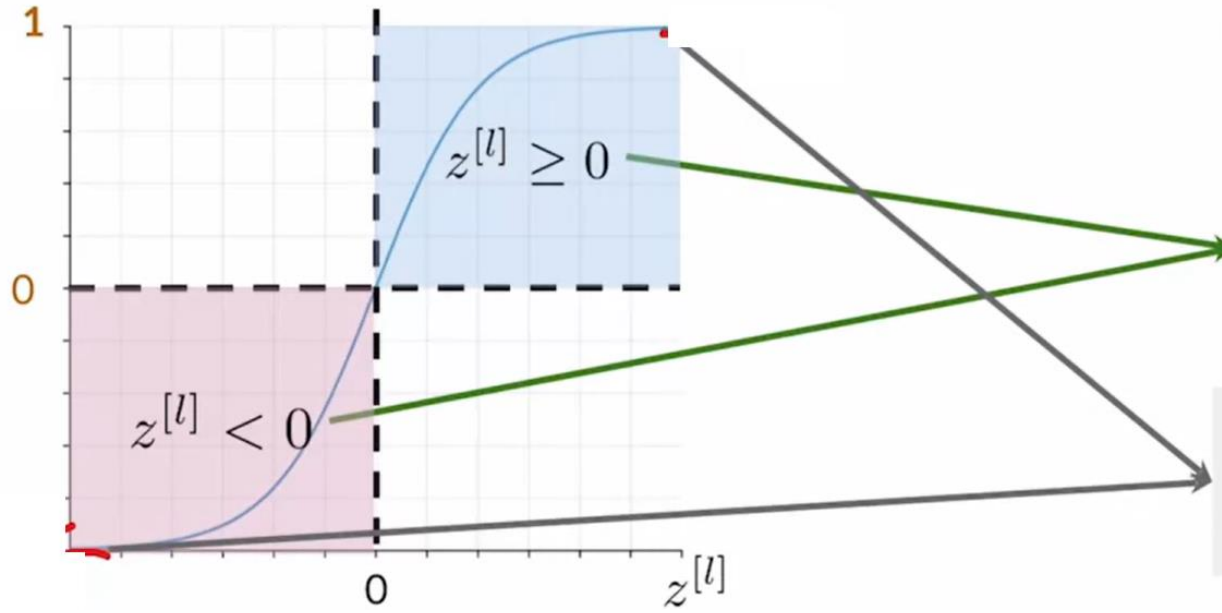
Vanishing gradient and saturation problems

Foundations of GAN

Deep Convolutional GAN - Activation Functions

Activations: Tanh

$$g^{[l]}(z^{[l]}) = tanh(z^{[l]})$$



Values between -1 and 1

Keeps the sign of the input

Same issues as Sigmoid

Foundations of GAN

Deep Convolutional GAN - Batch Normalization

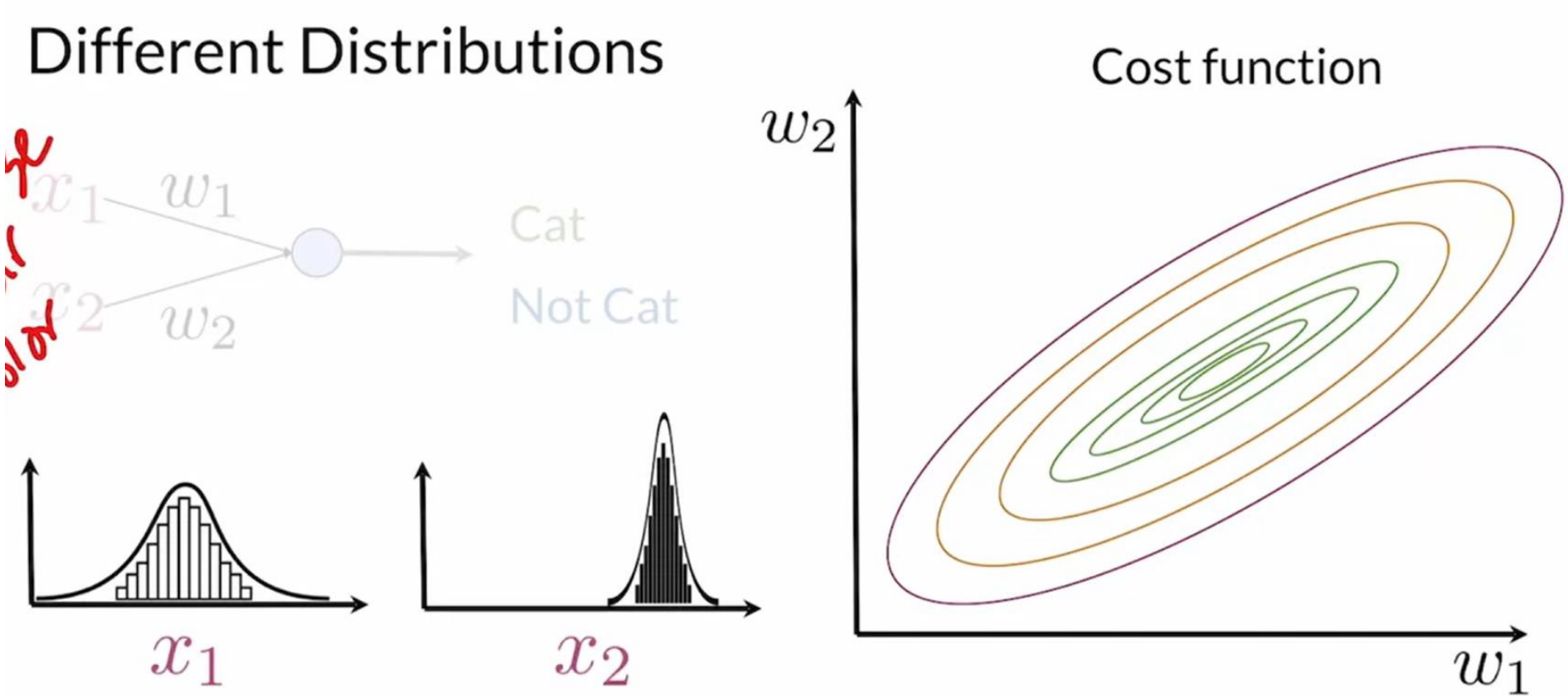
- Techniques that enhance training efficiency and stability, such as batch normalization, are vital for successful GAN implementation.
- Normalizes the output of each layer based on the batch's statistics (mean and standard deviation).

Benefits:

- **Smooths Cost Function**: Leads to a more balanced and easier-to-optimize cost function.
- **Reduces Internal Covariate Shift**: Helps stabilize the shifting distributions within hidden layers during training.
- **Speeds Up Training**: Enhances the efficiency of the training process by providing more stable gradients.

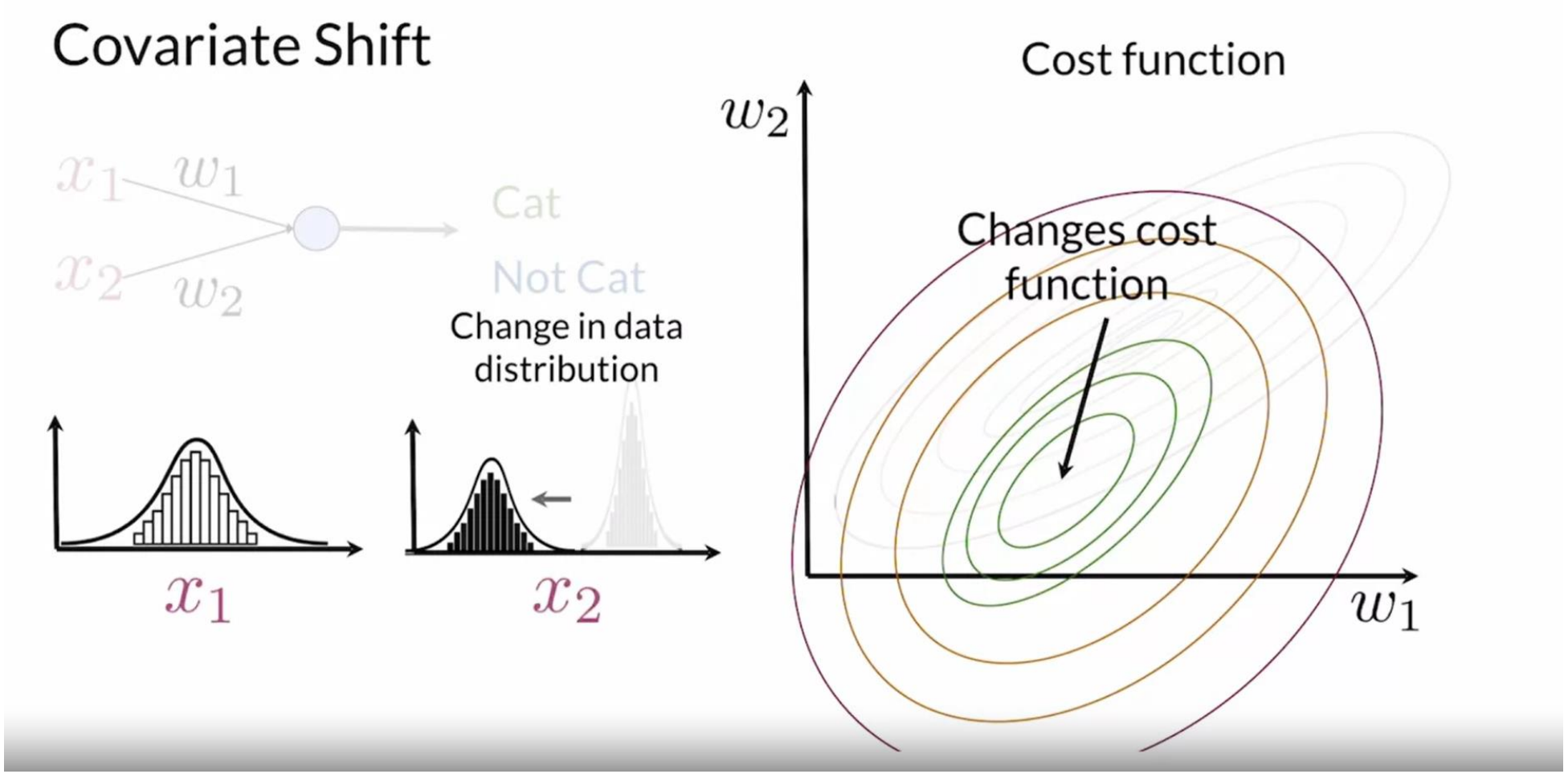
Foundations of GAN

Deep Convolutional GAN - Batch Normalization



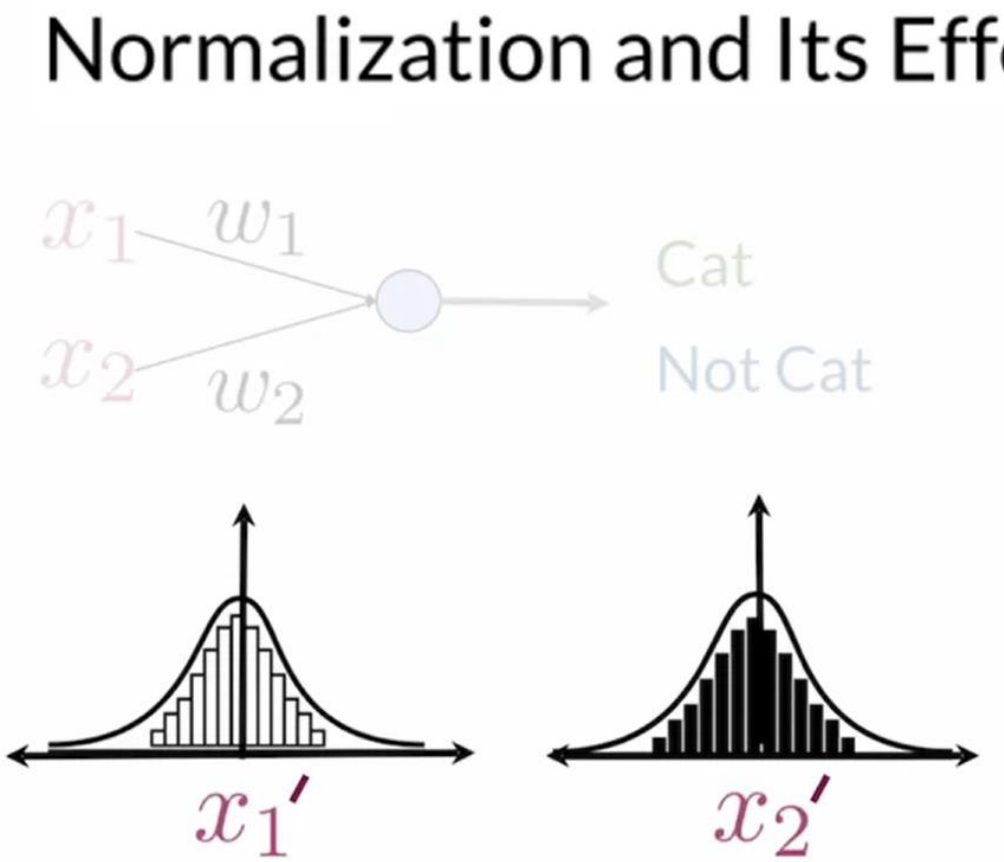
Foundations of GAN

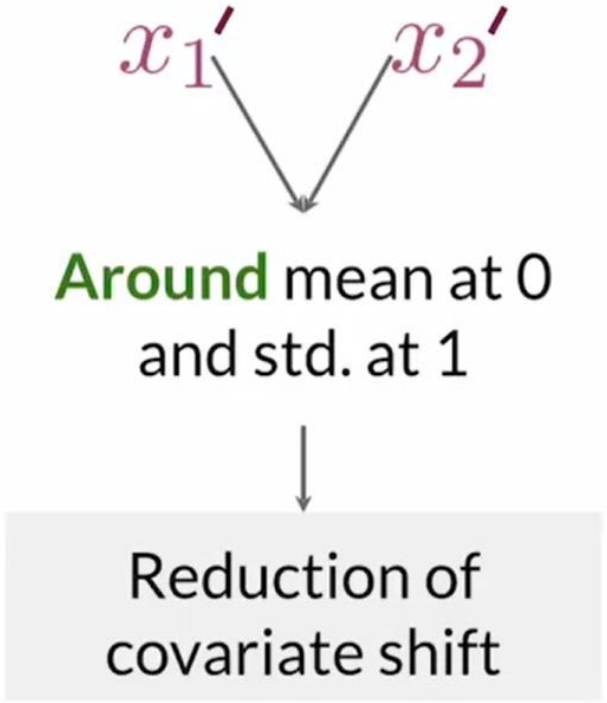
Deep Convolutional GAN - Batch Normalization



Foundations of GAN

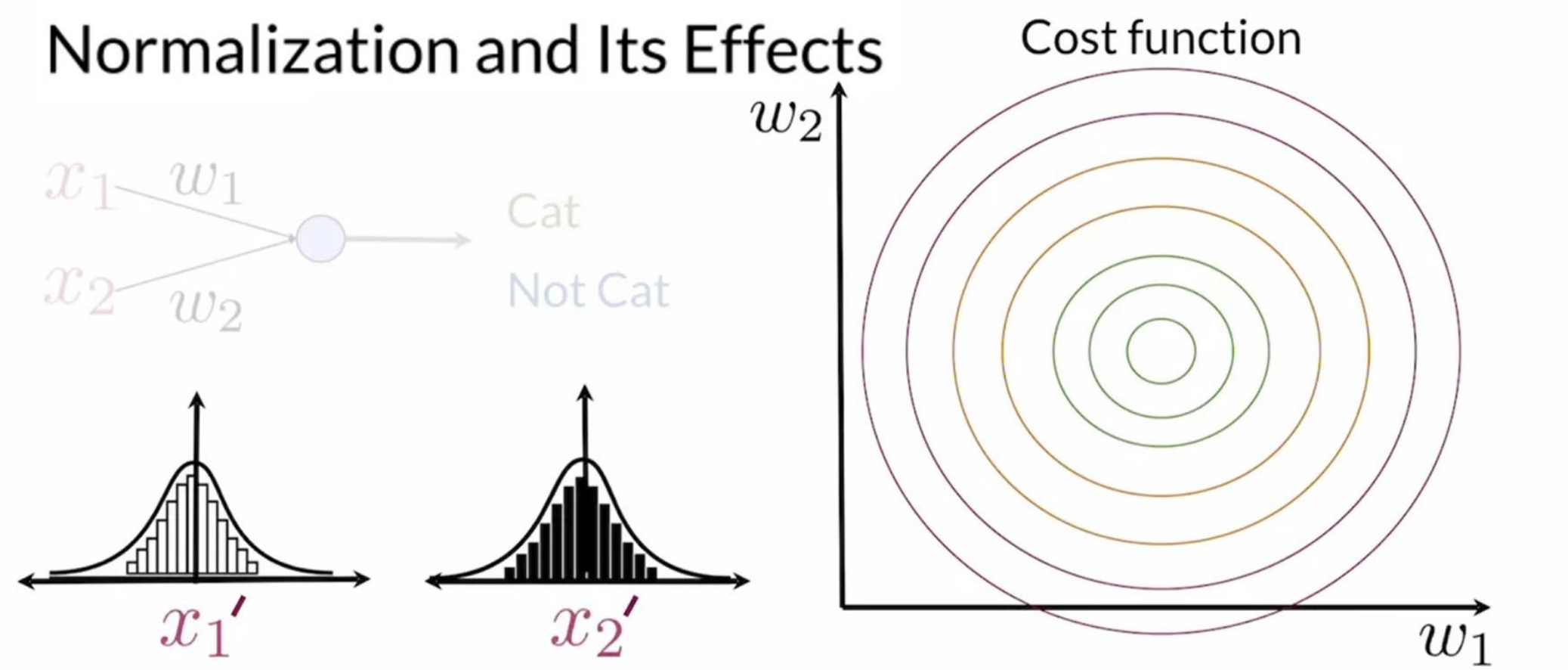
Deep Convolutional GAN - Batch Normalization





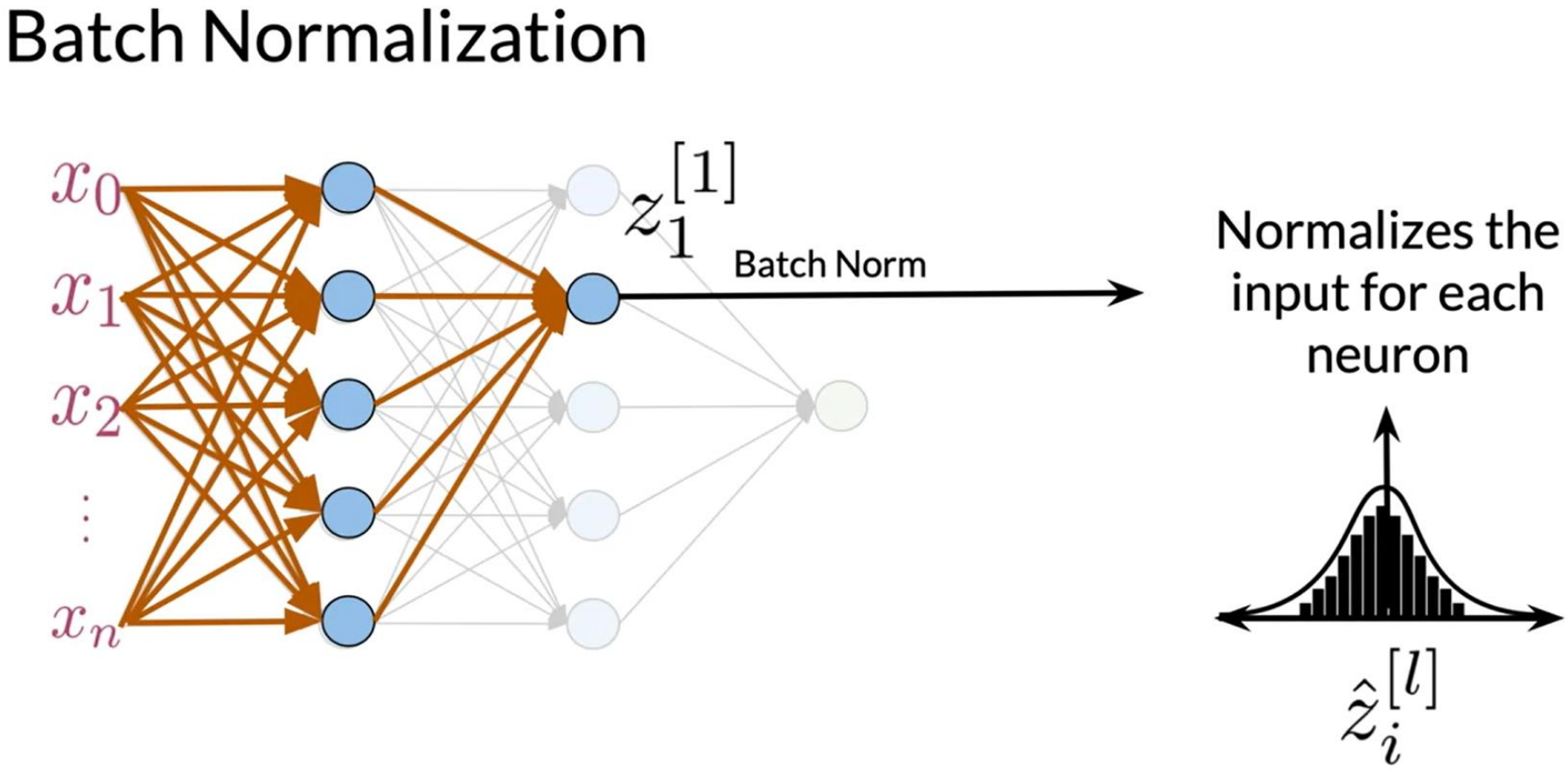
Foundations of GAN

Deep Convolutional GAN - Batch Normalization



Foundations of GAN

Deep Convolutional GAN - Batch Normalization



Foundations of GAN

Deep Convolutional GAN - Batch Normalization

Summary:

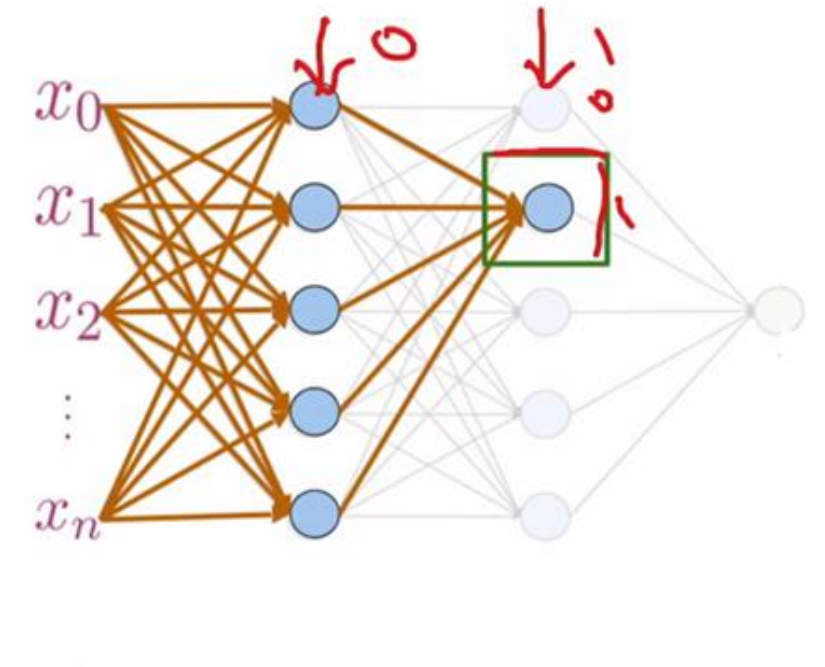
- **Key Benefit**: Batch normalization smooths the cost function and mitigates internal covariate shift
- **Use in GANs**: Critical for speeding up and stabilizing training, making it an essential technique for effective GAN development.

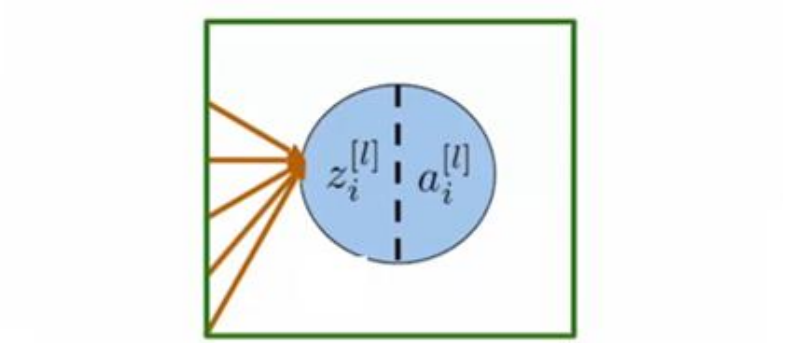
Foundations of GAN

Deep Convolutional GAN - Batch Normalization (Procedure)

Normalizes the inputs of each layer to stabilize and speed up training.

•**Training**: Uses batch statistics (mean and variance) calculated from the current batch to normalize inputs.





$$z_i^{[l]} = \sum_{i=0} W_i^{[l]} a_i^{[l-1]}$$ → For every example in the batch

$$\hat{z}_i^{[l]} = \frac{z_i^{[l]} - \mu_{z_i^{[l]}}}{\sqrt{\sigma^2_{z_i^{[l]}} + \epsilon}}$$

Batch mean of $z_i^{[l]}$

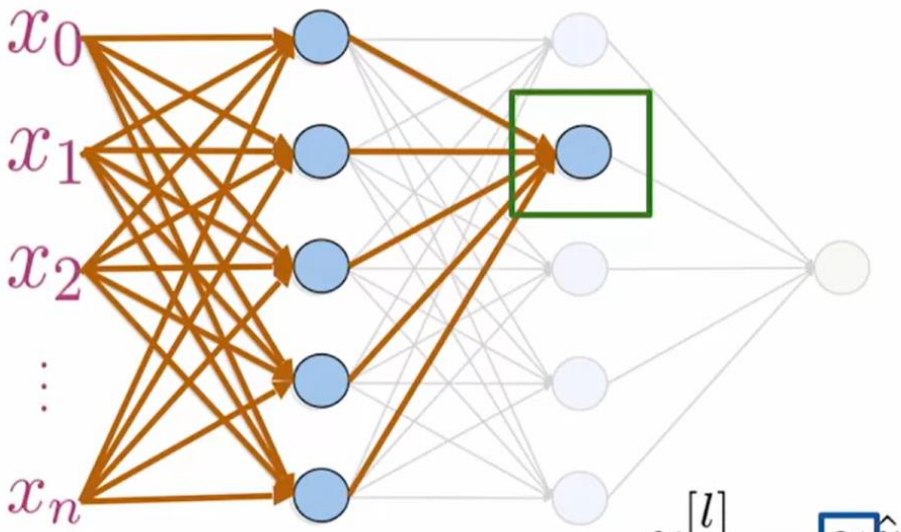
Batch std of $z_i^{[l]}$

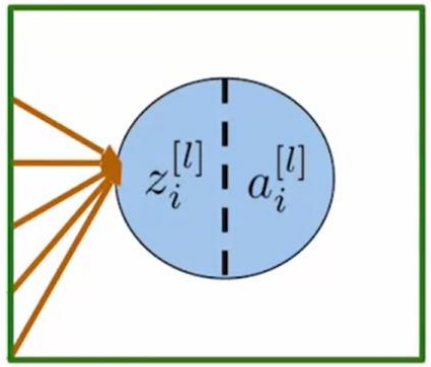
Foundations of GAN

Deep Convolutional GAN - Batch Normalization (Procedure)

Normalizes the inputs of each layer to stabilize and speed up training.

