Copyright Notice

deeplearning.ai

Generative Models

Outline

- What are generative models
- Types of generative models

mirror each other

Generative Models vs. Discriminative Models

Discriminative models

like classification model

Features Class

$$X \rightarrow Y$$

$$P(Y|X)$$

model prob of class Y

Generative models

learn realistic representation of a class

Noise Class Features

(random) $$\xi, Y \rightarrow X$$

$$P(X|Y)$$

Not same dog every time

Generative Models vs. Discriminative Models

Generative models

Noise Class Features

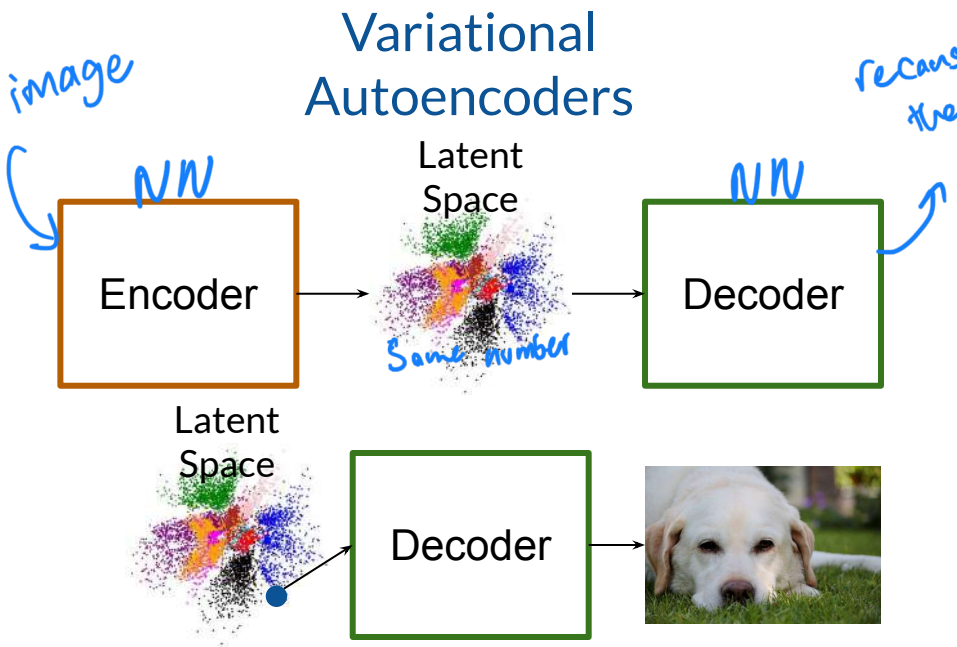
$$\xi, Y \rightarrow X$$

$$P(X|Y)$$

NN (neural network)

Generative Models

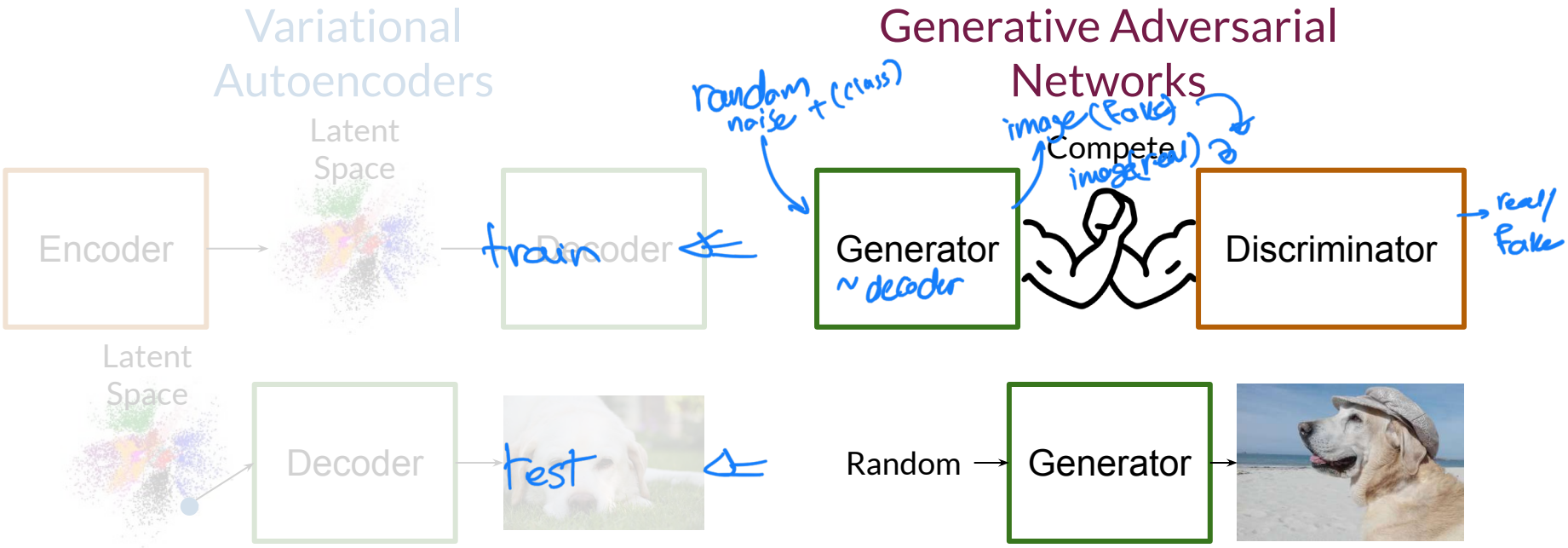
Variational Autoencoders



Generative Adversarial Networks

Available from: https://arxiv.org/abs/1804.00891

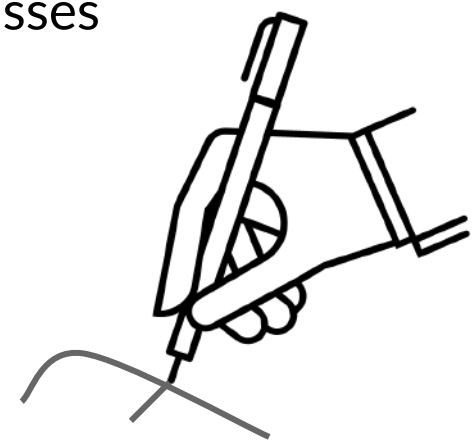
Generative Models



Available from: https://arxiv.org/abs/1804.00891

Summary

- Generative models learn to produce examples
- Discriminative models distinguish between classes
- Up next, GANs!