•**Training**: Uses batch statistics (mean and variance) calculated from the current batch to normalize inputs.





$$\hat{z}_i^{[l]} = \frac{z_i^{[l]} - \mu_{z_i^{[l]}}}{\sqrt{\sigma^2_{z_i^{[l]}} + \epsilon}}$$

$$y_i^{[l]} = \gamma \hat{z}_i^{[l]} + \beta$$

Shift factor

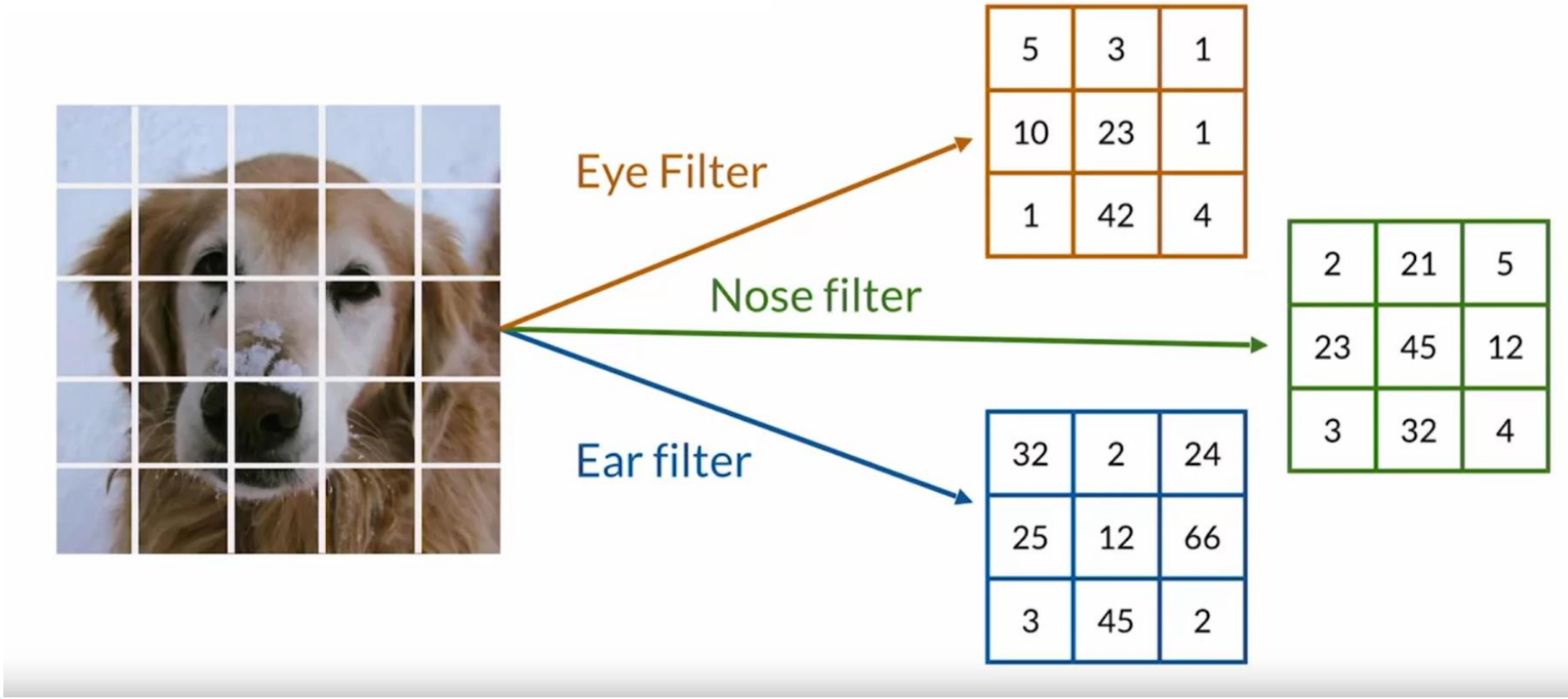
Scale Factor

Learnable parameters to get the optimal dist.

Foundations of GAN

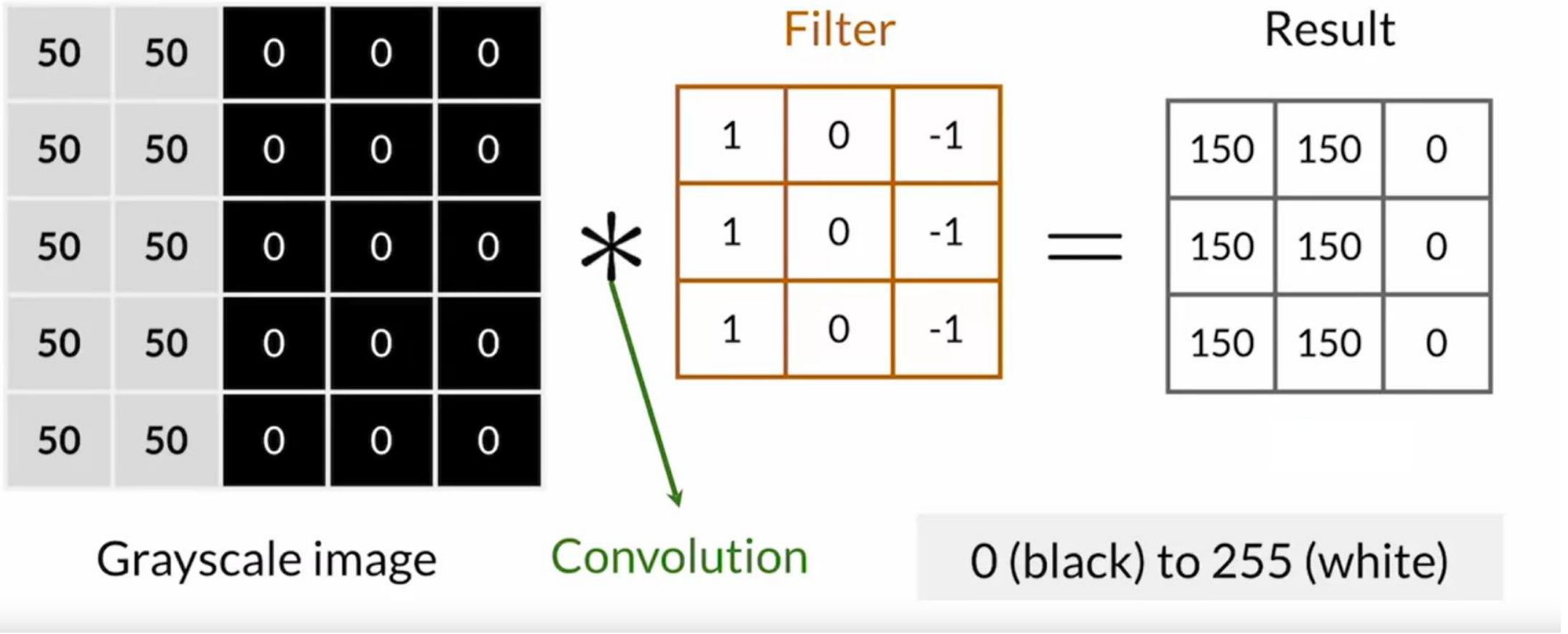
Convolutions in GANs

- Essential for detecting key features in images.
- Used extensively to process and generate images.



Foundations of GAN

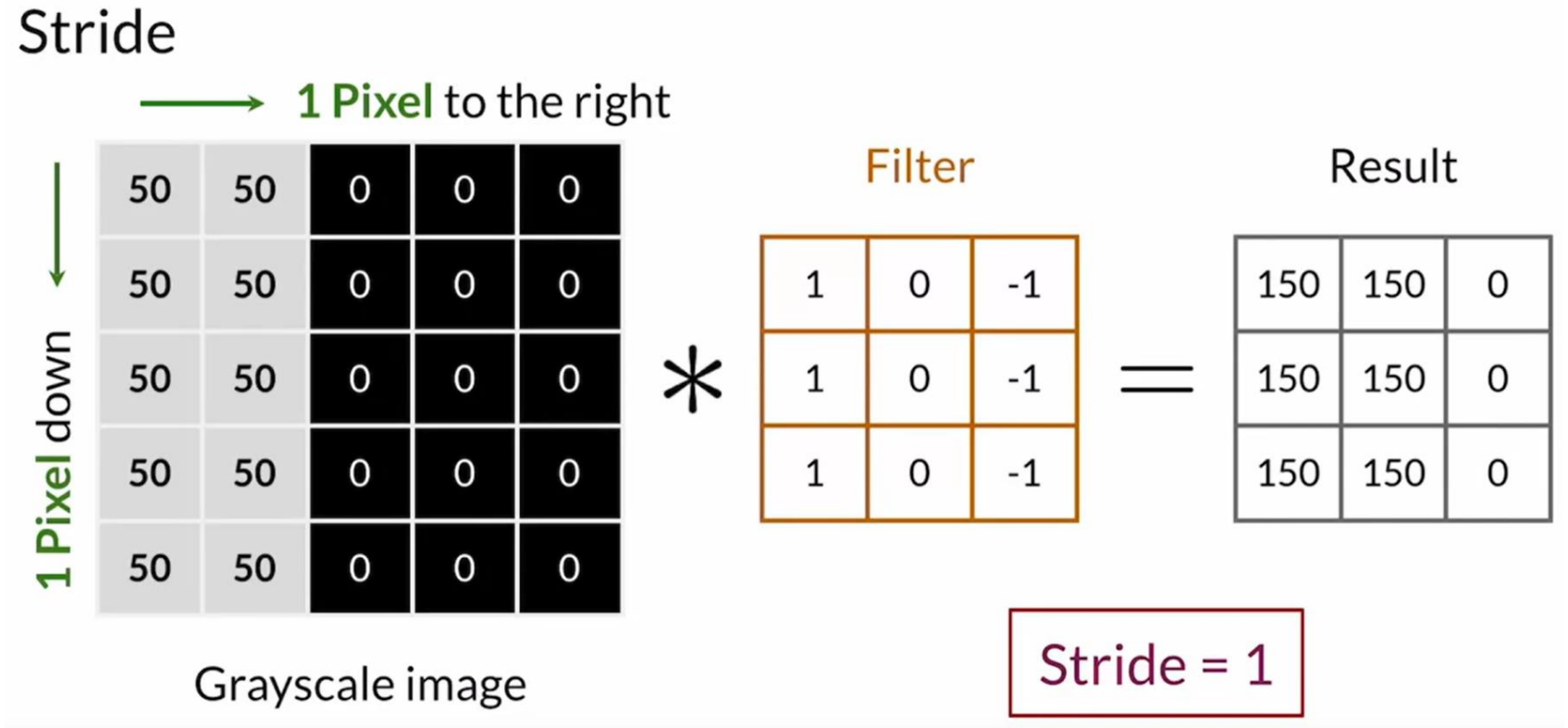
Convolutions in GANs



Foundations of GAN

Convolutions in GANs -Padding and Stride

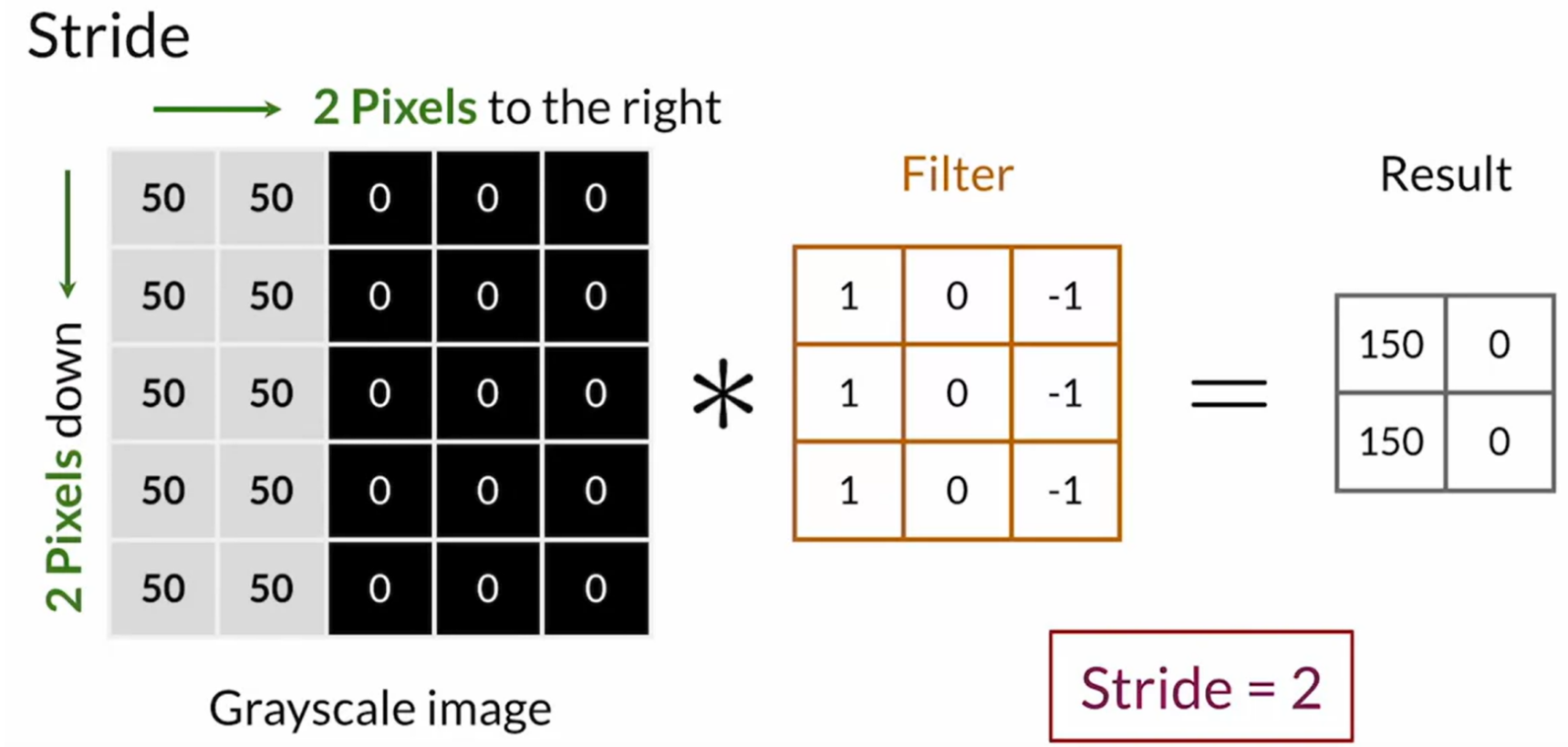
Stride: Determines filter movement across the image, influencing output size and pixel visitation frequency.



Foundations of GAN

Convolutions in GANs -Padding and Stride

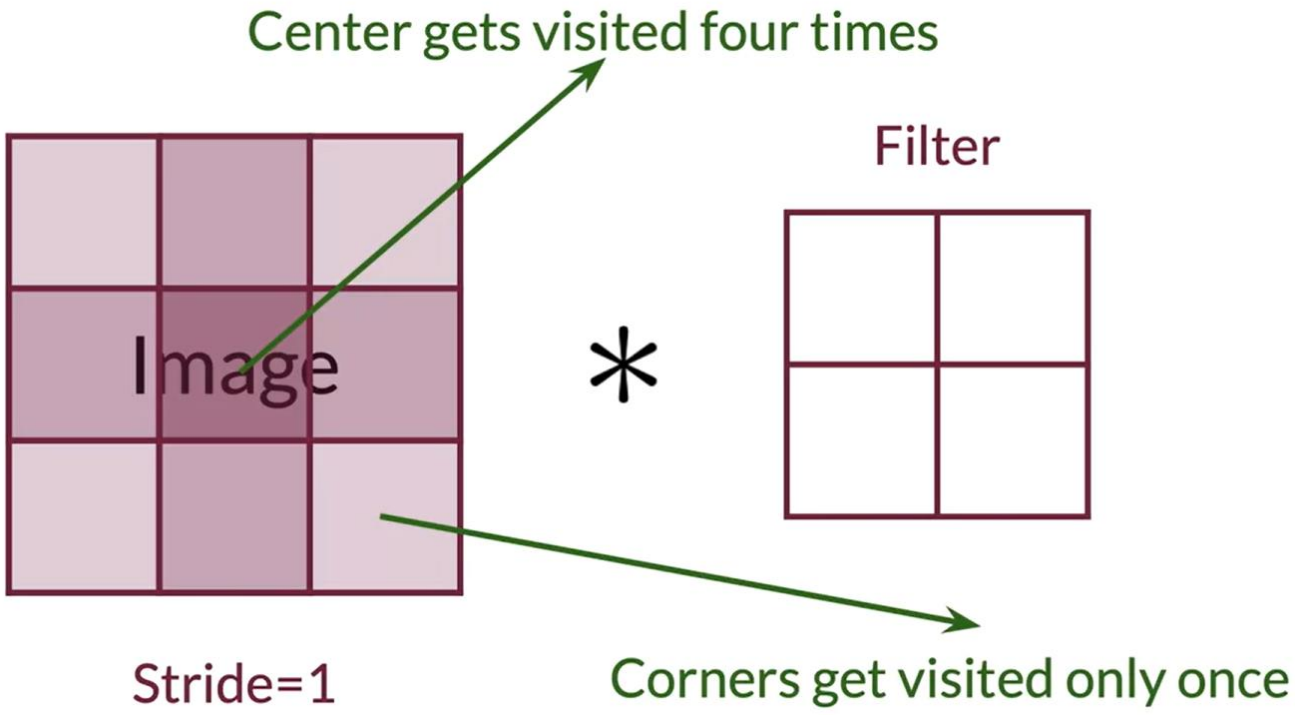
Stride: Determines filter movement across the image, influencing output size and pixel visitation frequency.



Foundations of GAN

Convolutions in GANs -Padding and Stride

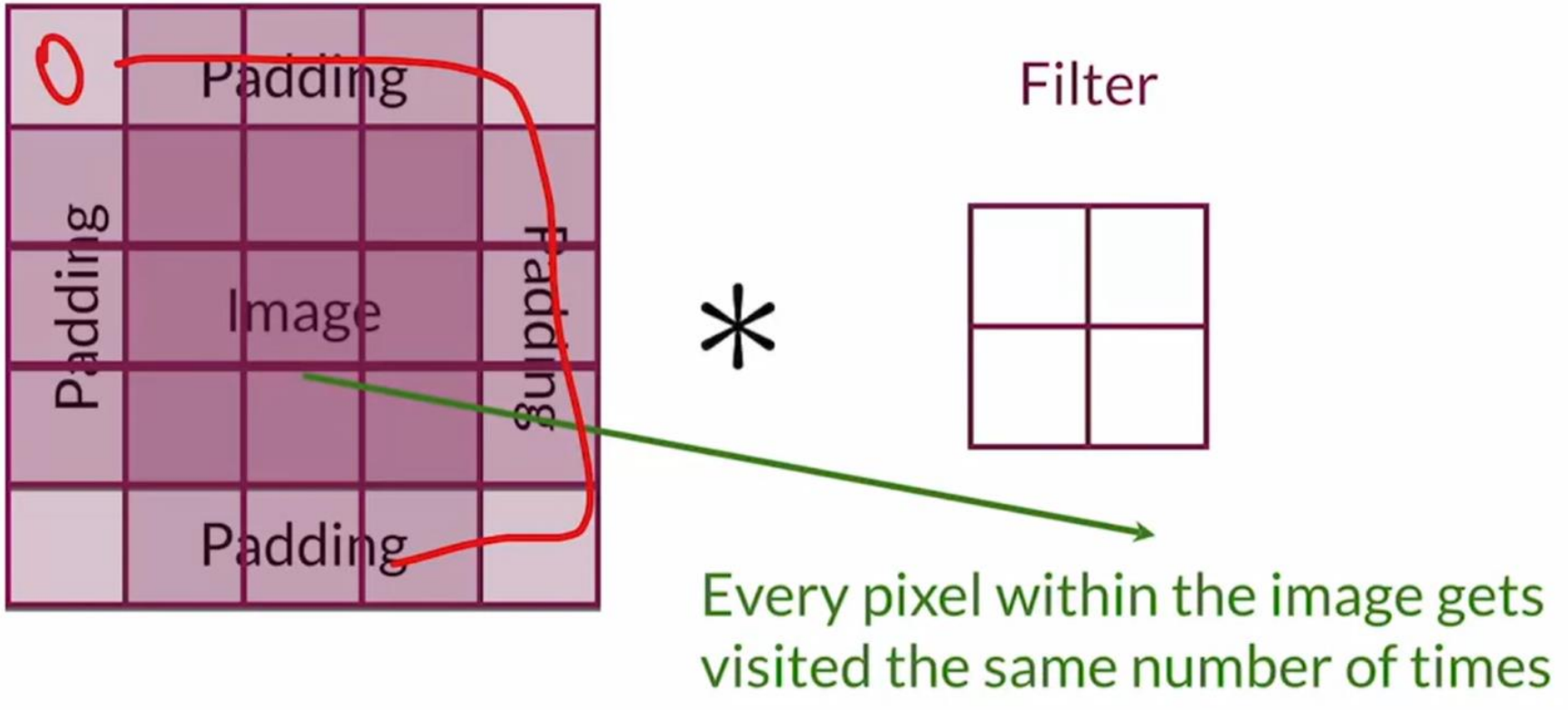
Padding: Adds a frame around the image to ensure edge pixels receive equal attention, typically using zero-padding for simplicity.



Foundations of GAN

Convolutions in GANs -Padding and Stride

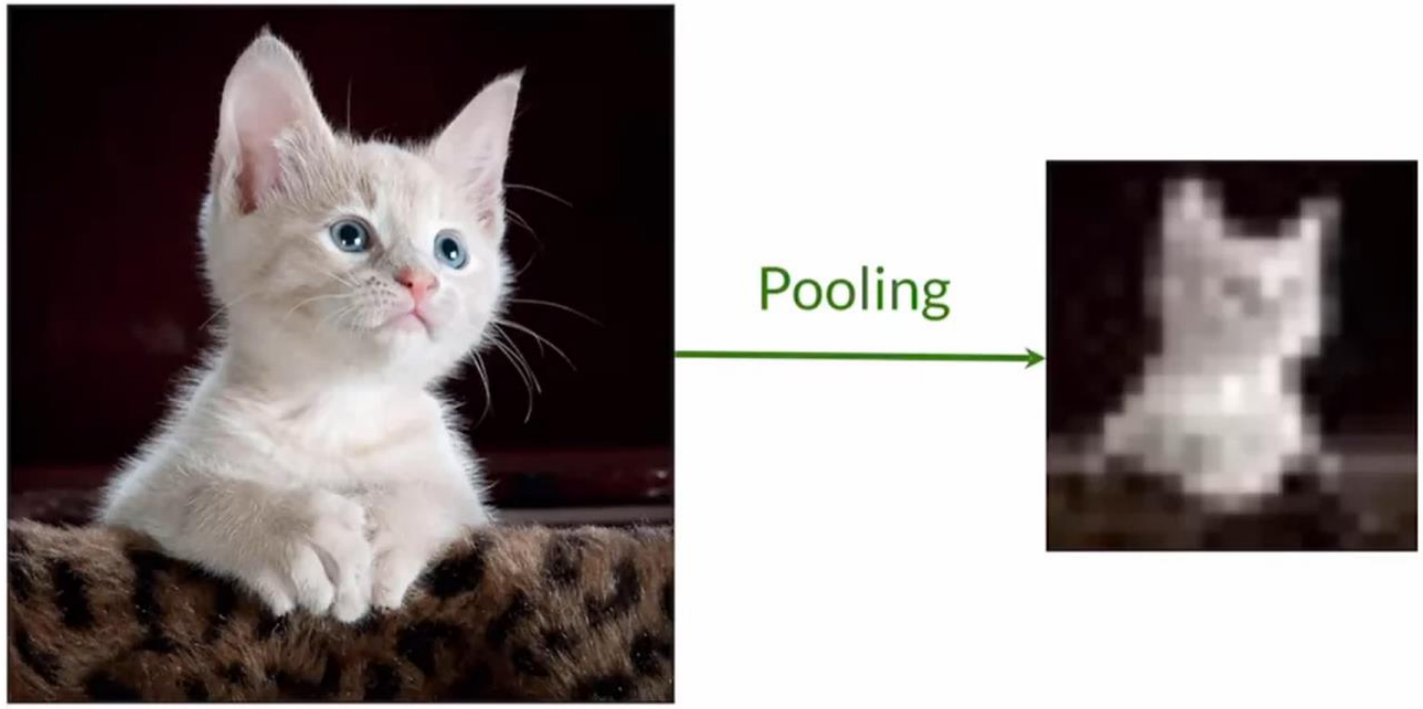
Padding: Adds a frame around the image to ensure edge pixels receive equal attention, typically using zero-padding for simplicity.



Foundations of GAN

Convolutions in GANs -Pooling and Upsampling

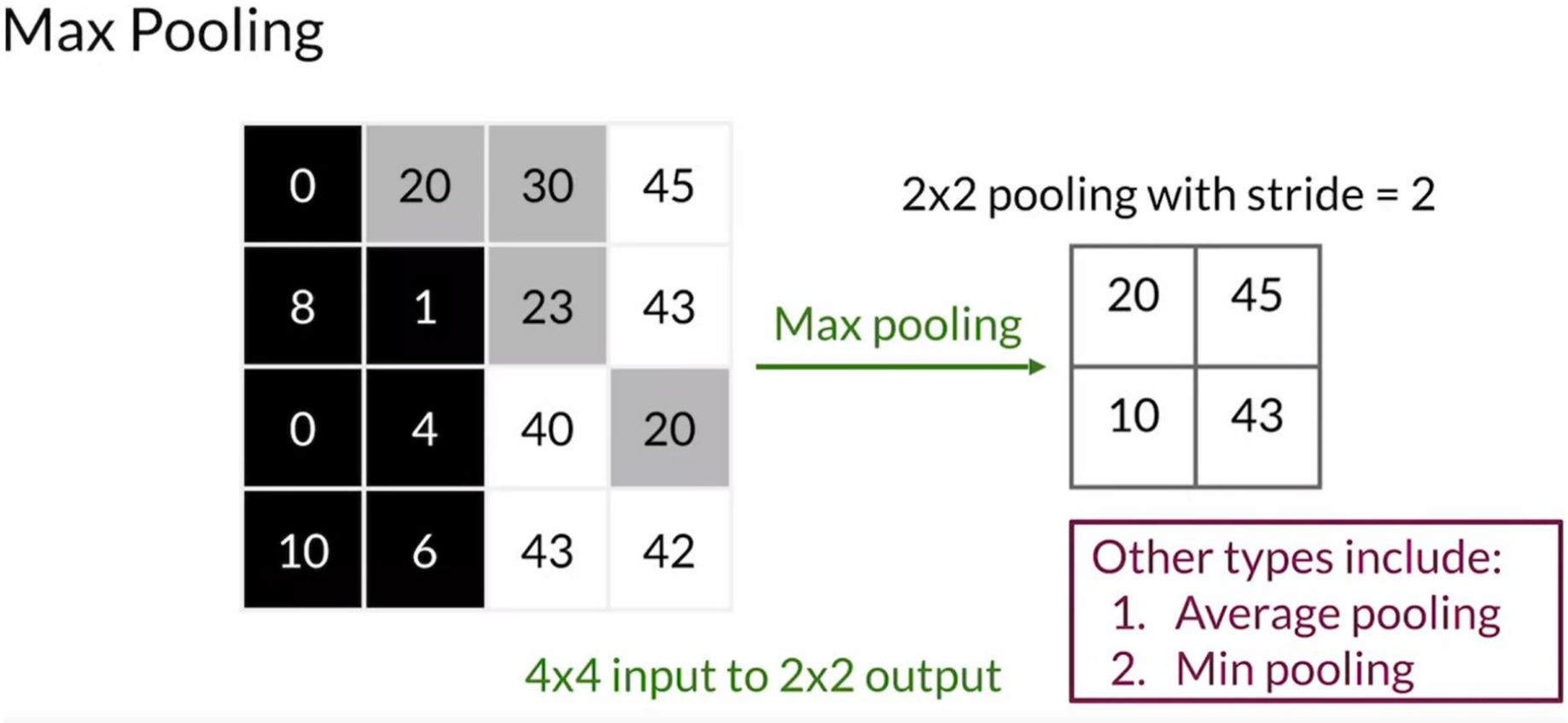
- **Pooling**: Reduces the size of the input
- **Up-Sampling**: Increases the size of the input.



Foundations of GAN

Convolutions in GANs -Pooling and Upsampling

- **Pooling**: Reduces the size of the input



Foundations of GAN

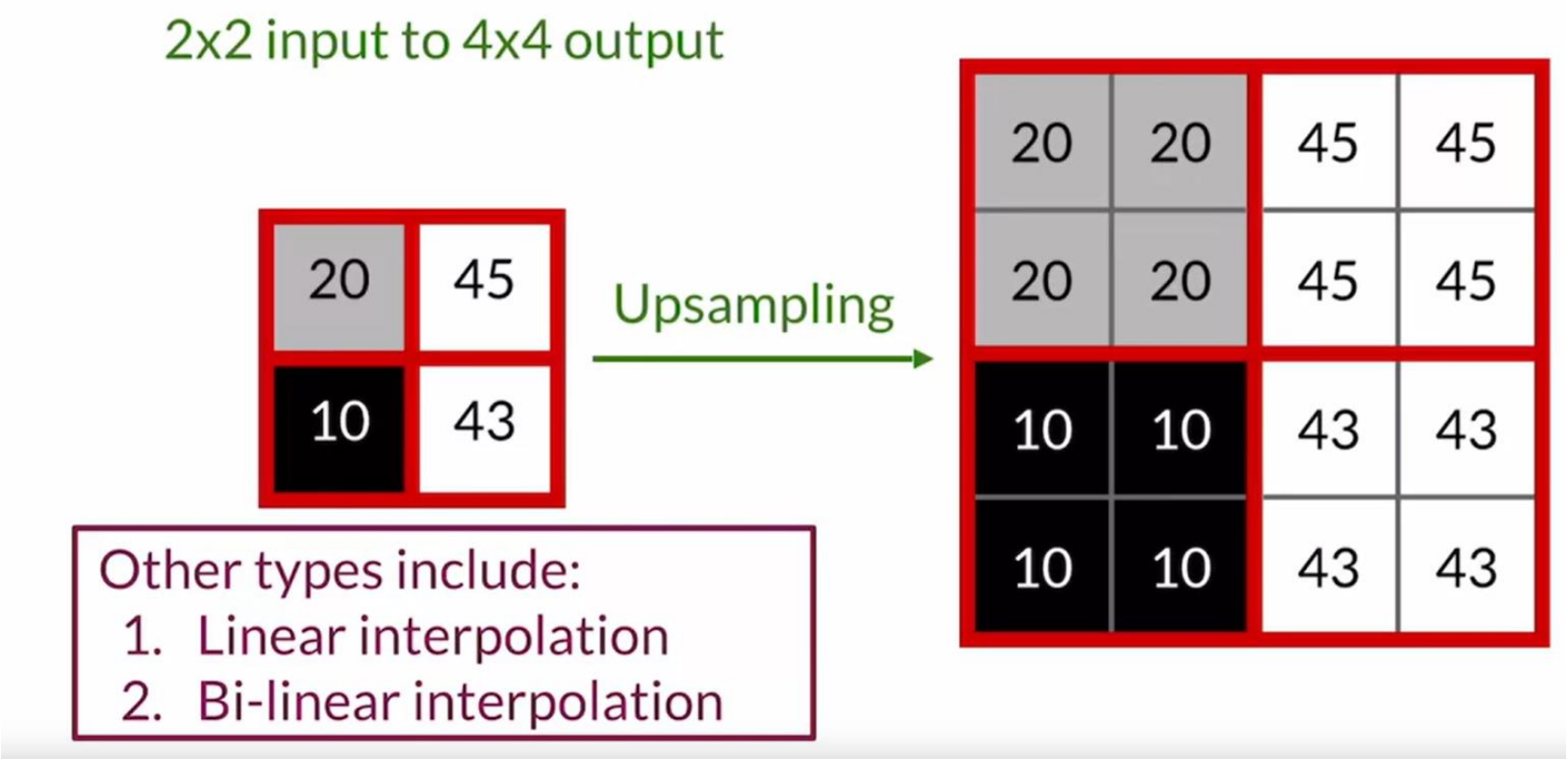
Convolutions in GANs -Pooling and Upsampling

- **Up-Sampling**: Increases the size of the input.

Foundations of GAN

Convolutions in GANs -Pooling and Upsampling

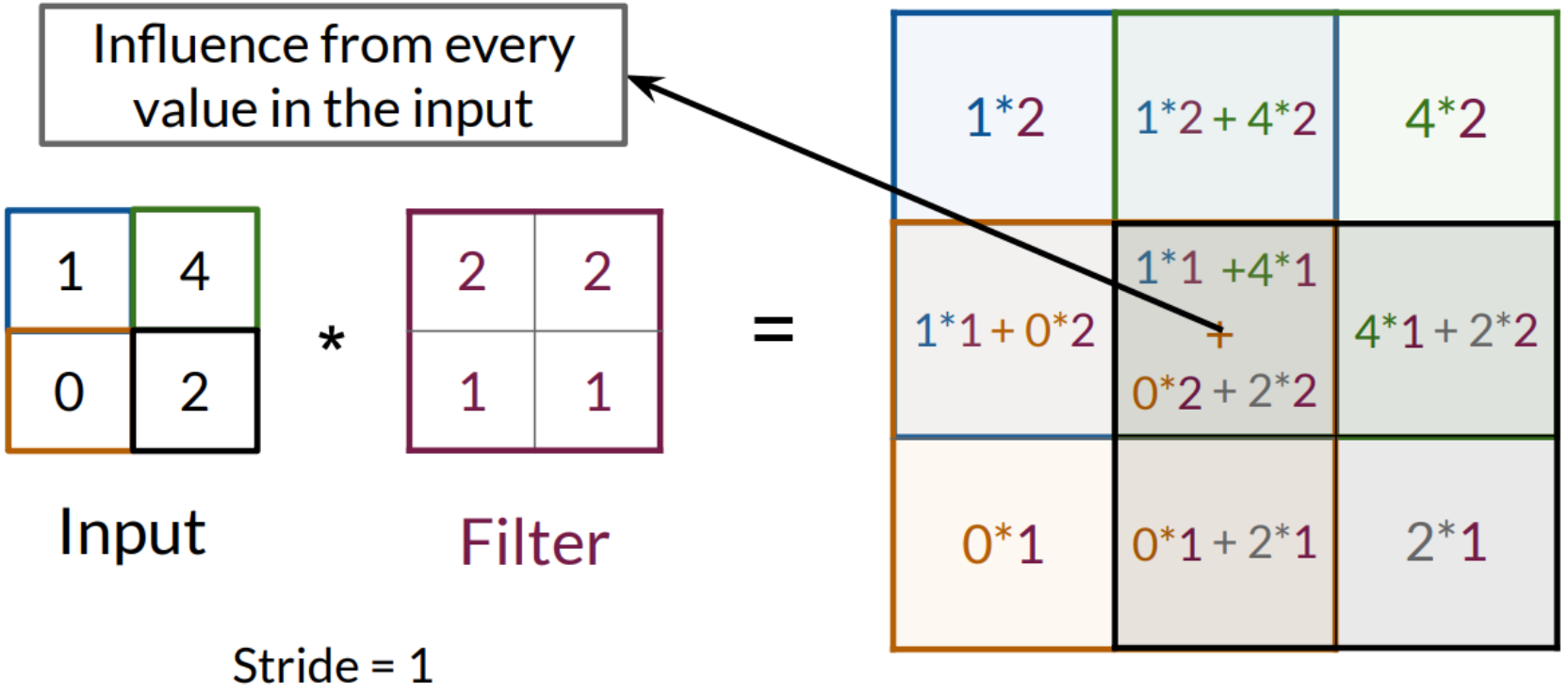
Upsampling Nearest Neighbors



Foundations of GAN

Convolutions in GANs -Transposed Convolutions

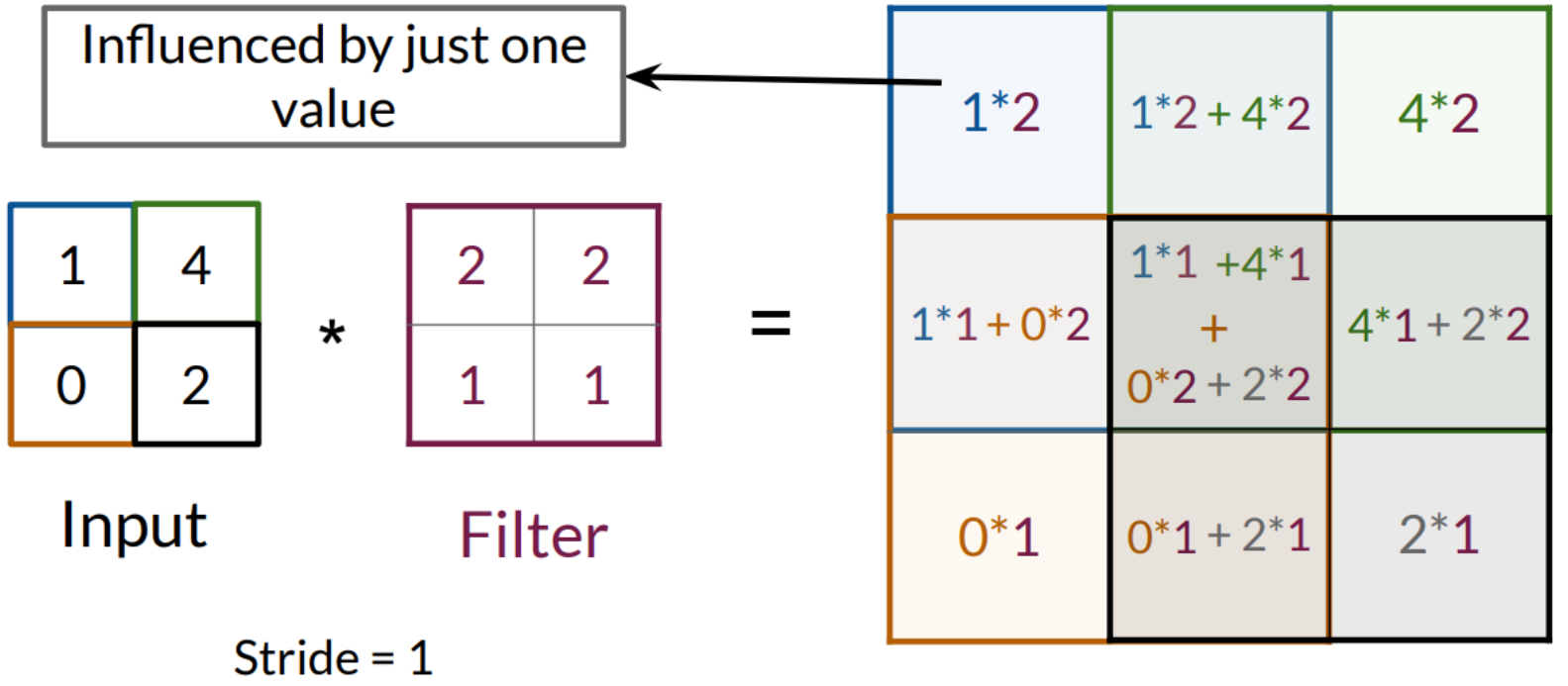
- Upsample an input image using a learnable filter.
- Similar to convolutions but in reverse, enlarging the input size.



Foundations of GAN

Convolutions in GANs -Transposed Convolutions

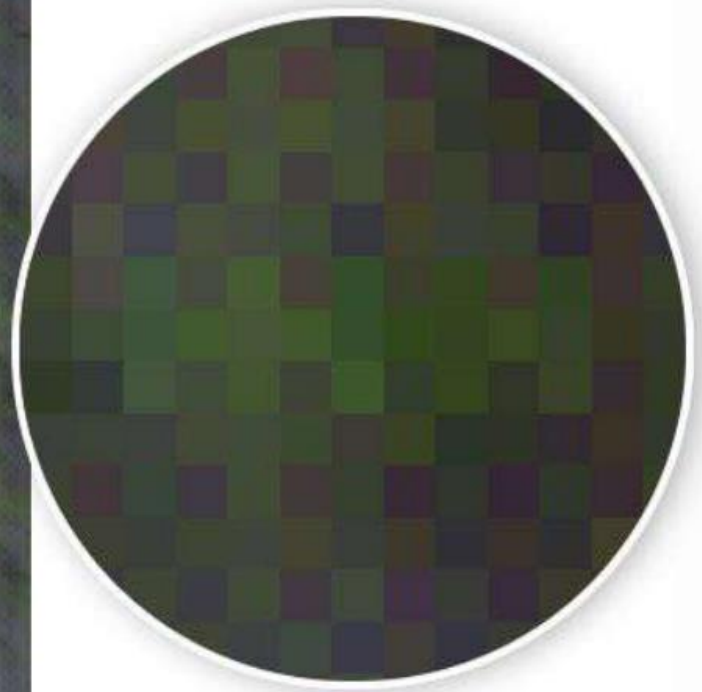
- Upsample an input image using a learnable filter.
- Similar to convolutions but in reverse, enlarging the input size.



Foundations of GAN

Convolutions in GANs -Transposed Convolutions

The Problems with Transposed Convolution



Checkerboard Pattern

Foundations of GAN

Demo: Deep Convolutional Generative Adversarial Network (DCGAN)

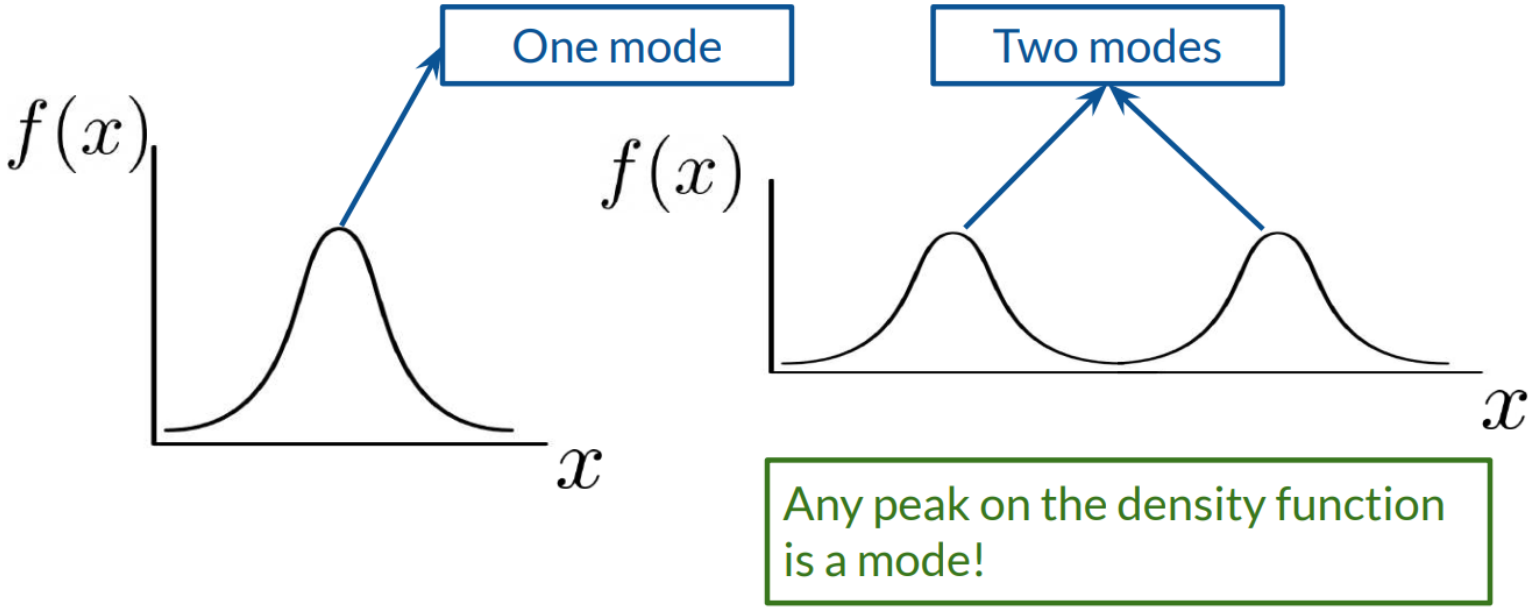
PRACTICAL
DEMONSTRATION

Issues with Vanilla GANs

Foundations of GAN

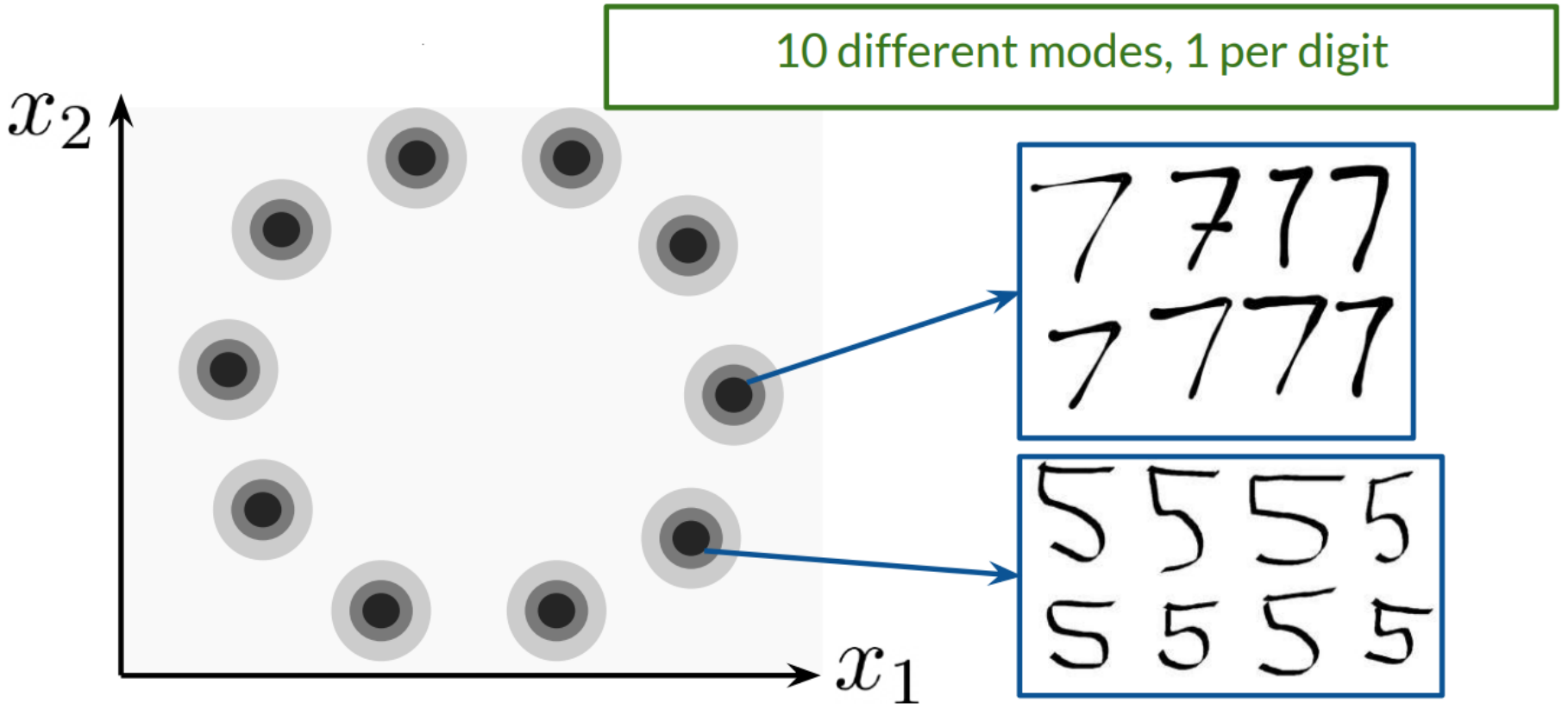
Wasserstein GANs with Normalization – Mode Collapse in GANs

- Traditional GANs using binary cross-entropy (BCE) loss can face mode collapse and vanishing gradients.
- A mode is an area with a high concentration of observations in a probability distribution.
- Distributions can be unimodal, bimodal, or multimodal, indicating one, two, or multiple peaks respectively.



Foundations of GAN

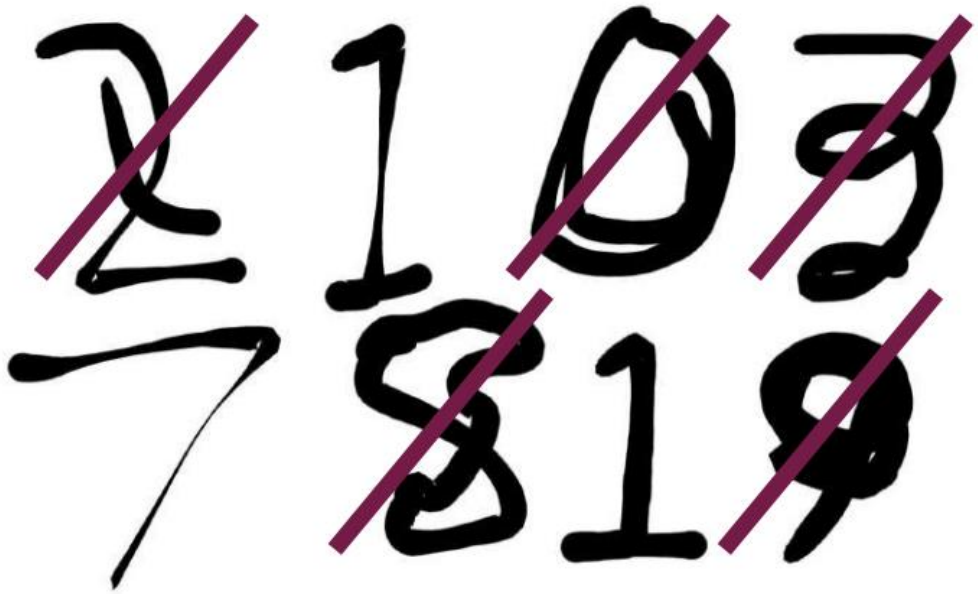
Wasserstein GANs with Normalization – Mode Collapse in GANs



Foundations of GAN

Wasserstein GANs with Normalization – Mode Collapse in GANs

Discriminator

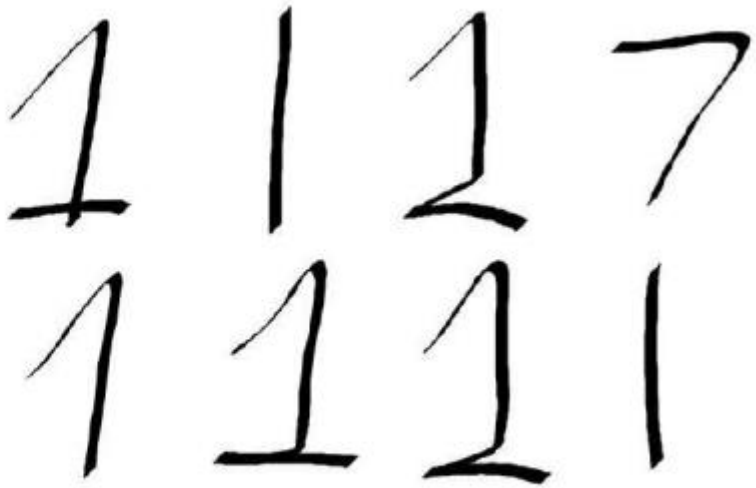


Fakes

Discriminator only predicts 1 and 7 as Real and rest as Fakes

Foundations of GAN

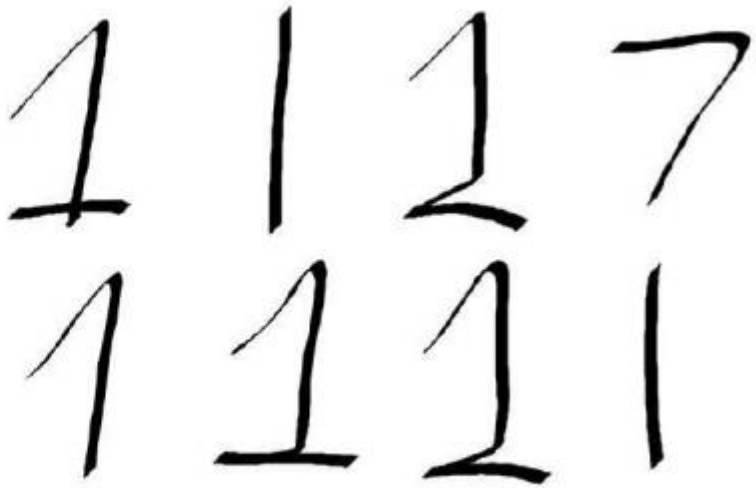
Wasserstein GANs with Normalization – Mode Collapse in GANs



Generator gets the feedback and starts generating only 1 and 7

Foundations of GAN

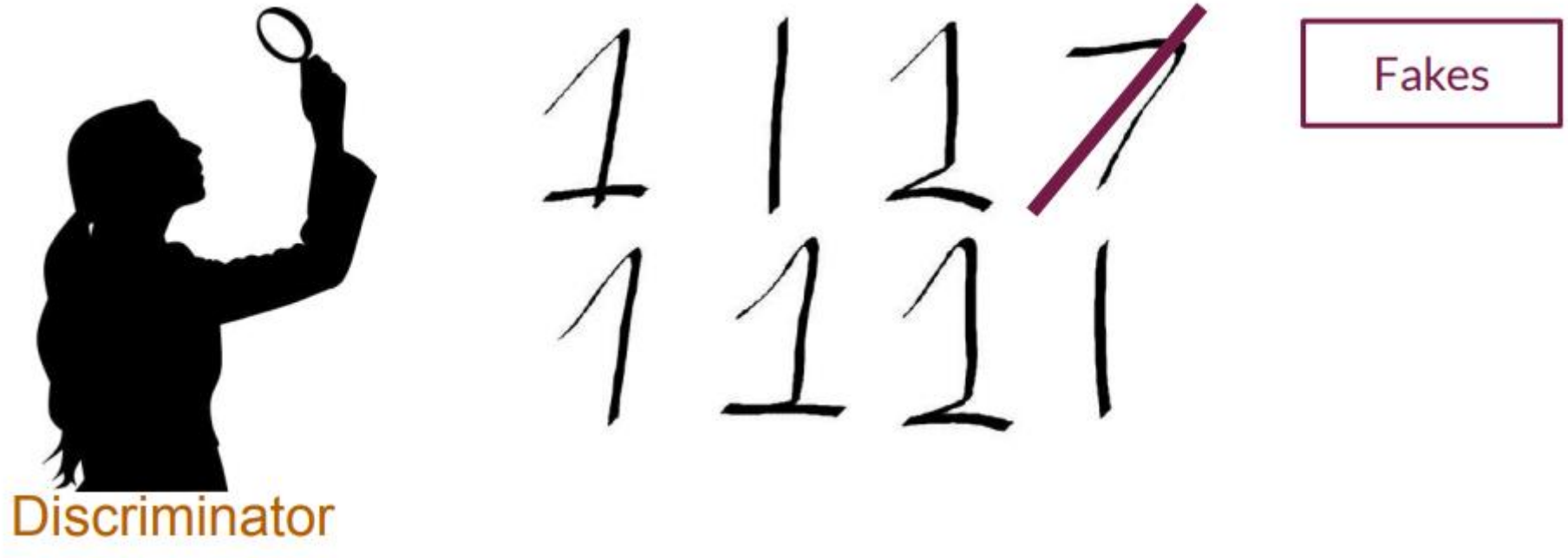
Wasserstein GANs with Normalization – Mode Collapse in GANs



Generator gets the feedback and starts generating only 1 and 7

Foundations of GAN

Wasserstein GANs with Normalization – Mode Collapse in GANs

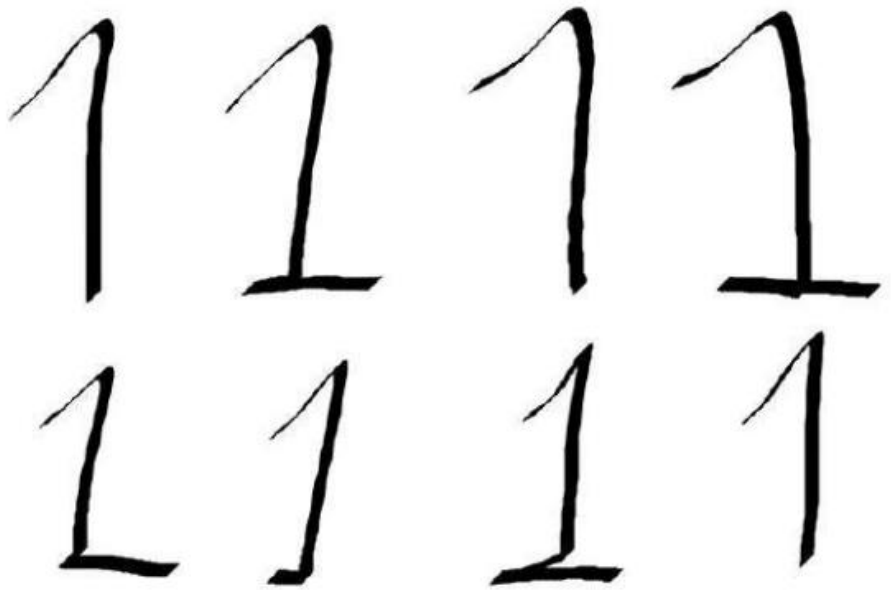


Discriminator only predicts 1 as Real and rest as Fakes

Foundations of GAN

Wasserstein GANs with Normalization – Mode Collapse in GANs

Generator



Generator gets the feedback and starts generating only 1

Foundations of GAN

Problem with BCE Loss

- BCE loss is traditionally used for training GANs but is not optimal.
- It is prone to mode collapse and vanishing gradient problems.

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log h(x^{(i)},\theta) + (1-y^{(i)})\log(1-h(x^{(i)},\theta))\right]$$

Prediction

Label

Features

Parameters

Generator

Maximize cost

Discriminator

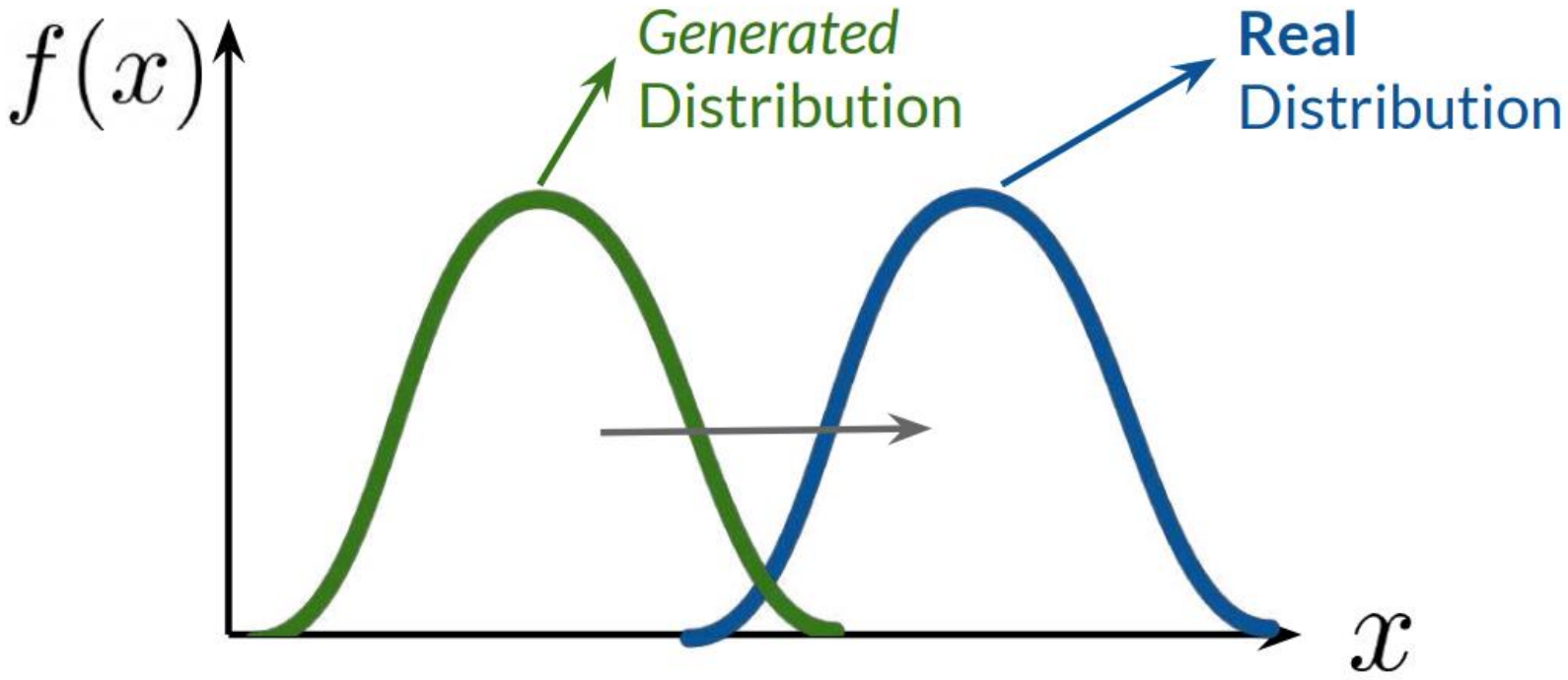
Minimize cost

Foundations of GAN

Problem with BCE Loss

- Objective in GANs

Make the generated and real distributions look similar



Foundations of GAN

Problem with BCE Loss

- **BCE Loss in GANs**

Criticizing is more straightforward

Discriminator

Single output

Easier to train than the generator

Generator

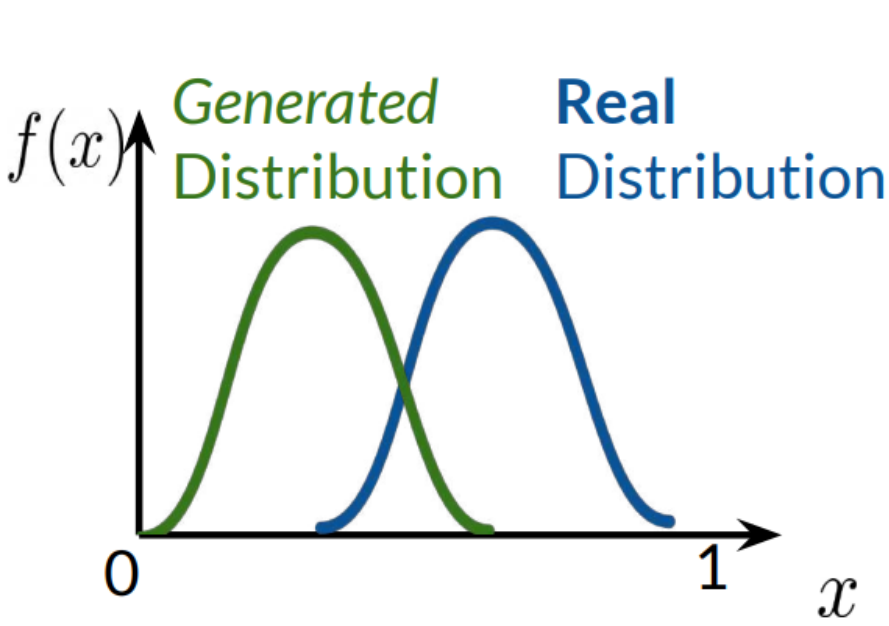
Complex output

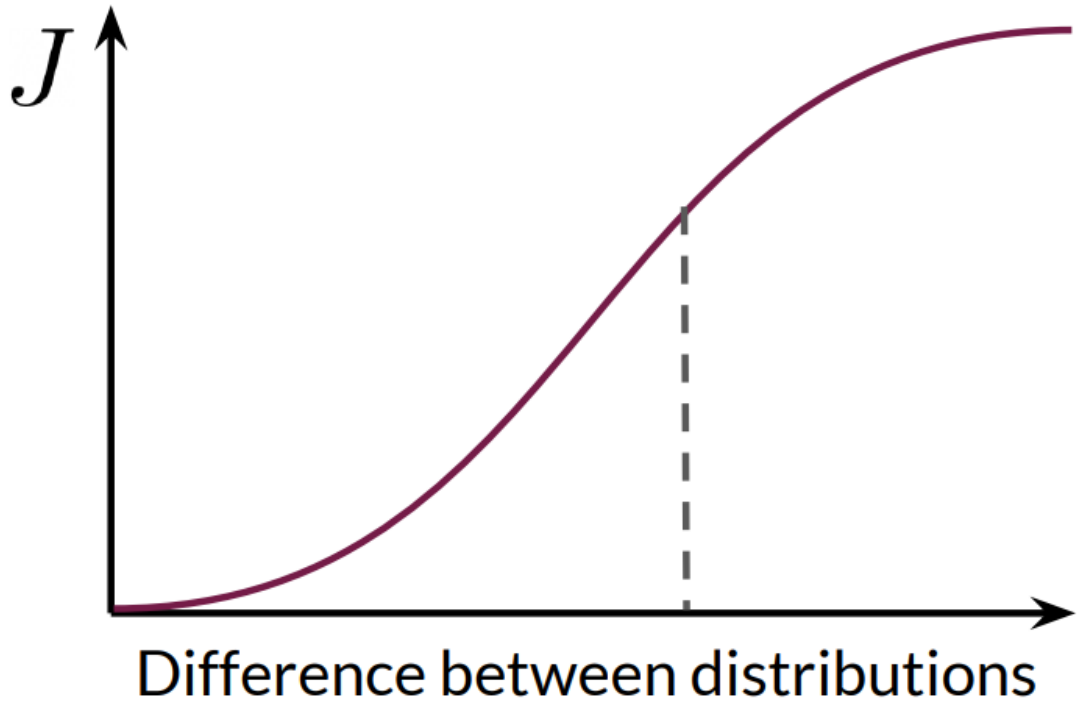
Difficult to train

Often, the discriminator gets better than the generator

Foundations of GAN

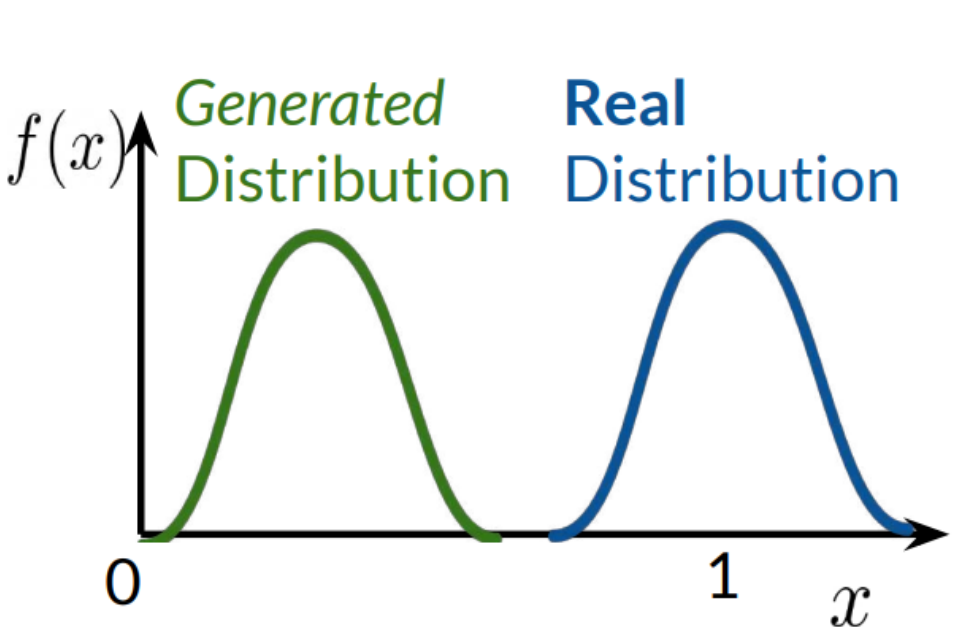
Problem with BCE Loss

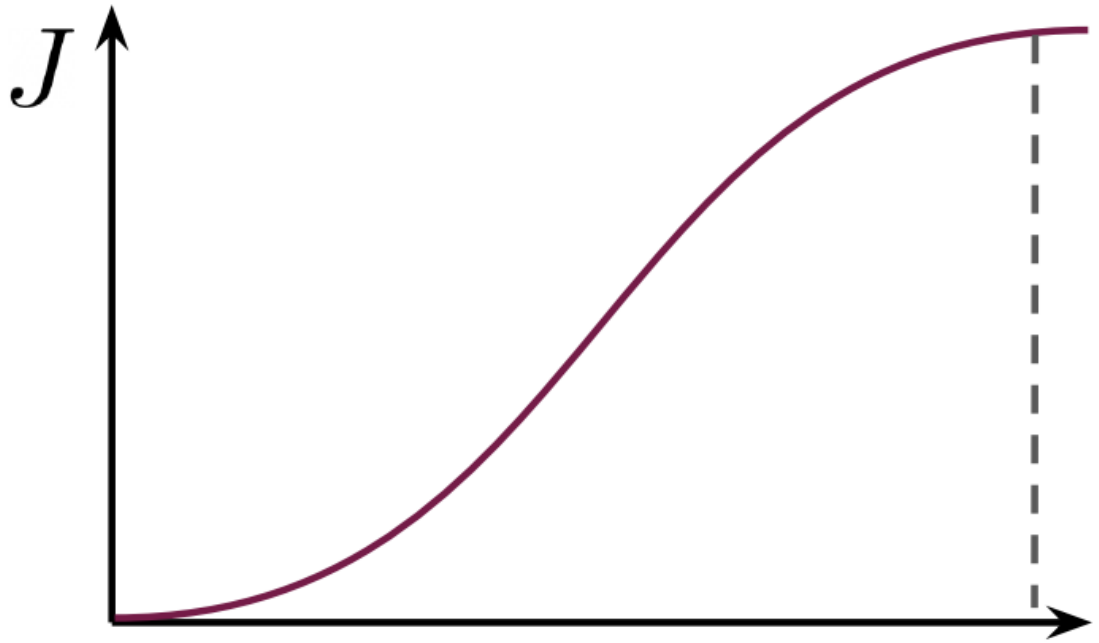




Foundations of GAN

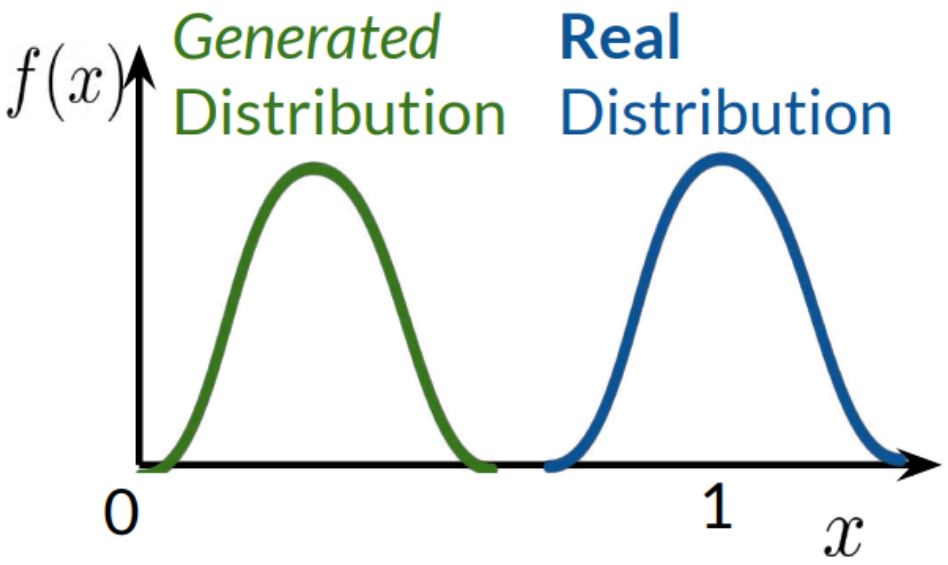
Problem with BCE Loss

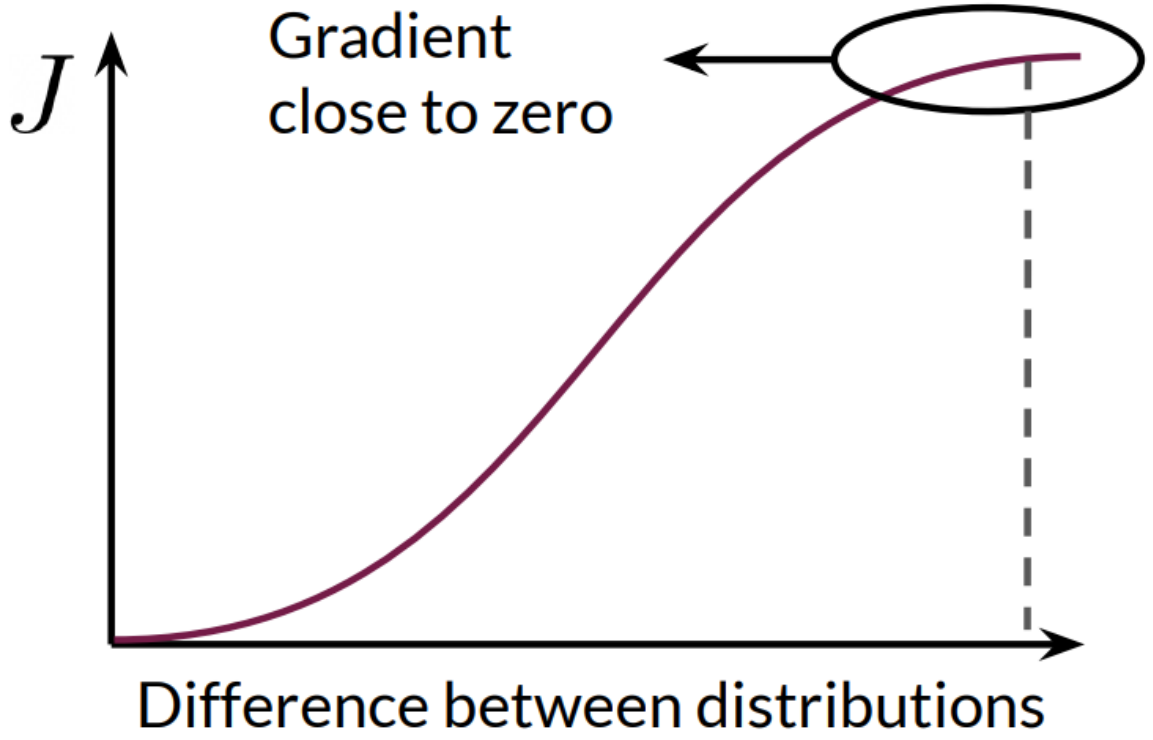




Foundations of GAN

Problem with BCE Loss



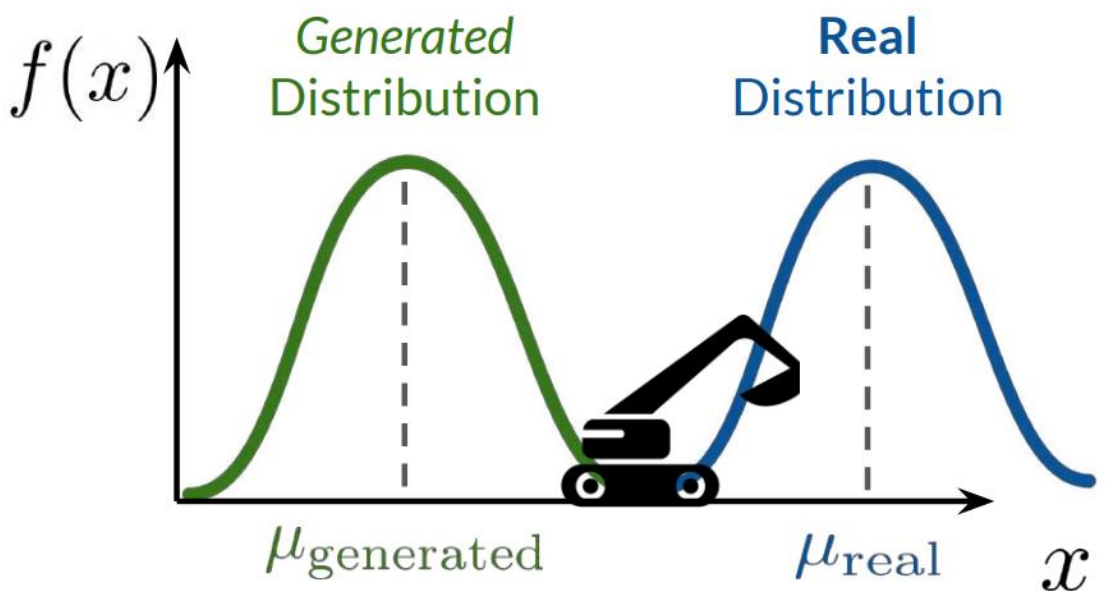


Foundations of GAN

Earth Mover's Distance

•EMD measures the distance between two distributions by estimating the effort required to transform one distribution into the other.

•it quantifies how much and how far "dirt" from one distribution needs to be moved to match the other distribution.

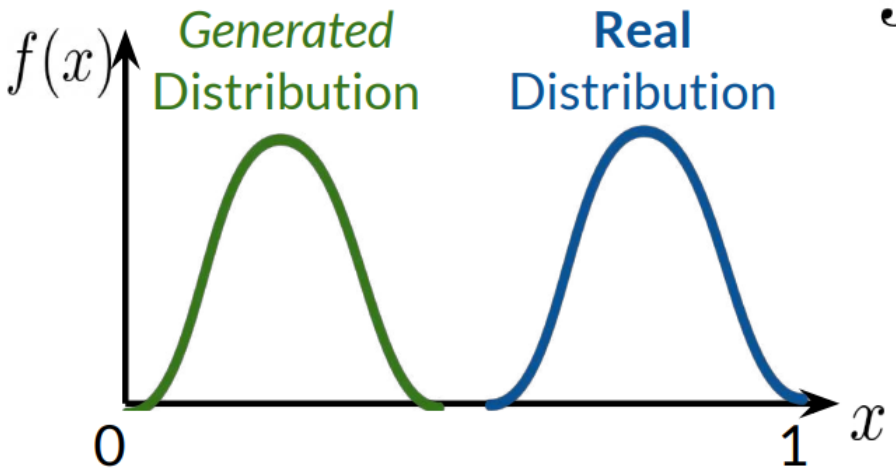


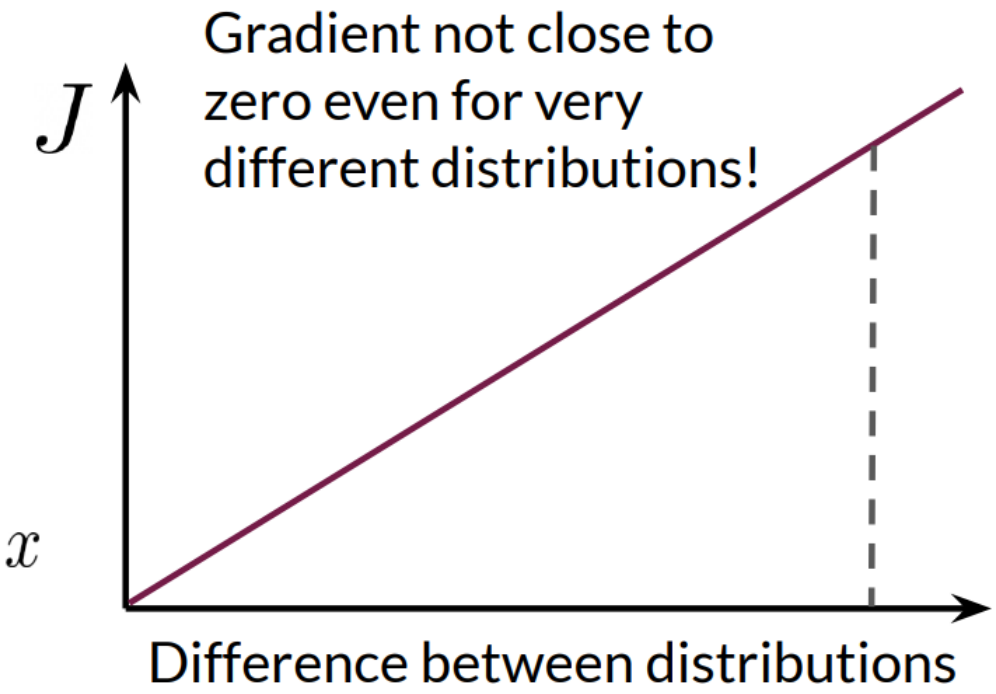
Effort to make the *generated* distribution equal to the **real** distribution

Depends on the distance and amount moved

Foundations of GAN

Earth Mover's Distance





Foundations of GAN

Wasserstein Loss - BCE Loss Simplified

- **Approximates Earth Mover's Distance (EMD)**: Measures the effort to transform the generated distribution into the real distribution.
- **Comparison to BCE Loss**: W-Loss is simpler and avoids the limitations of BCE Loss

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log h(x^{(i)},\theta) + (1-y^{(i)})\log(1-h(x^{(i)},\theta))]$$

$$\min_{d}\max_{g}$$

Discriminator

Minimize cost

Generator

Maximize cost

Foundations of GAN

Wasserstein Loss - BCE Loss Simplified

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log h(x^{(i)},\theta) + (1-y^{(i)})\log(1-h(x^{(i)},\theta))\right]$$

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

Minimize cost

Maximize cost

Foundations of GAN

Wasserstein Loss - BCE Loss Simplified

W-Loss : Approximates the Earth Movers distance

$$\min_{g} \max_{c} \; \mathbb{E}(c(x)) - \mathbb{E}(c(g(z)))$$

Generator

Minimize the distance

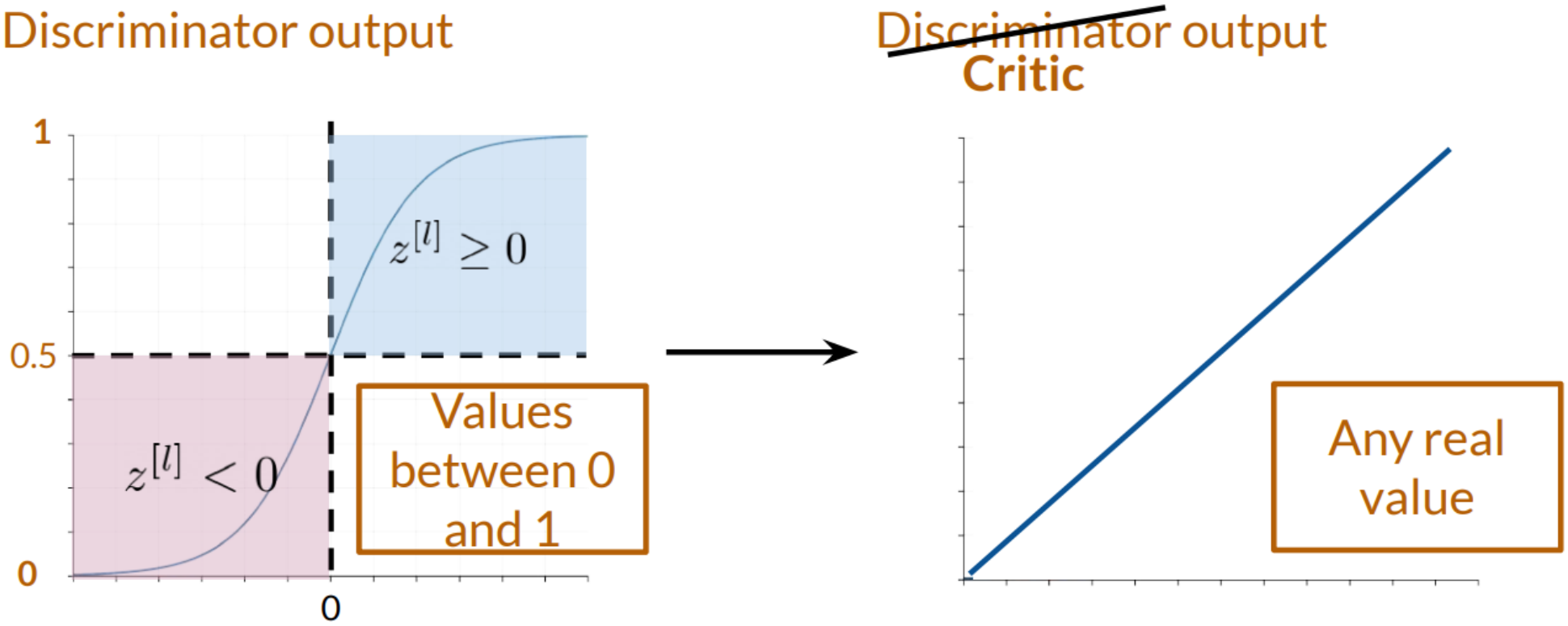
Critic

Maximize the distance

Foundations of GAN

Wasserstein Loss - BCE Loss Simplified

W-Loss : Approximates the Earth Movers distance



Foundations of GAN

Wasserstein Loss - BCE Loss Simplified

BCE Loss Vs W-Loss

BCE Loss	W-Loss
Discriminator outputs between 0 and 1	Critic outputs any number
$[\mathbb{E}(\log(d(x))) + \mathbb{E}(1 - \log(d(g(z))))]$	$\mathbb{E}(c(x)) - \mathbb{E}(c(g(z)))$

W-Loss helps with mode collapse and vanishing gradient problems

Foundations of GAN

Condition on Wasserstein Critic

$$\min_{g} \max_{c} \; \mathbb{E}(c(x)) - \mathbb{E}(c(g(z)))$$

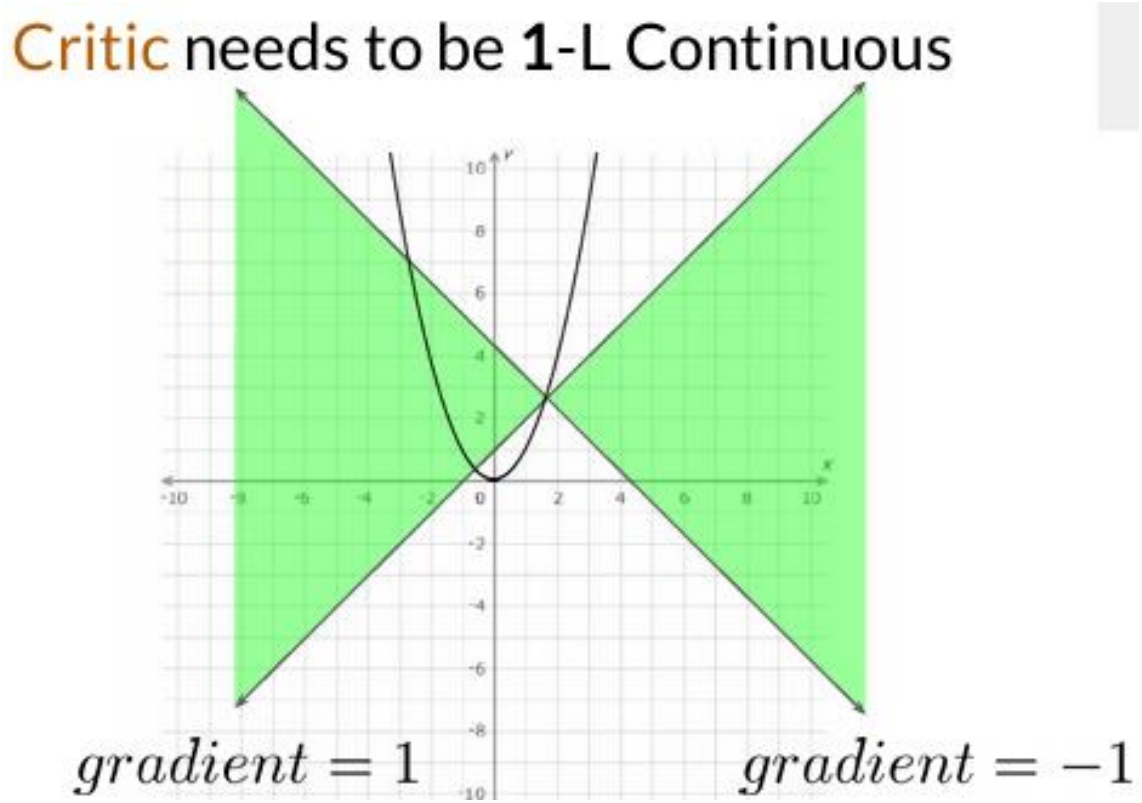
$$\min_{g} \max_{\boxed{c}} \; \mathbb{E}(\boxed{c}(x)) - \mathbb{E}(\boxed{c}(g(z)))$$

Needs to be 1-Lipschitz Continuous

Foundations of GAN

Condition on W-Loss

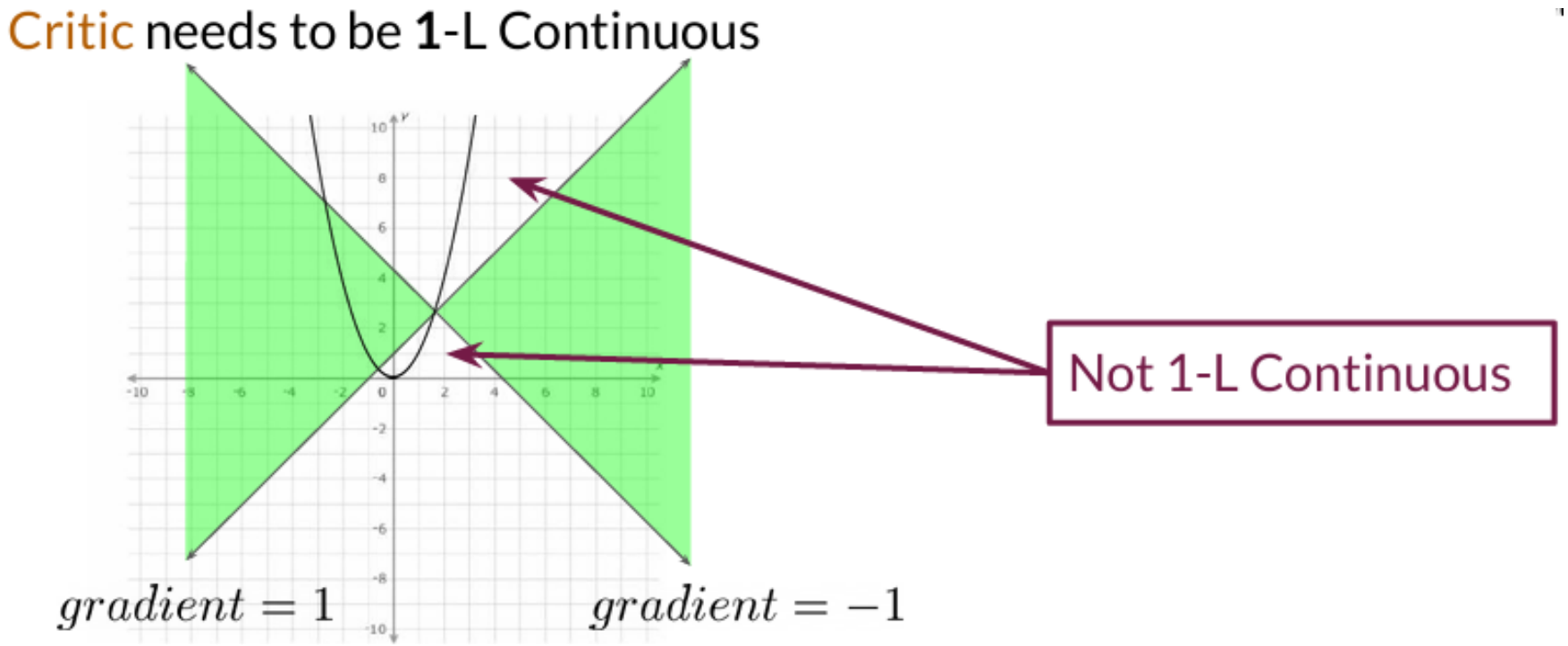
The norm of the gradient should be at most 1 for every point



Foundations of GAN

Condition on W-Loss

The norm of the gradient should be at most 1 for every point

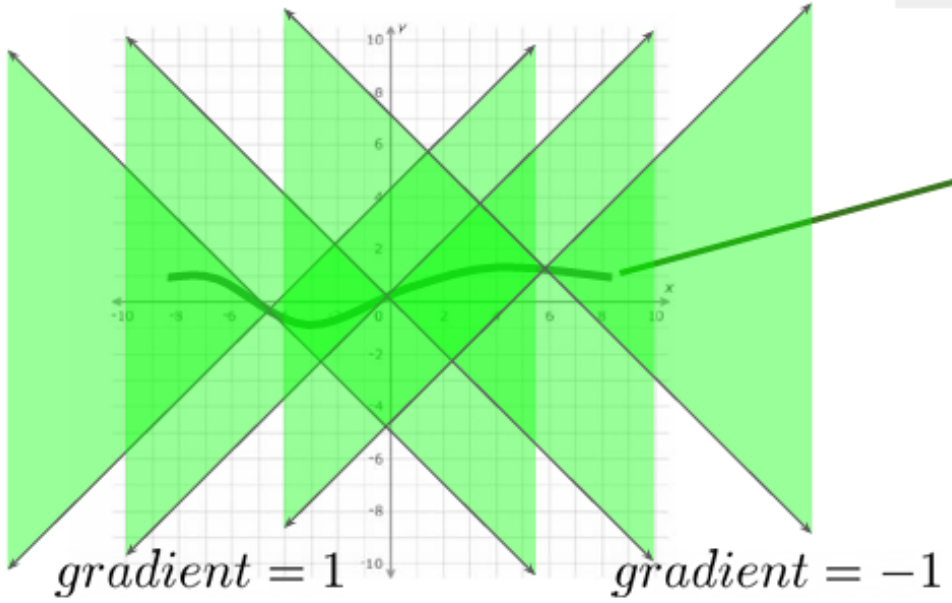


Foundations of GAN

Condition on W-Loss

Critic needs to be **1**-L Continuous

The norm of the gradient should be at most **1** for *every point*



1-L Continuous

↓

W-Loss is valid

Needed for training stable neural networks with W-Loss

Summary :

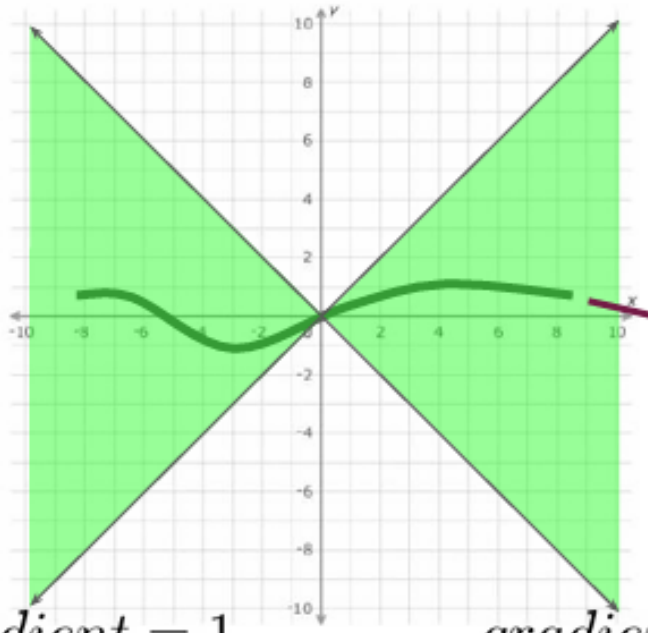
- Critic's neural network needs to be 1-L Continuous when using W-Loss
- This condition ensures that W-Loss is validly approximating Earth Mover's Distance

Foundations of GAN

1-Lipschitz Continuity Enforcement

1-L Enforcement

Critic needs to be 1-L Continuous



$gradient = 1$ $gradient = -1$

Norm of the gradient at most 1

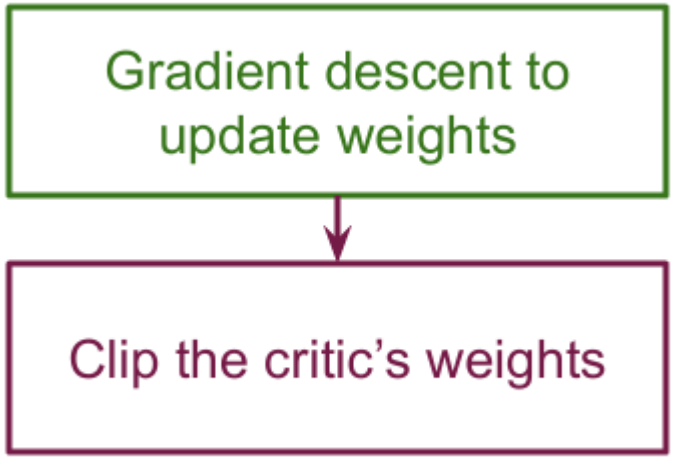
$$||\nabla f(x)||_2 \leq 1$$

Slope of the function at most 1

Foundations of GAN

1-Lipschitz Continuity Enforcement

- 1-L Enforcement: Weight Clipping Weight clipping forces the weights of the critic to a fixed interval
- Limits the learning ability of the critic



Foundations of GAN

1-Lipschitz Continuity Enforcement

- 1-L Enforcement: Gradient Penalty

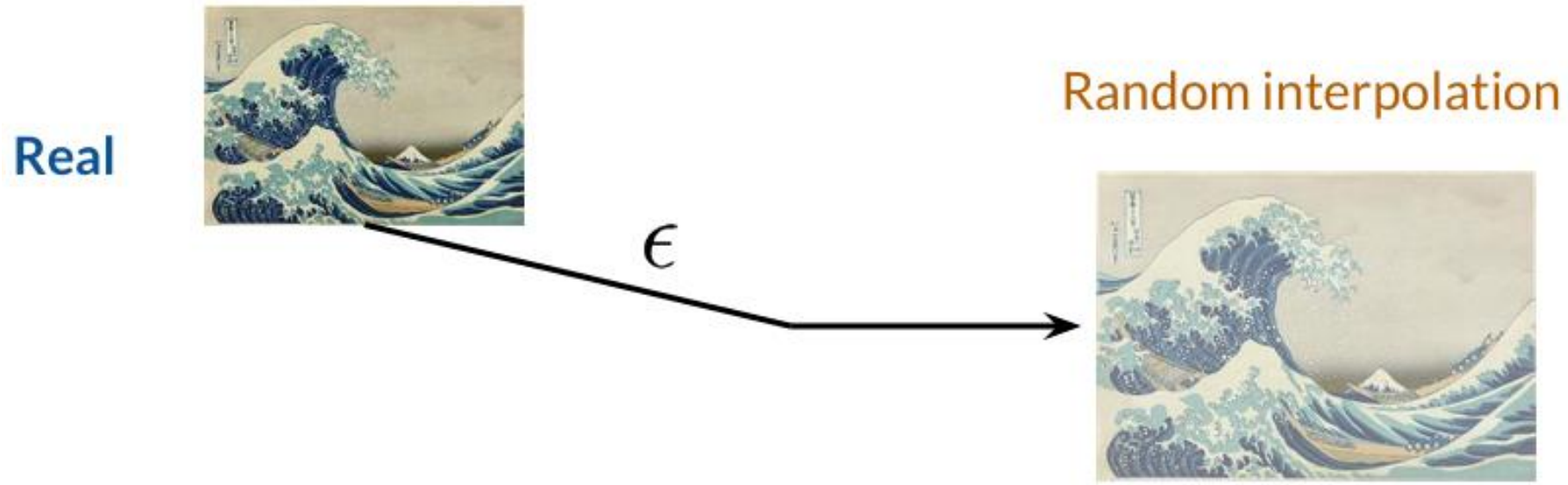
$$\min_{g} \max_{c} \; \mathbb{E}(c(x)) - \mathbb{E}(c(g(z))) + \lambda \text{reg}$$

Regularization of the critic's gradient

Foundations of GAN

1-L Enforcement: Gradient Penalty

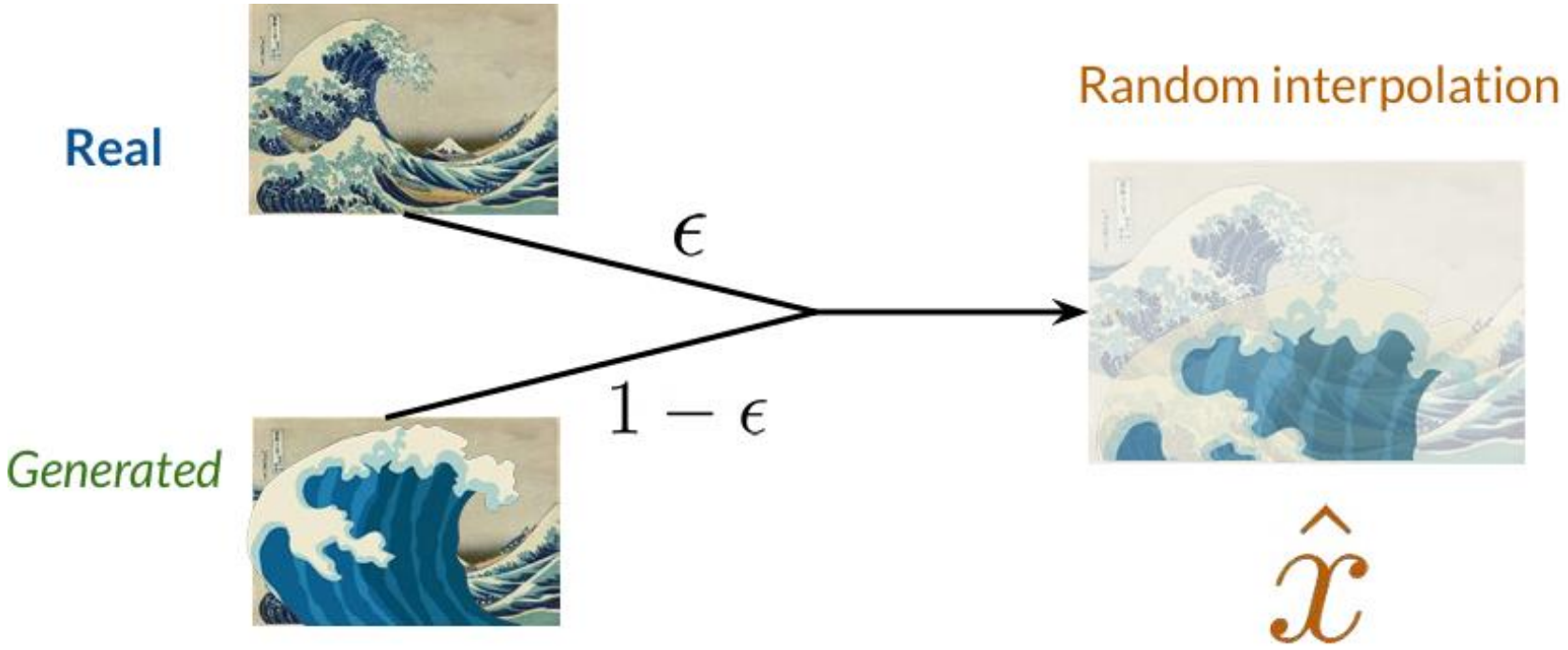
- 1-L Enforcement: Gradient Penalty



Foundations of GAN

1-L Enforcement: Gradient Penalty

- 1-L Enforcement: Gradient Penalty



Foundations of GAN

1-L Enforcement: Gradient Penalty

- 1-L Enforcement: Gradient Penalty

$$\mathbb{E}(||\nabla c(\hat{x})||_2 - 1)^2$$

Regularization term

$$\epsilon x + (1 - \epsilon) g(z)$$

Real

Generated

Interpolation

Foundations of GAN

Putting It All Together

- 1-L Enforcement: Gradient Penalty

$$\min_g \max_c \mathbb{E}(c(x)) - \mathbb{E}(c(g(z))) + \lambda \mathbb{E}(||\nabla c(\hat{x})||_2 - 1)^2$$

- Makes the GAN less prone to mode collapse and vanishing gradient
- Tries to make the critic be 1-L Continuous, for the loss function to be continuous and differentiable

Summary

- Weight clipping and gradient penalty are ways to enforce 1-L continuity
- Gradient penalty tends to work better

Foundations of GAN

Demo: Wasserstein GAN with Gradient Penalty (WGAN-GP)

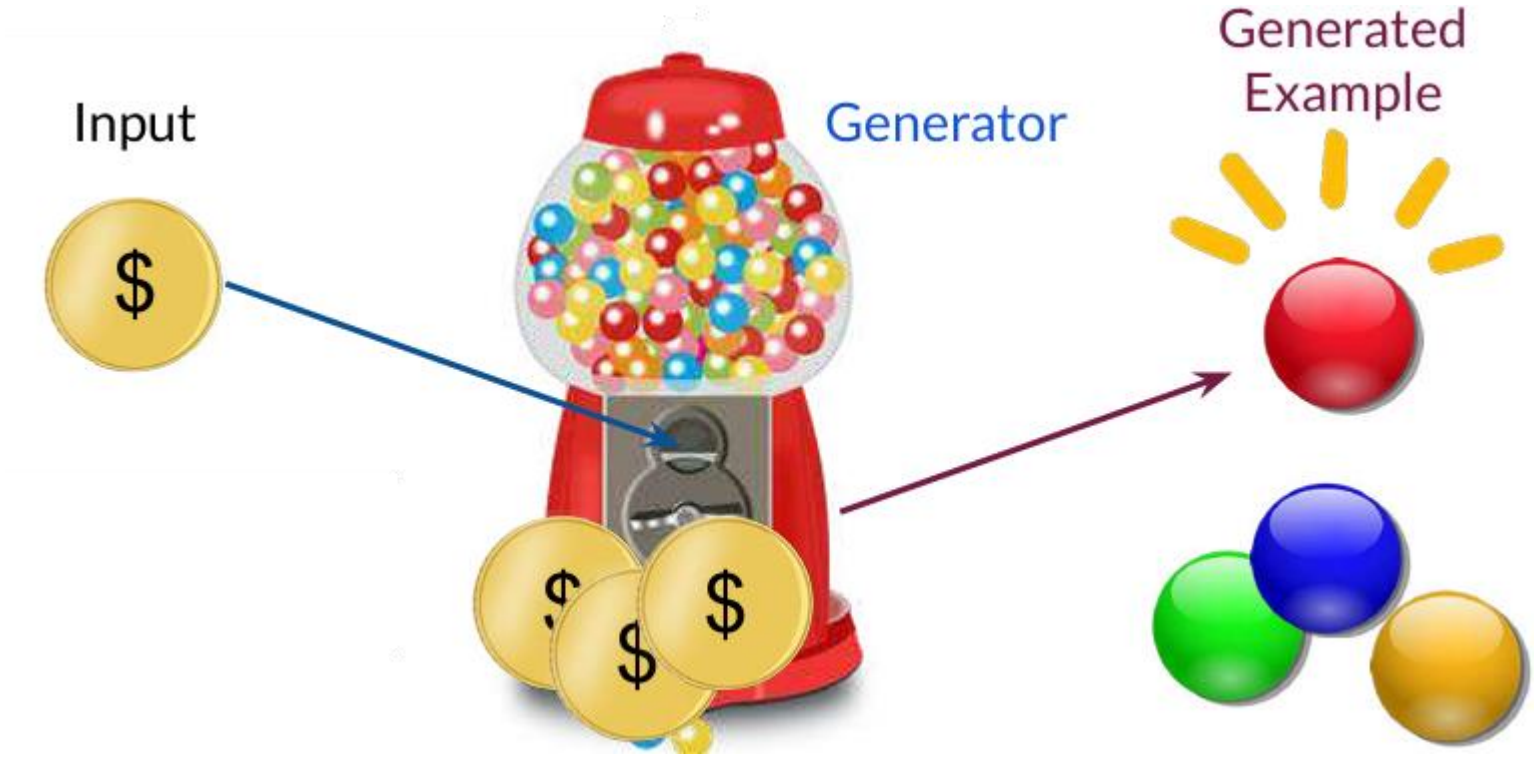
Dataset: MNIST

Foundations of GAN

Conditional and Controllable GAN

Unconditional Generation

- Get outputs from a random class

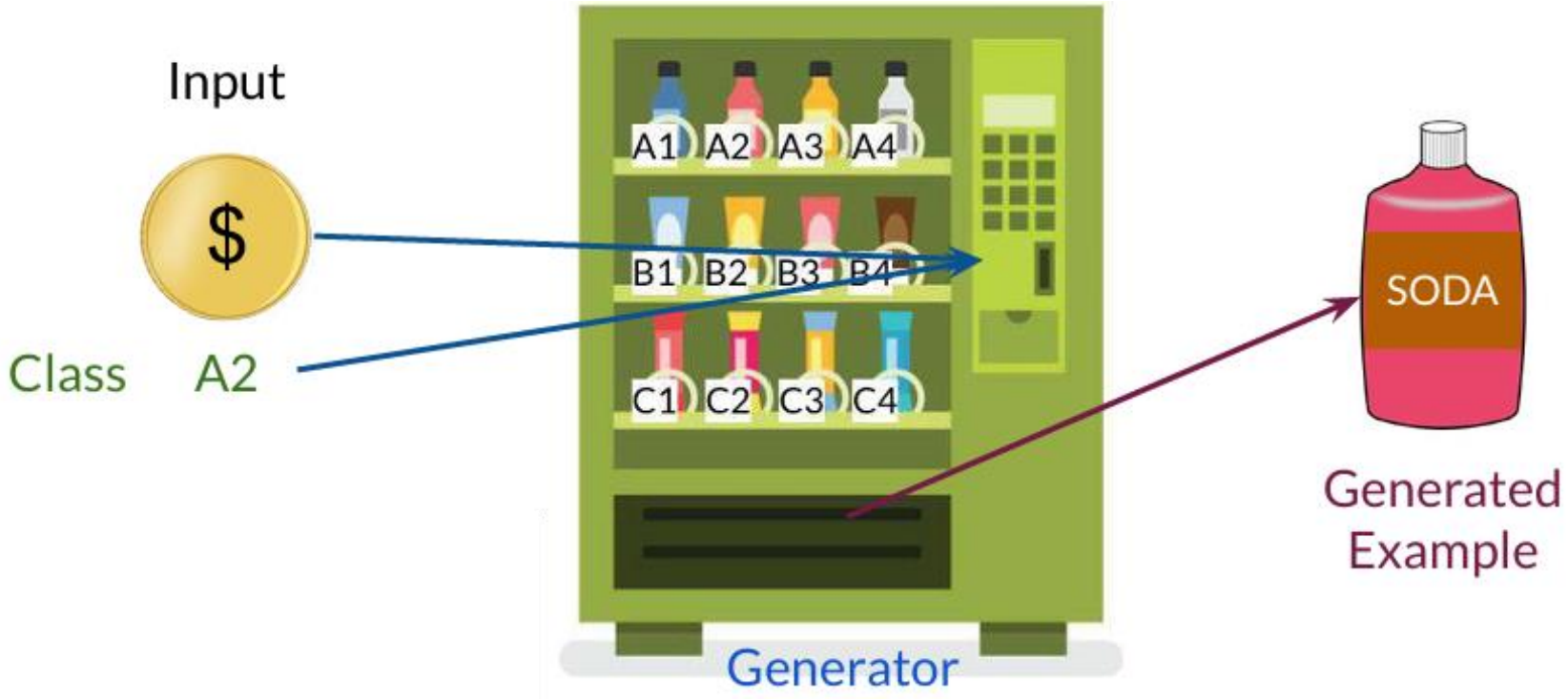


Foundations of GAN

Conditional and Controllable GAN

Conditional Generation

- You get what you ask for



Foundations of GAN

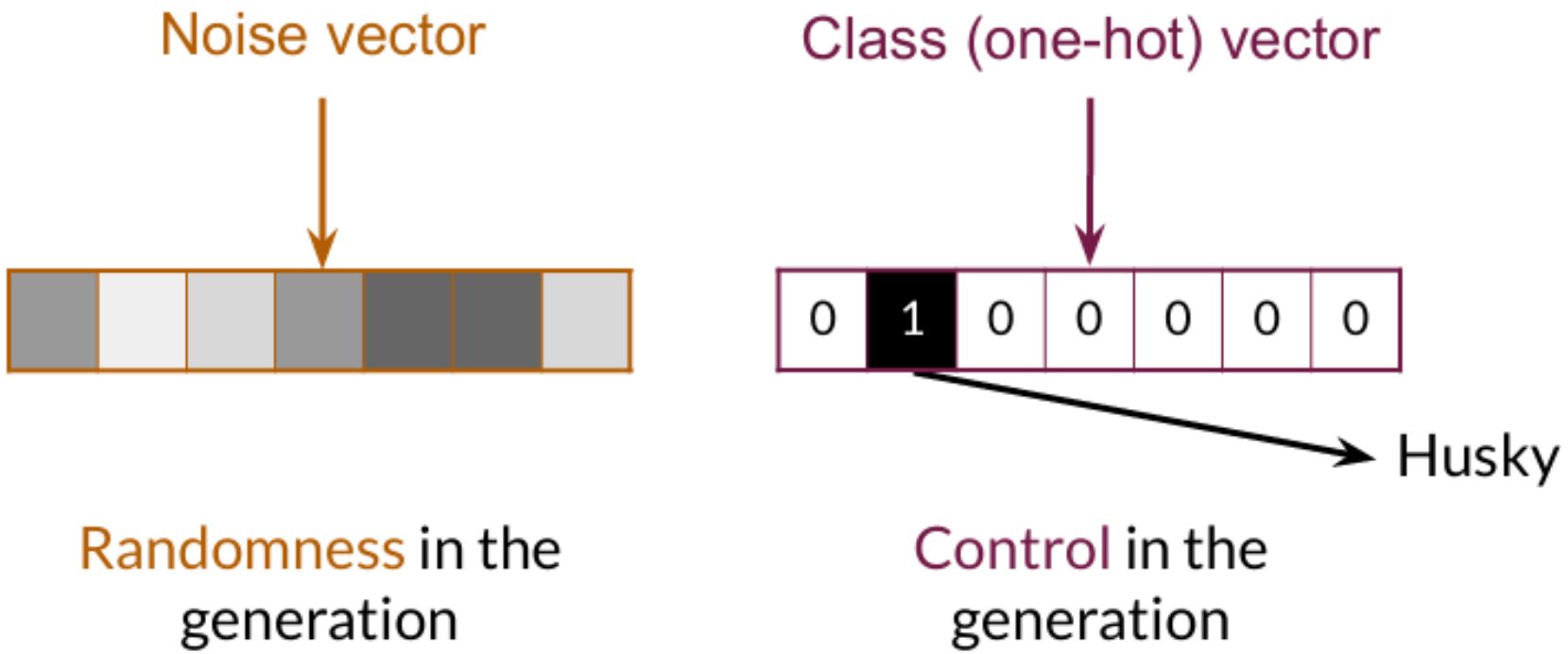
Conditional and Controllable GAN

Conditional vs. Unconditional Generation

Conditional	Unconditional
Examples from **the classes you want**	Examples from random classes
Training dataset needs to be labeled	Training dataset doesn't need to be labeled

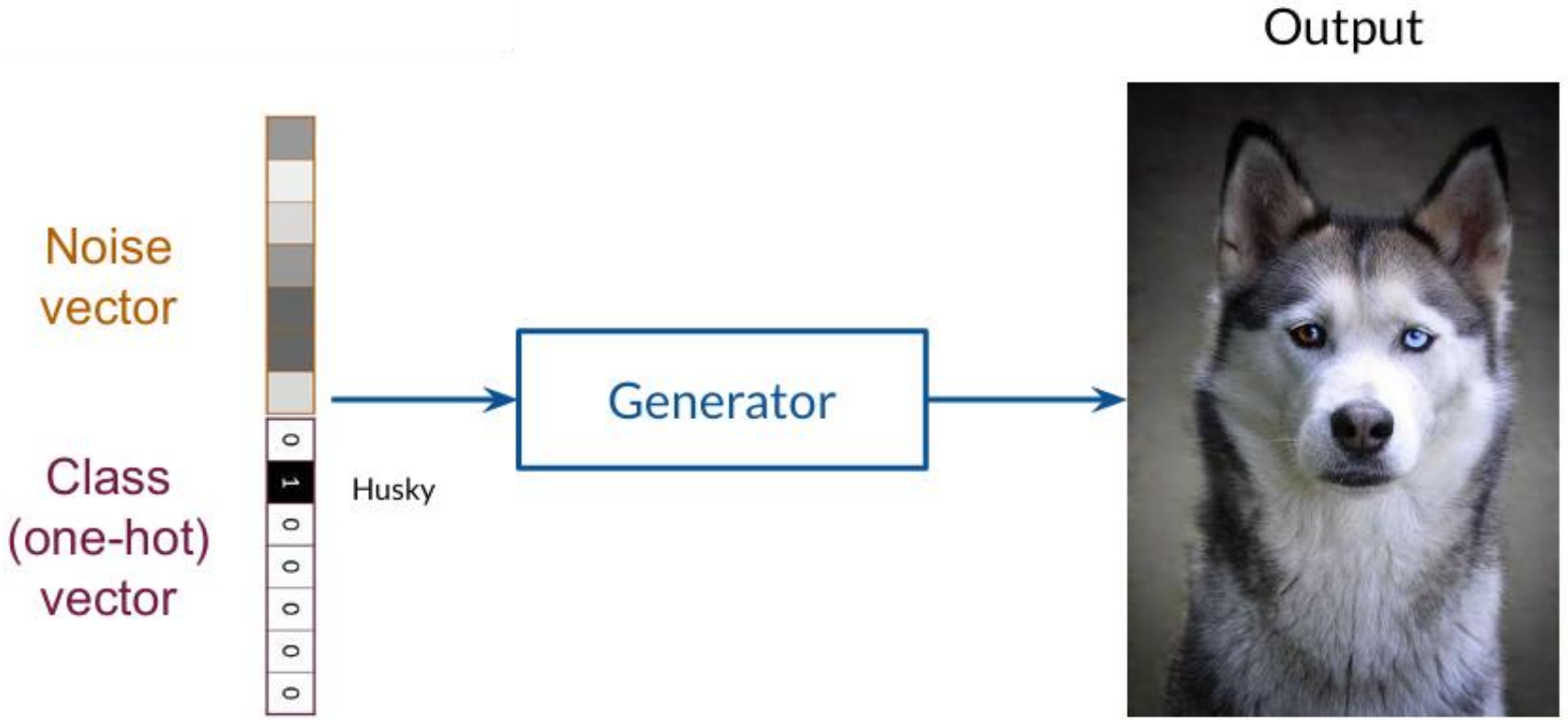
Foundations of GAN

Generation Input



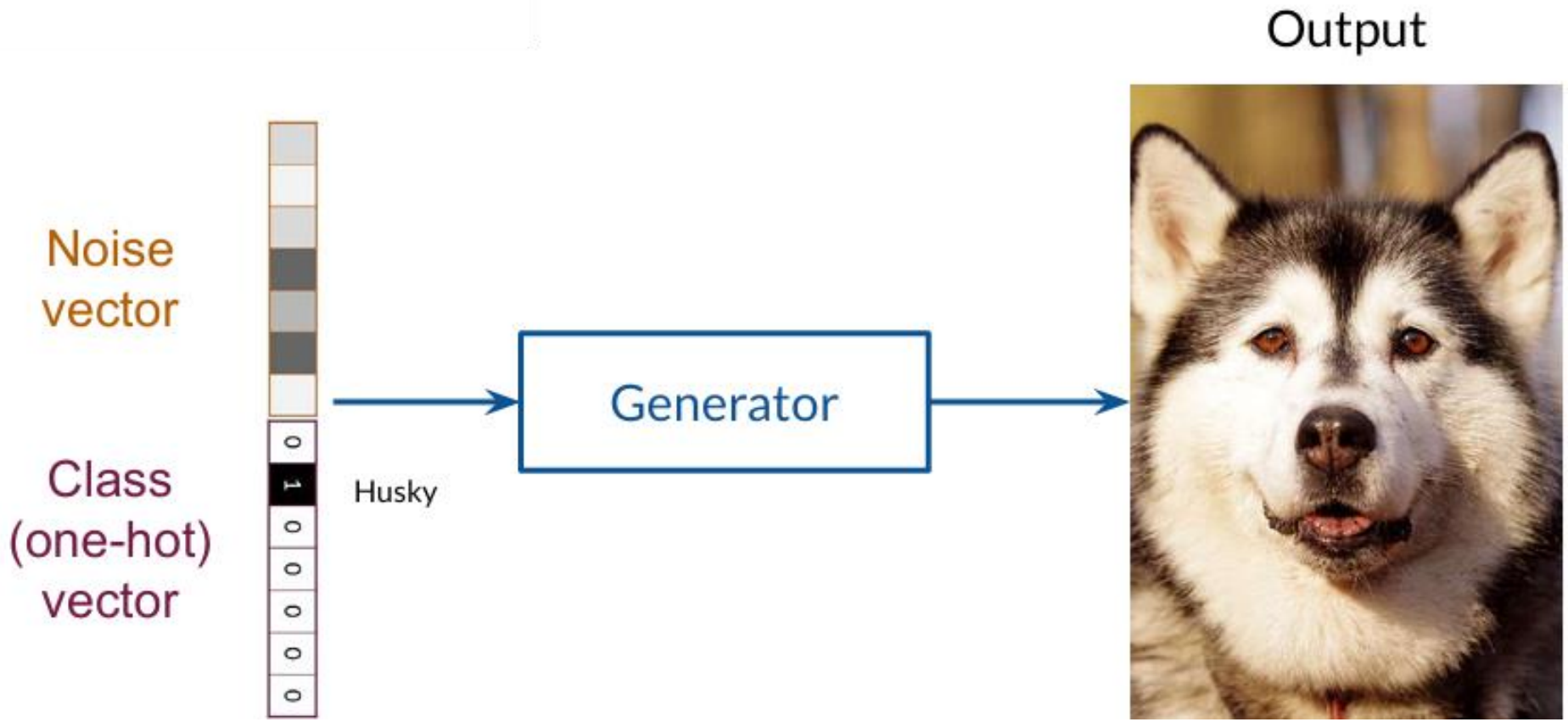
Foundations of GAN

Generation Input



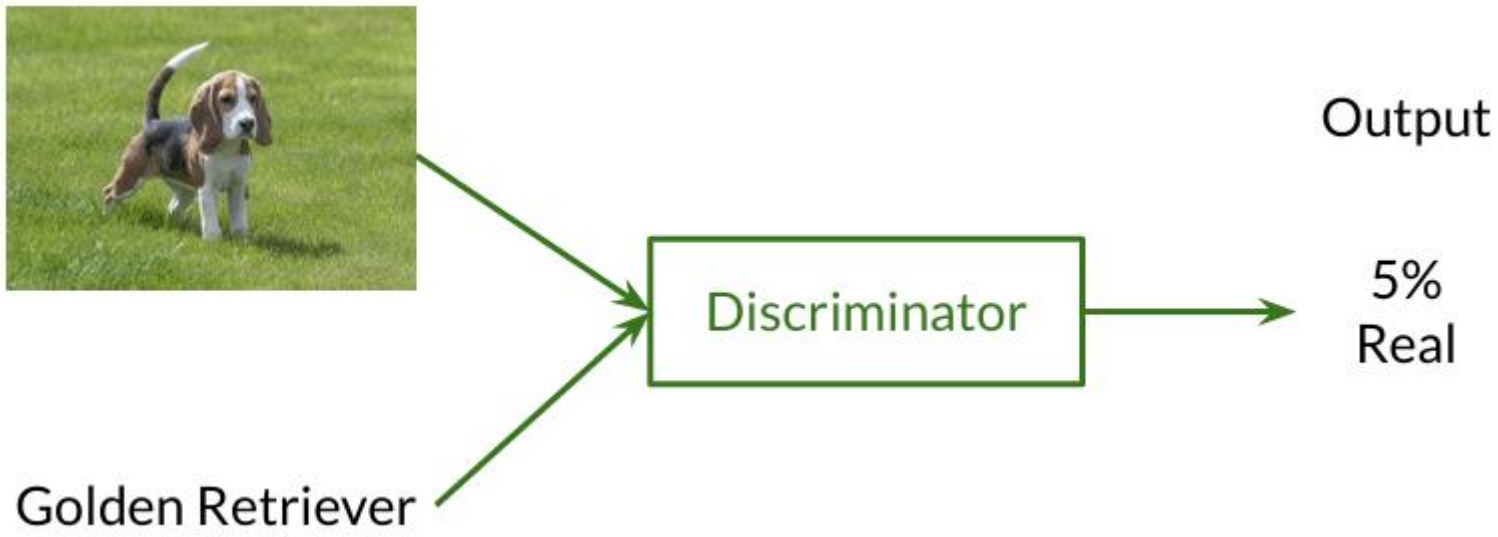
Foundations of GAN

Generation Input



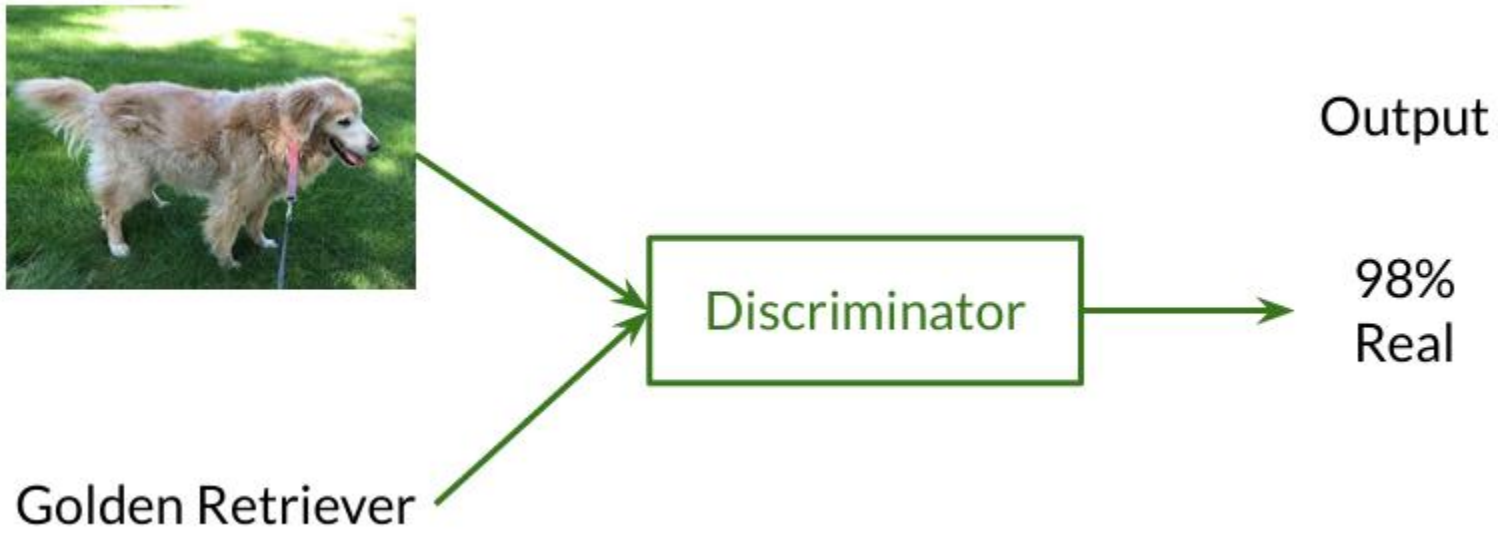
Foundations of GAN

Discriminator Input



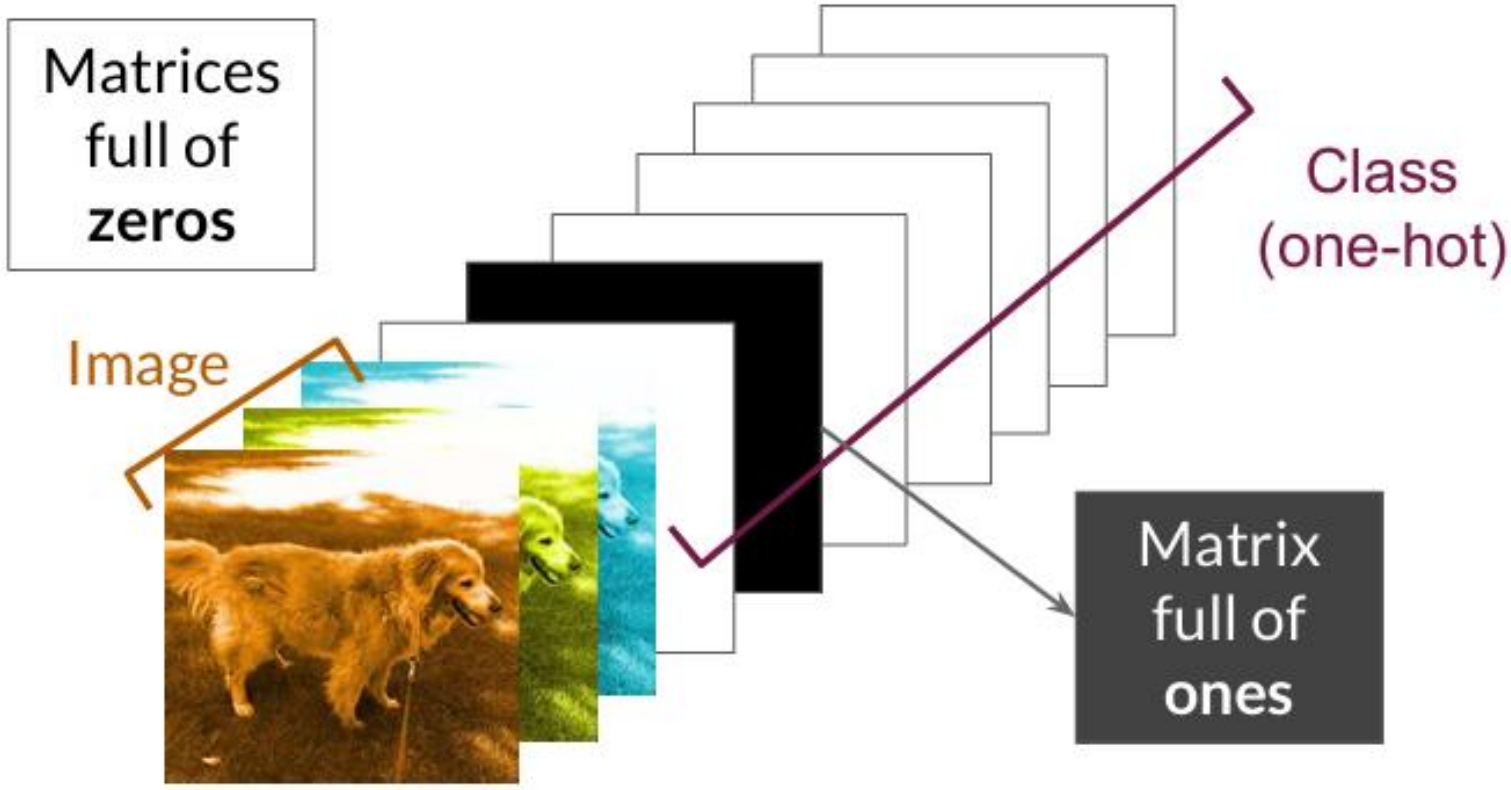
Foundations of GAN

Discriminator Input



Foundations of GAN

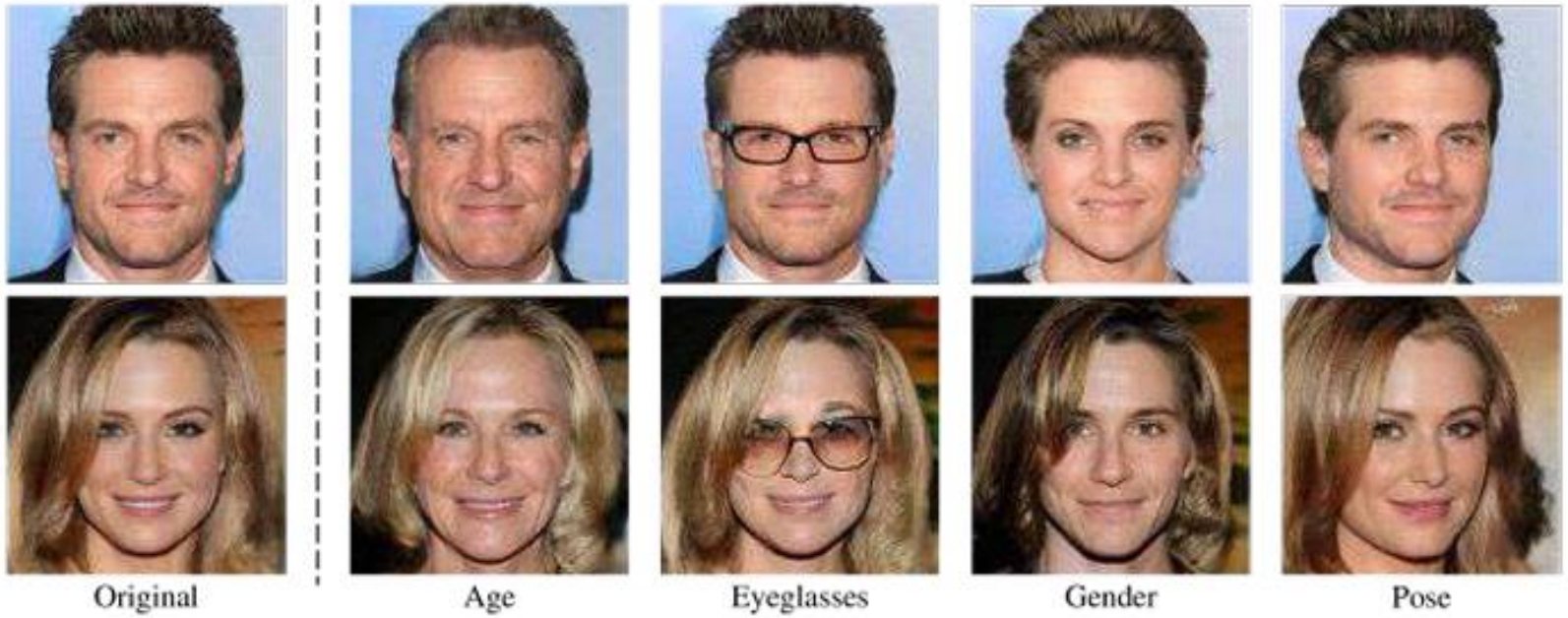
Discriminator Input



Foundations of GAN

Controllable Generation -GAN

Change specific features of the output

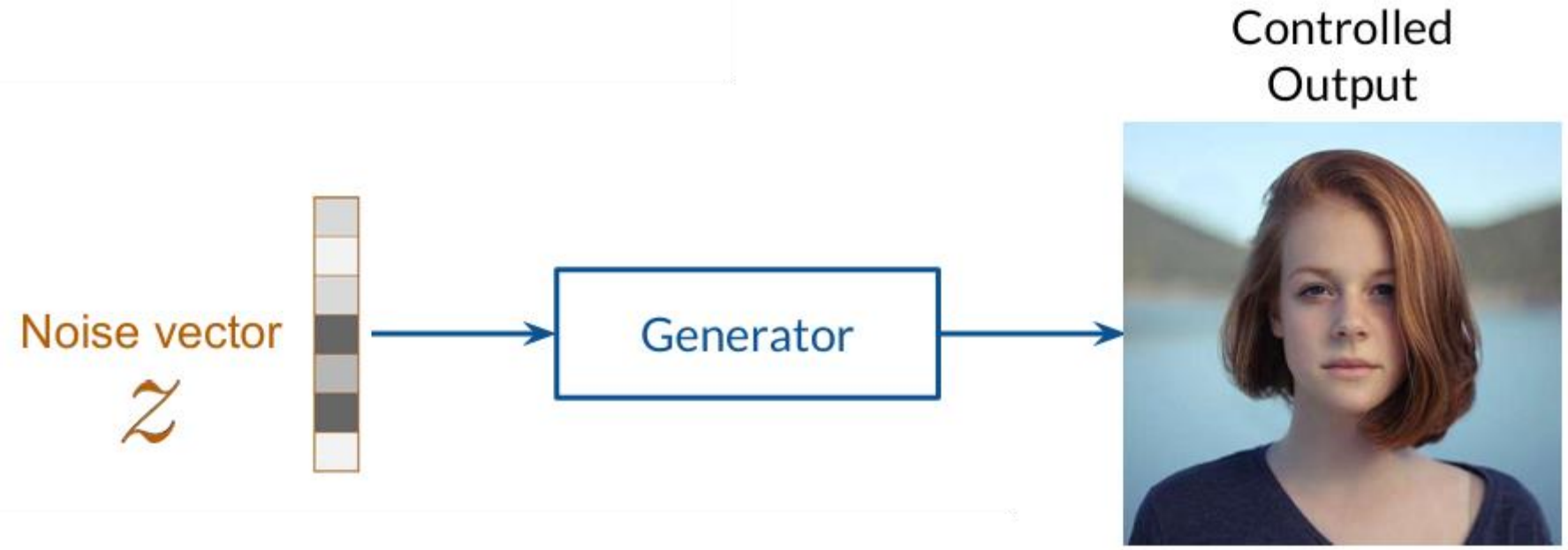


Available from: https://arxiv.org/abs/1907.10786

Foundations of GAN

Controllable Generation -GAN

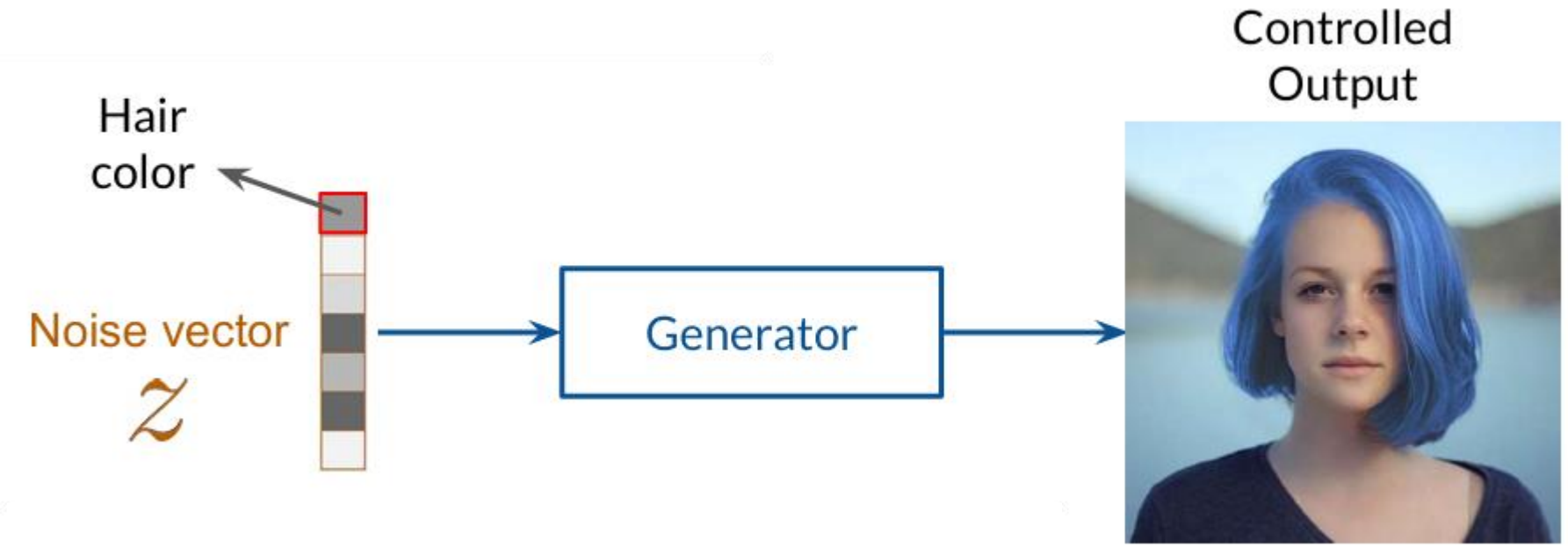
Tweak the input noise vector to get different features on the output



Foundations of GAN

Controllable Generation -GAN

Tweak the input noise vector to get different features on the output



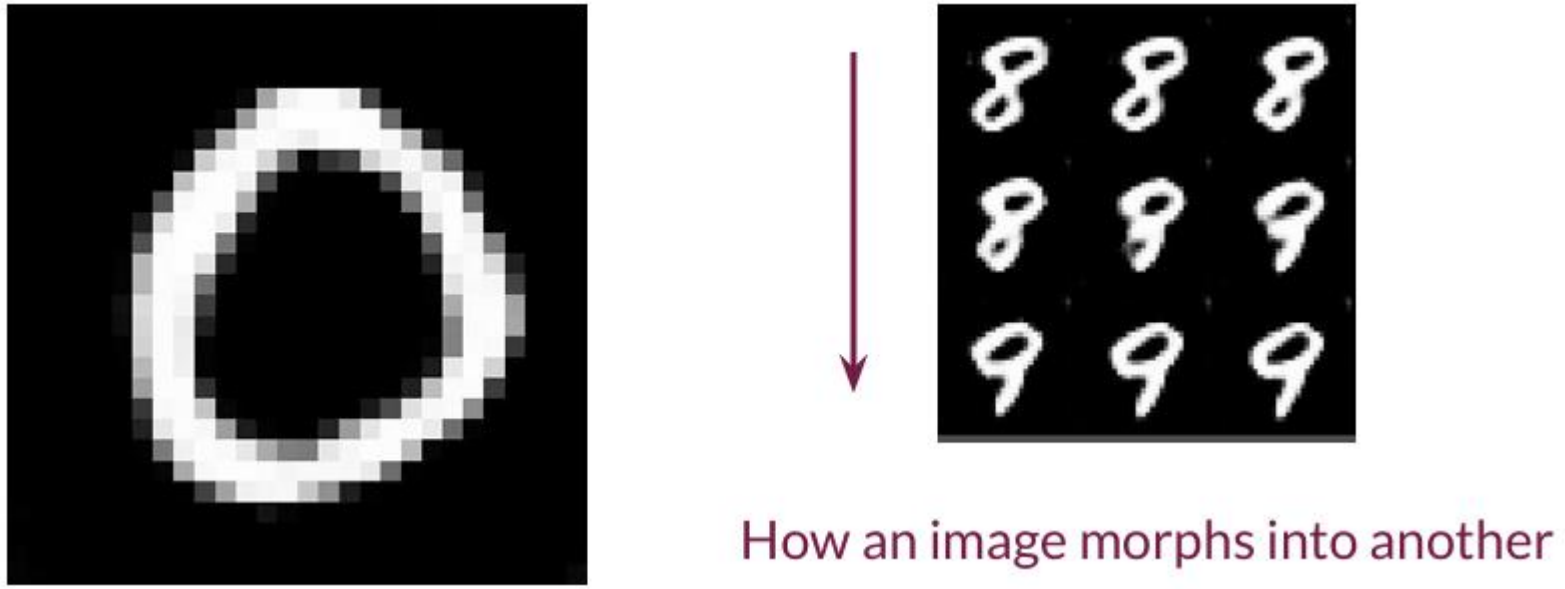
Foundations of GAN

Controllable Generation vs. Conditional Generation

Controllable	Conditional
Examples with the features that you want	Examples from the classes you want
Training dataset doesn't need to be labeled	Training dataset needs to be labeled
Manipulate the z vector input	Append a class vector to the input

Foundations of GAN

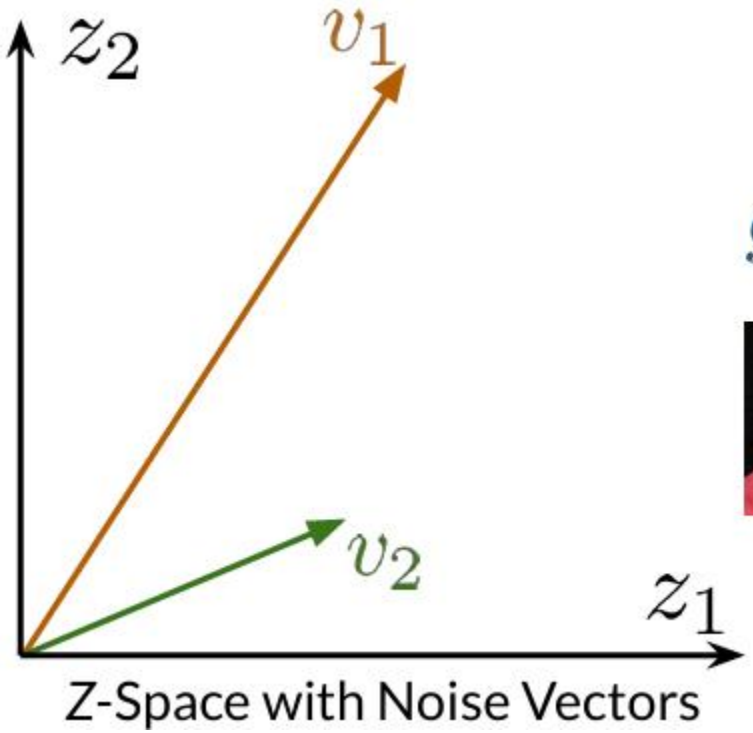
Interpolation Using the Z-Space



How an image morphs into another

Foundations of GAN

Interpolation Using the Z-Space

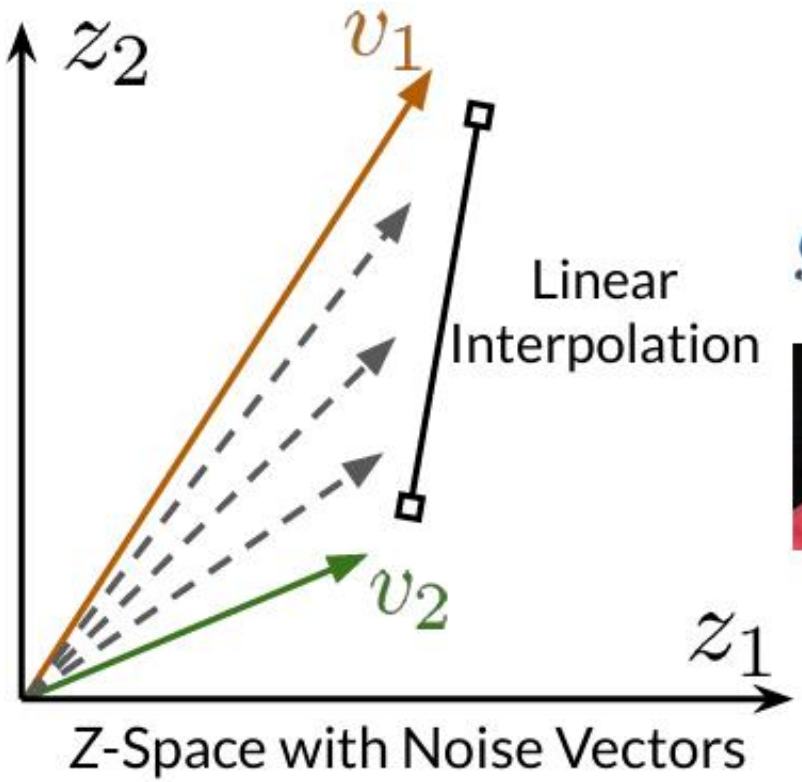


$g(v_1)$

$g(v_2)$

Foundations of GAN

Interpolation Using the Z-Space

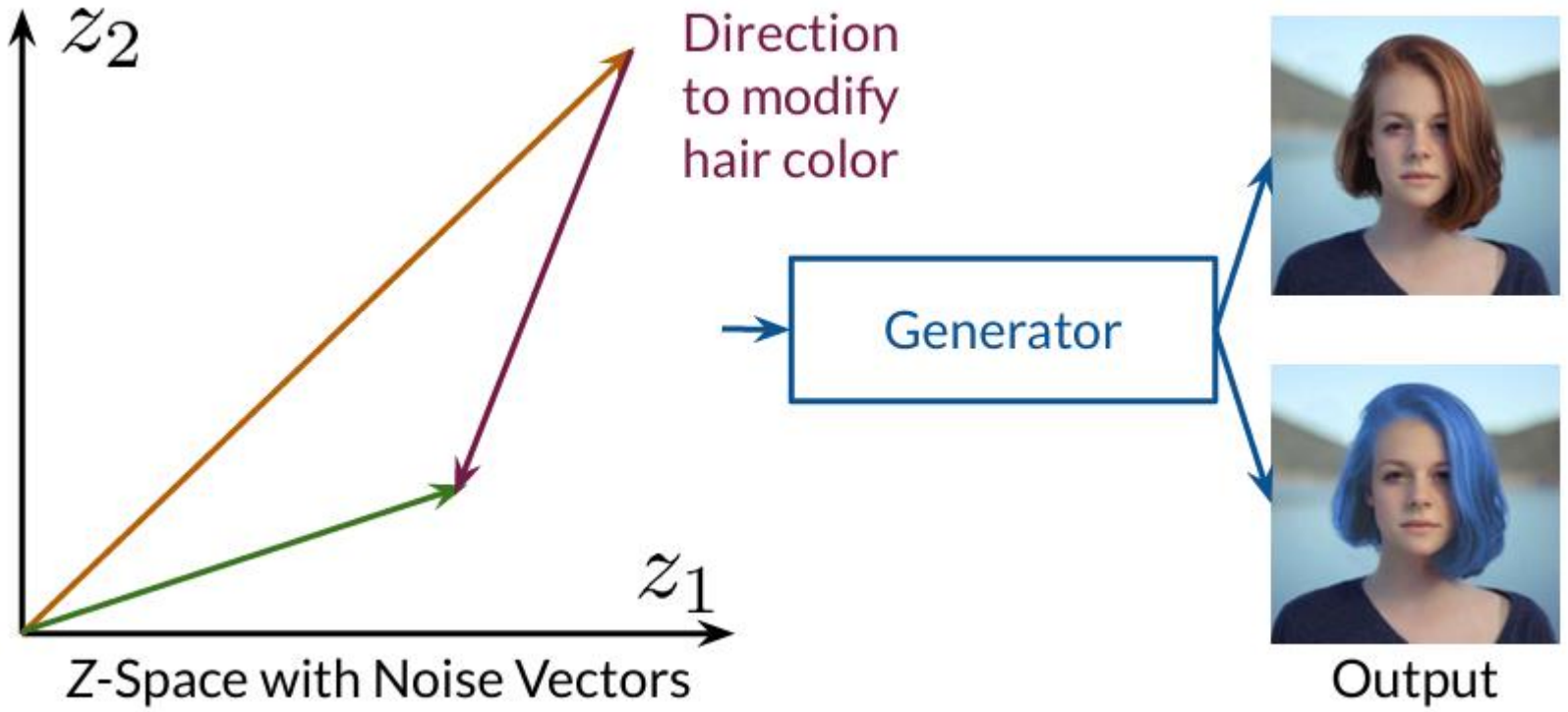


Intermediate images using the Z-space

$g(v_1)$ ⟷ $g(v_2)$

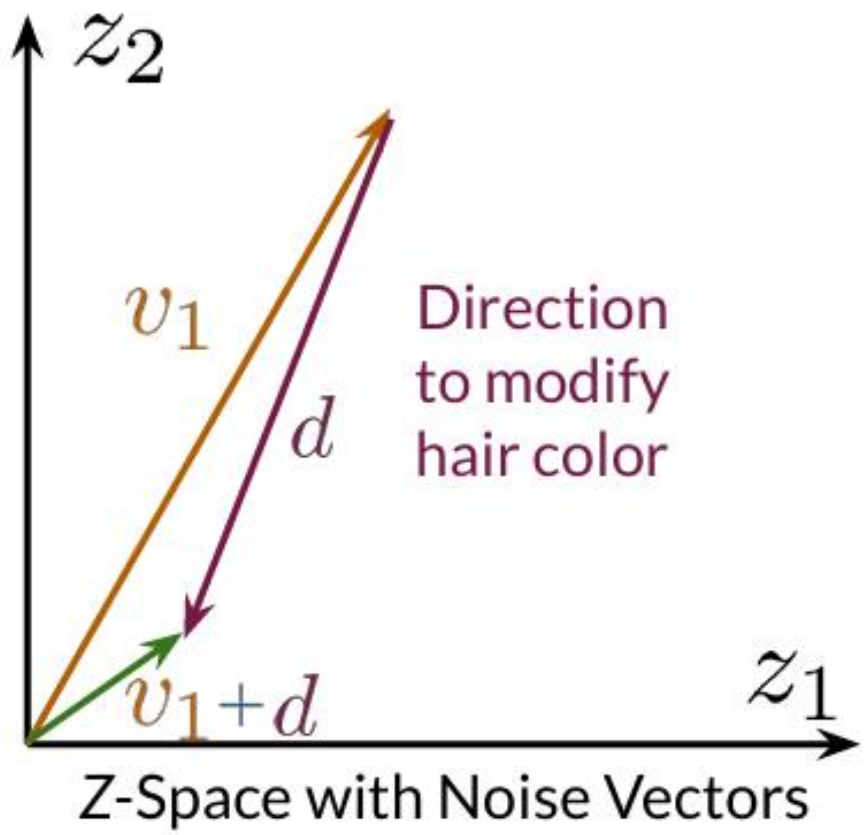
Foundations of GAN

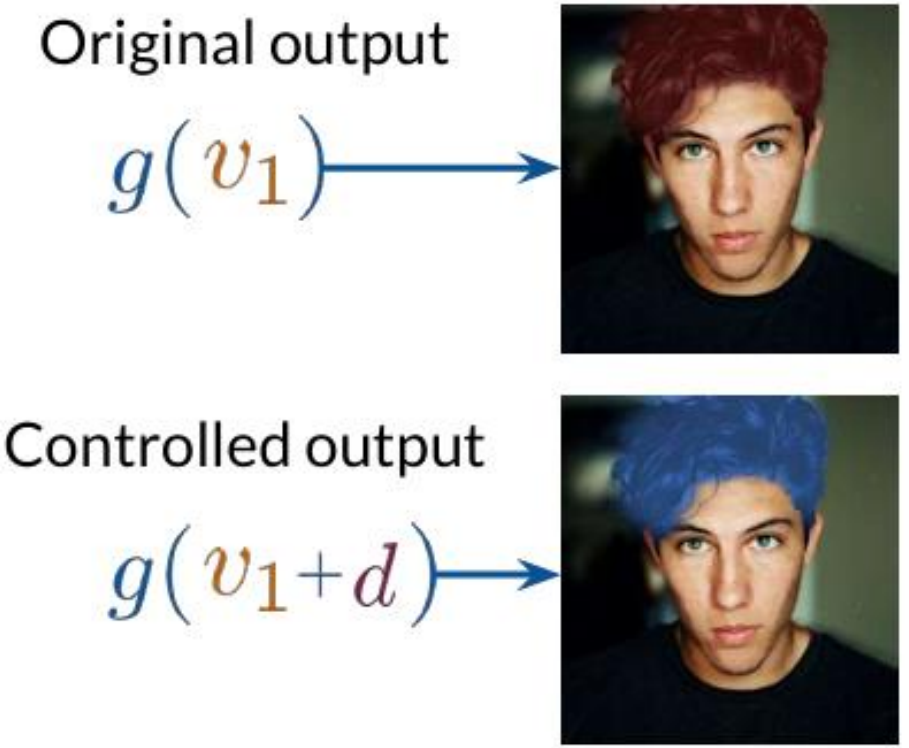
Z-Space and Controllable Generation



Foundations of GAN

Z-Space and Controllable Generation

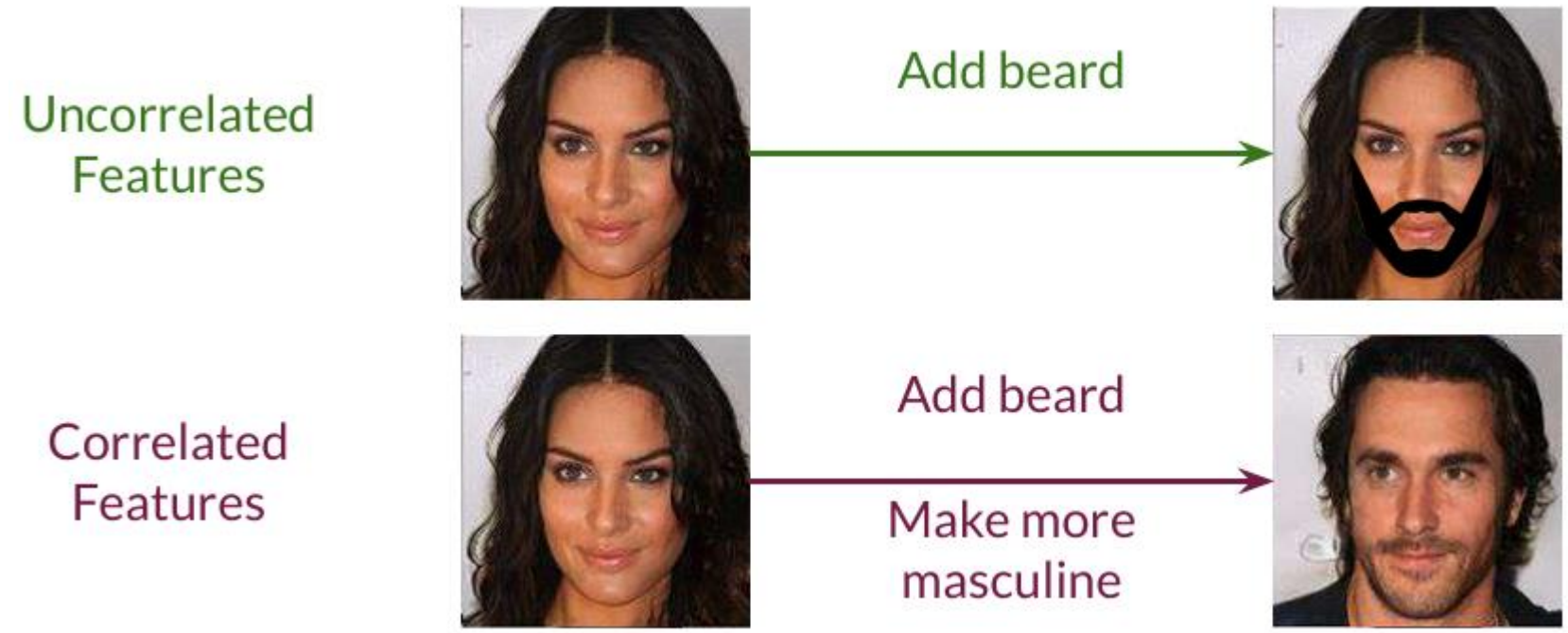




Foundations of GAN

Challenges with Controllable Generation

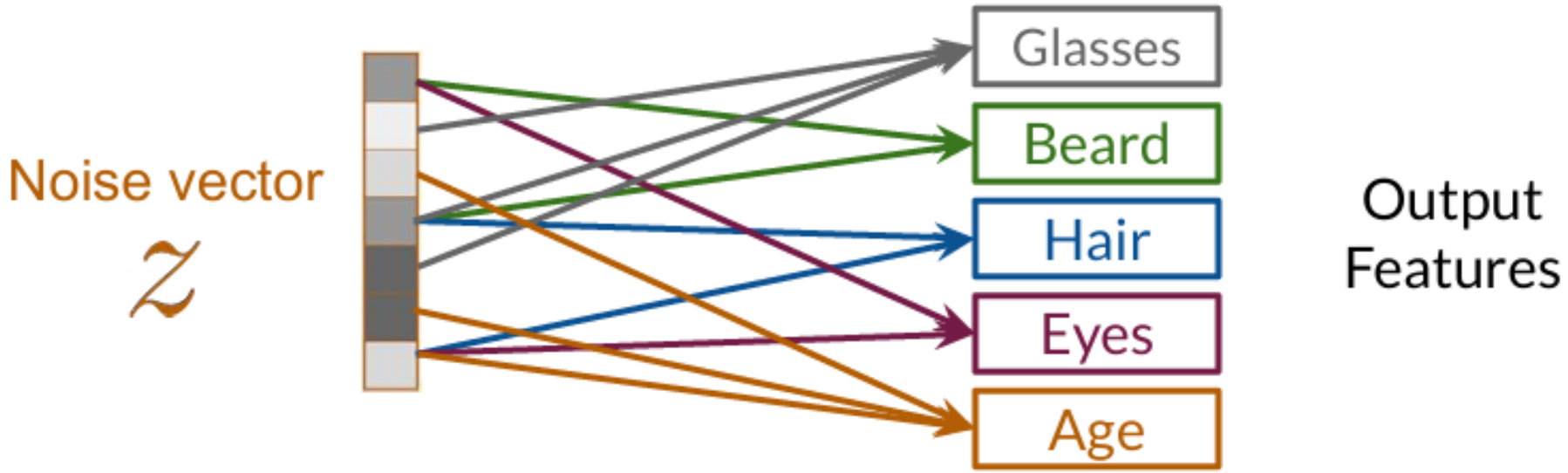
Feature Correlation



Foundations of GAN

Challenges with Controllable Generation

Z-Space Entanglement

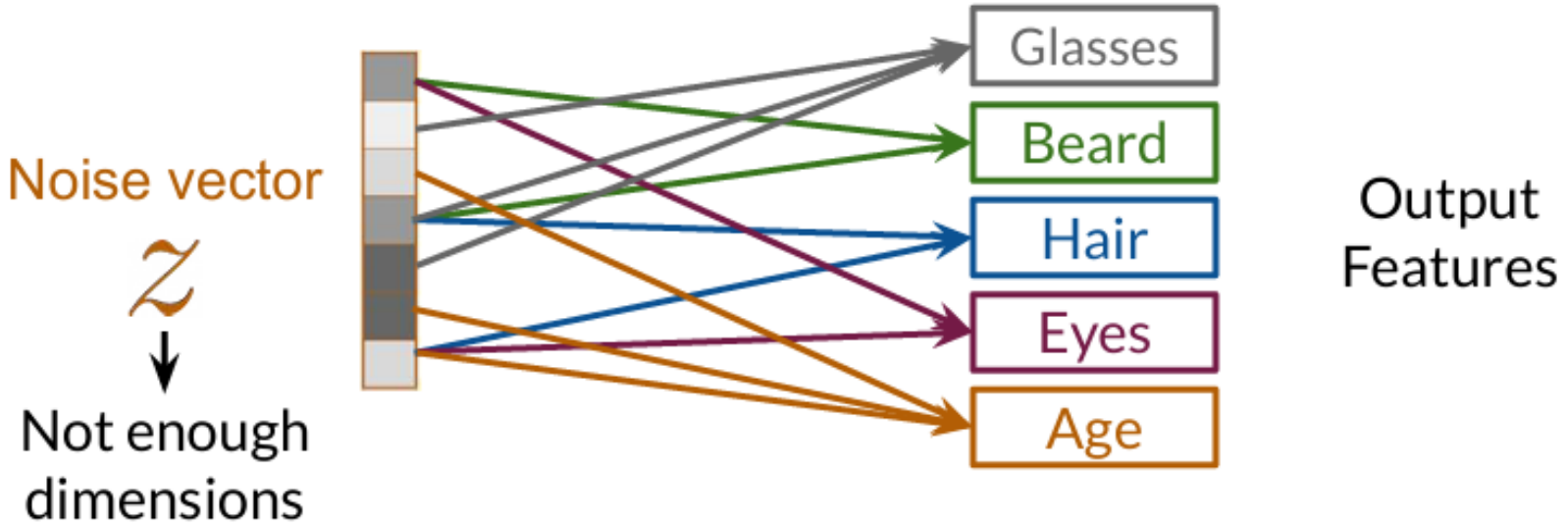


Foundations of GAN

Challenges with Controllable Generation

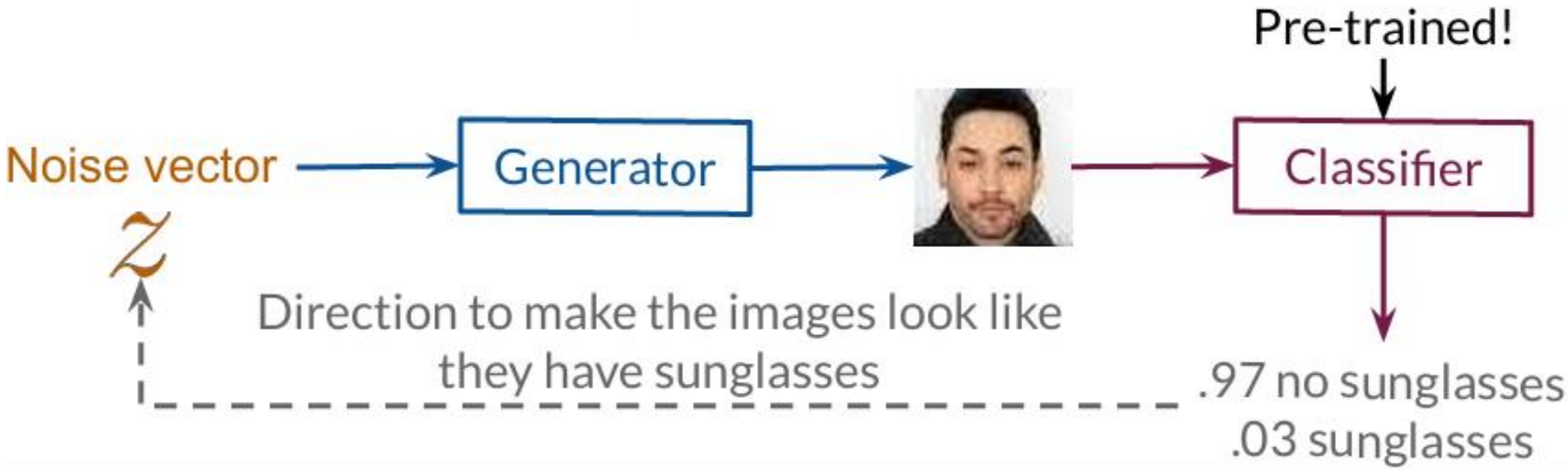
Z-Space Entanglement

It is not possible to control single output features



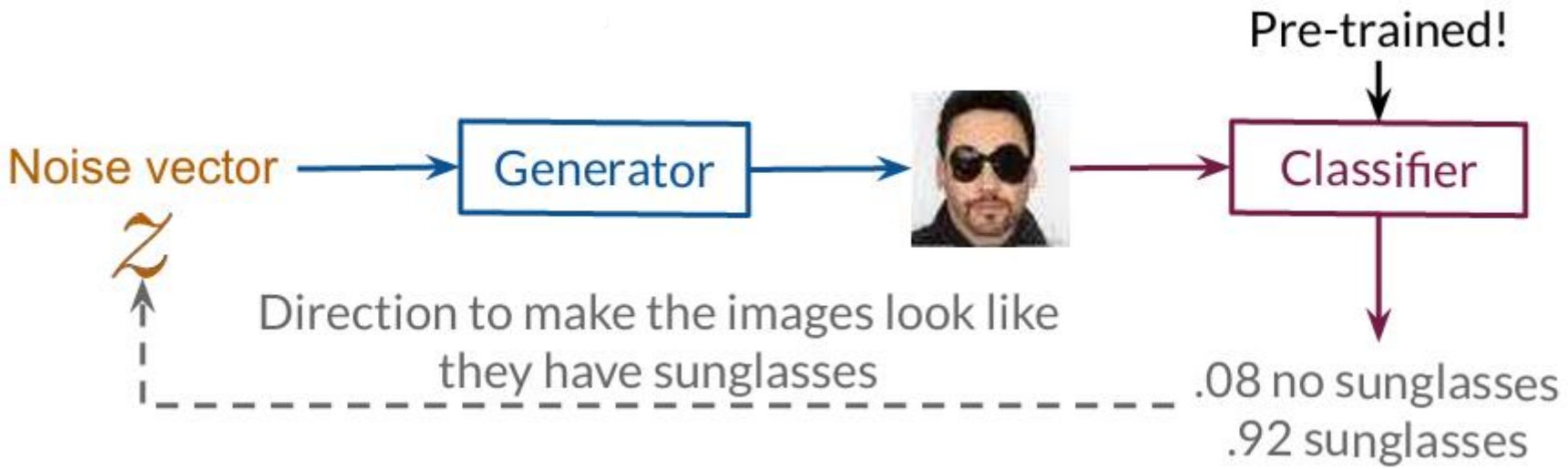
Foundations of GAN

Classifier Gradients



Foundations of GAN

Classifier Gradients

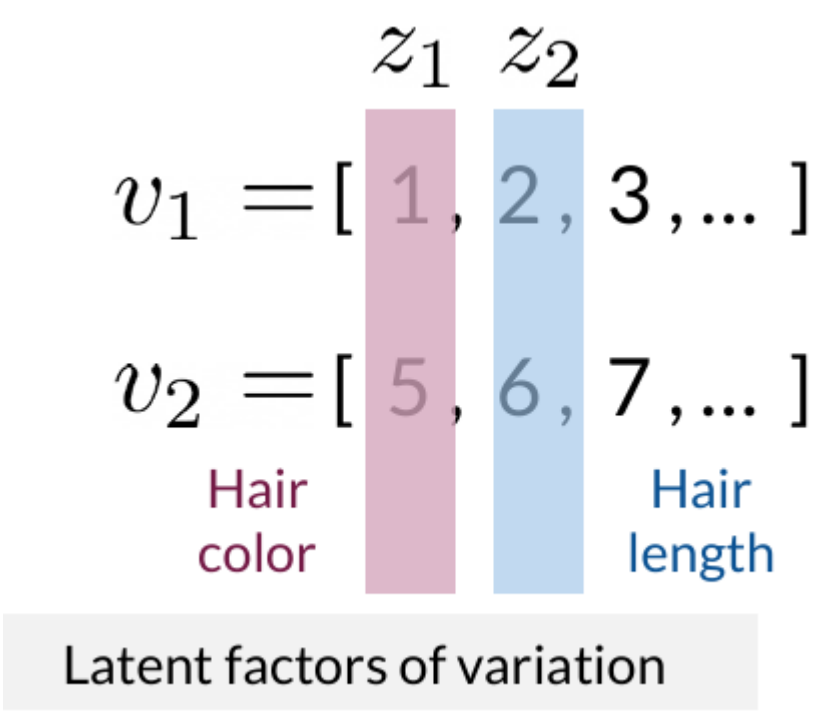


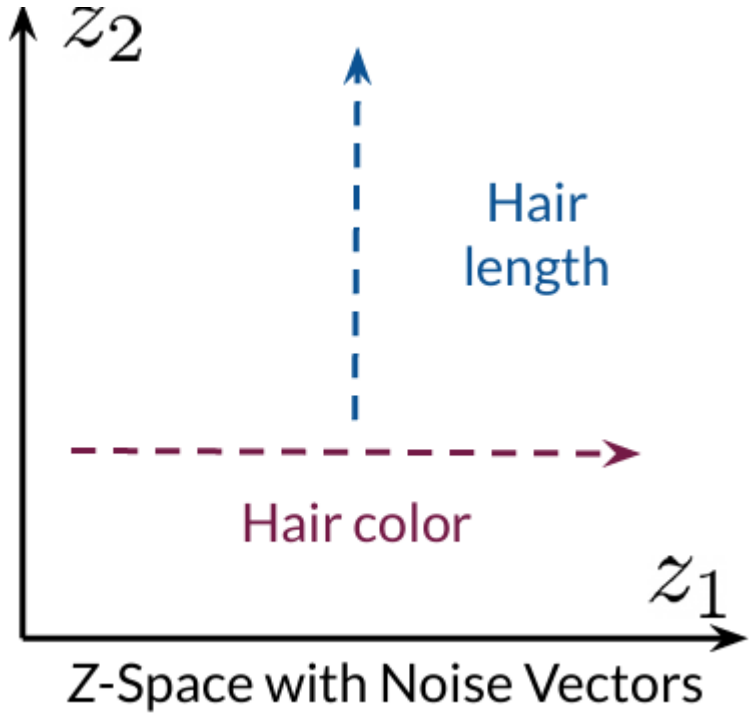
Modify just the noise vector until the feature emerges

Foundations of GAN

Disentanglement

Disentangled Z-Space

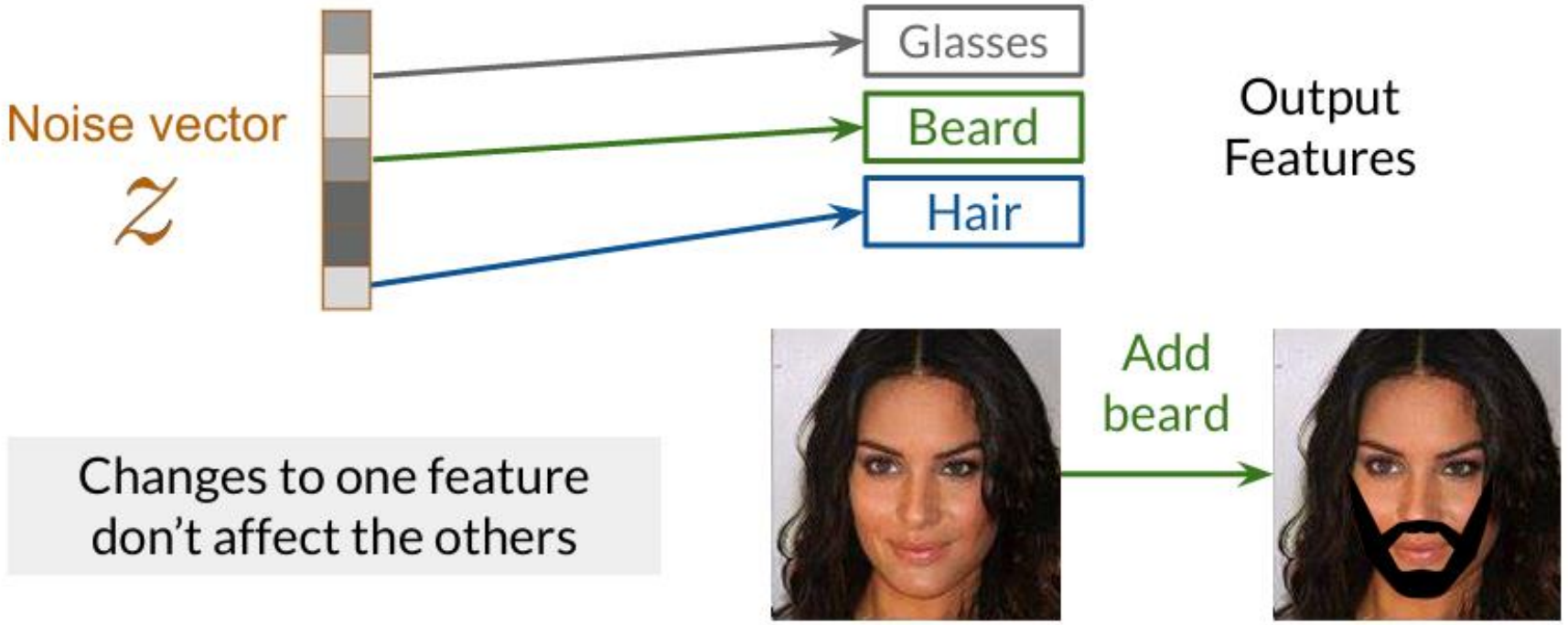




Foundations of GAN

Disentanglement

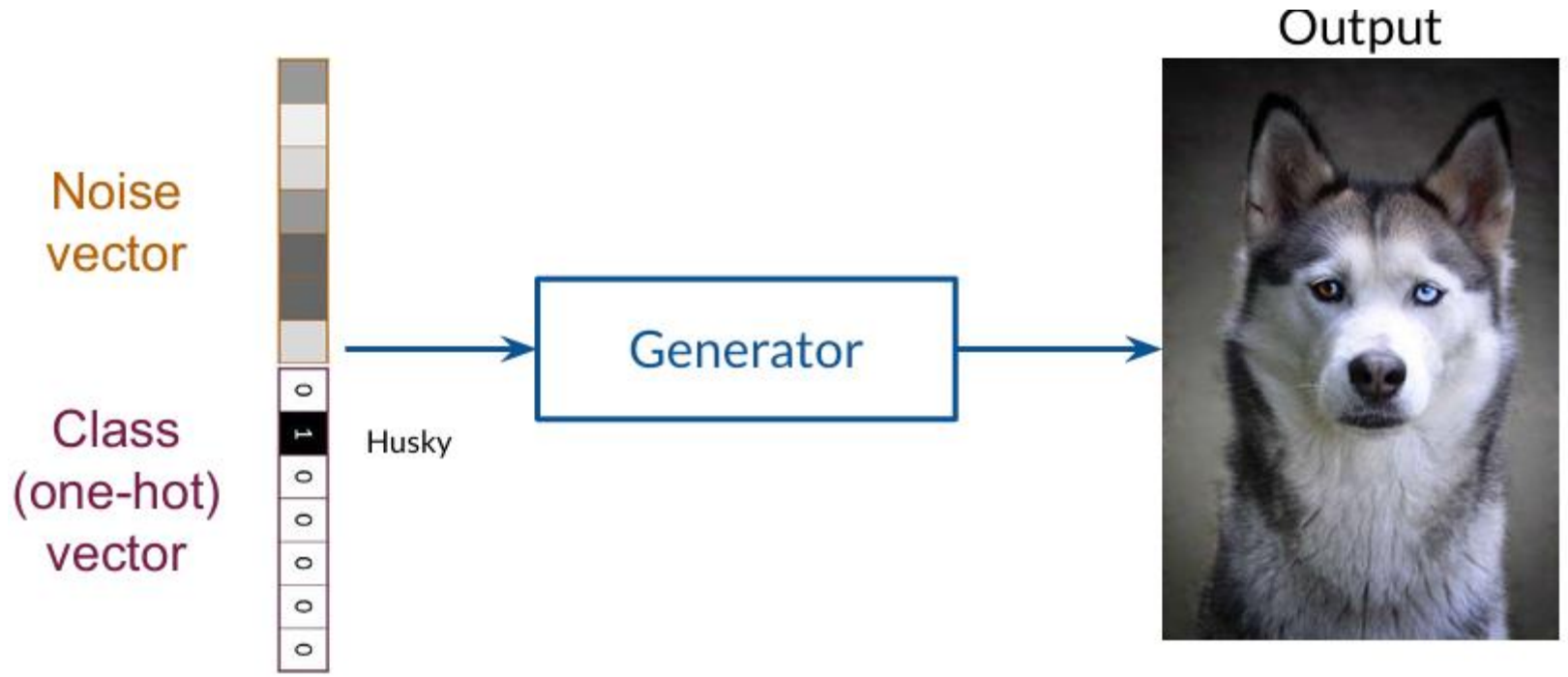
Disentangled Z-Space



Foundations of GAN

Disentanglement

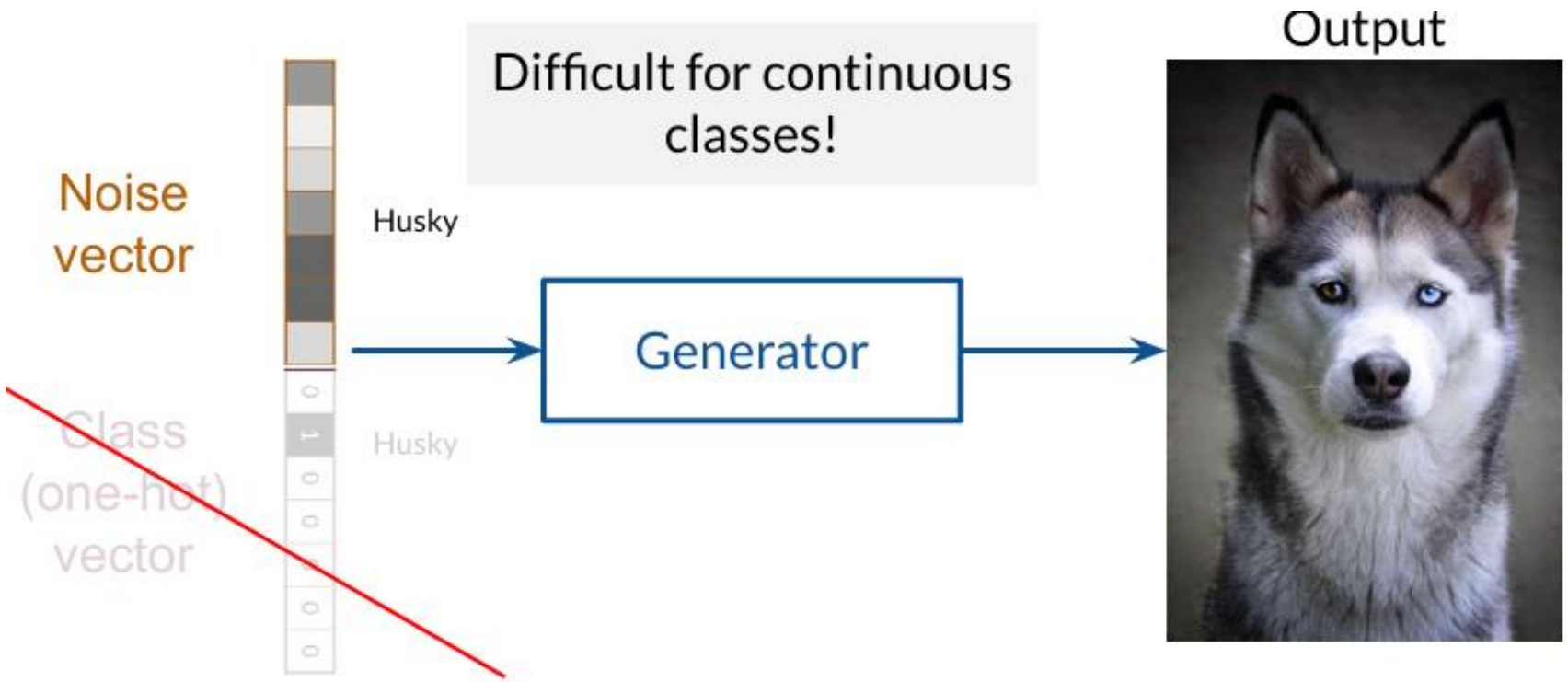
Encourage Disentanglement: Supervision



Foundations of GAN

Disentanglement

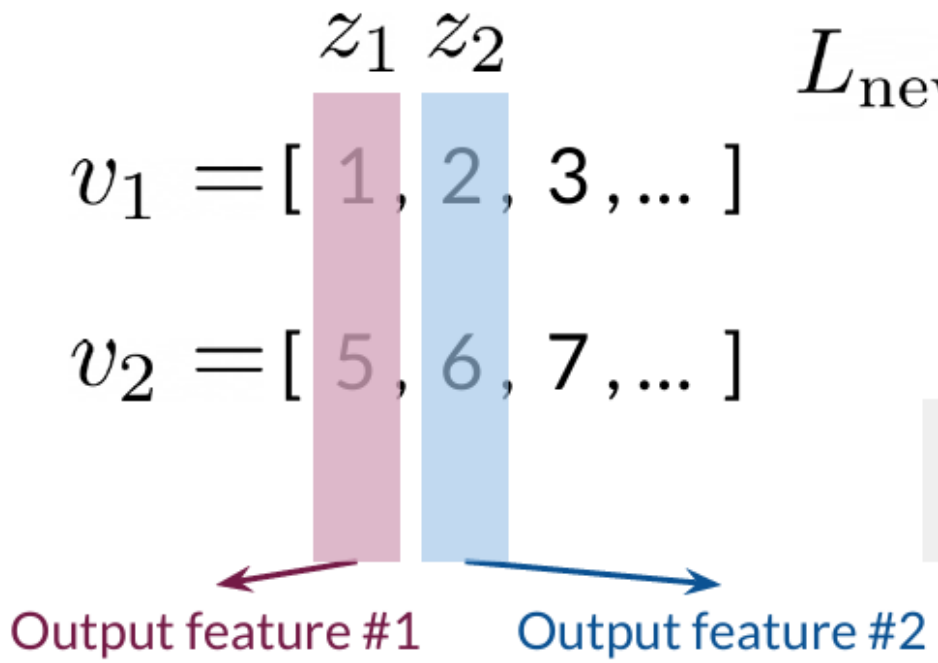
Encourage Disentanglement: Supervision



Foundations of GAN

Disentanglement

Encourage Disentanglement: Loss Function



$$L_{\text{new}} = L_{\text{original}} + \text{reg}_d$$

Original loss Regularization

Can be any loss function
(e.g. BCE, W-Loss)

- Disentangled Z-spaces let you control individual features by corresponding z values directly to them
- There are supervised and unsupervised methods to achieve disentanglement

Foundations of GAN

Demo: Build a Conditional GAN

Dataset: MNIST

Machine Learning: Supervised

Knowledge Check:

Q1. What is the role of the generator?

A. To find the real image in a pile of fakes.

B. To create realistic looking fake images.

C. To sort images by color.

D. To piece together real images to make a fake.

Machine Learning: Supervised

Knowledge Check:

Q2. What is the role of the discriminator?

A. To classify images as real or fake.

B. To create fake images.

C. To rank images based on complexity.

D. To tell the generator it’s doing a stellar job.

Machine Learning: Supervised

Knowledge Check:

Q3. What is the primary goal of the discriminator, in a probabilistic sense?

A. Capture the probability of x and y: P(x ∩ y).

B. Capture the probability of the class y: P(y).

C. Capture the conditional probability P(y | x).

D. Capture the probability of the features x: P(x).

Machine Learning: Supervised

Knowledge Check:

Q4. What is the primary goal of the generator, in a probabilistic sense?

A. Model the features x conditioned on class y: P(x | y).

B. Capture the probability of x and y: P(x ∩ y).

C. Capture the probability of the class y: P(y).

D. Model the union of x and y: P(x U y).

Machine Learning: Supervised

Knowledge Check:

Q5. How does the discriminator learn over time?

A. Getting feedback on if its classification was correct.

B. Receiving user input.

C. Using feedback from the generator.

D. Comparing the images to the ones on the internet.

Machine Learning: Supervised

Knowledge Check:

Q6. How does the generator learn over time?

A. Comparing the fake to a real image.

B. Generating more diverse images.

C. Receiving user feedback.

D. Using feedback from the discriminator.

Machine Learning: Supervised

Knowledge Check:

Q7. What should the skill levels of the discriminator and generator be?

A. Both should be at random levels.

B. The discriminator should be really good.

C. Both should be at similar skill levels.

D. The generator should be really good.

Machine Learning: Supervised

Knowledge Check:

Q8. What issue with ReLUs does Leaky ReLU resolve?

A. The ReLU function is linear.

B. The ReLU function returns random values.

C. When z >= 0, the derivative equals zero, causing some neurons to get stuck and stop learning.

D. When z <= 0, the derivative equals zero, causing some neurons to get stuck and stop learning.

Machine Learning: Supervised

Knowledge Check:

Q9. What points on a function are considered for the evaluation of 1-Lipschitz continuity?

A. Both axes: x-axis and y-axis.

B. Primarily the x-axis.

C. All points on the function.

D. The origin: (0,0).

Machine Learning: Supervised

Knowledge Check:

Q10. How can you use a classifier for controllable generation?

A. You can replace the discriminator with a classifier.

B. You can train a classifier on features to help train your generator.

C. You can calculate the loss between the classifier's results and the discriminator's and pass it to the generator.

D. You can calculate the gradient of the z vectors along certain features through the classifier to find the direction to move the z vectors.

?

Thank You !!!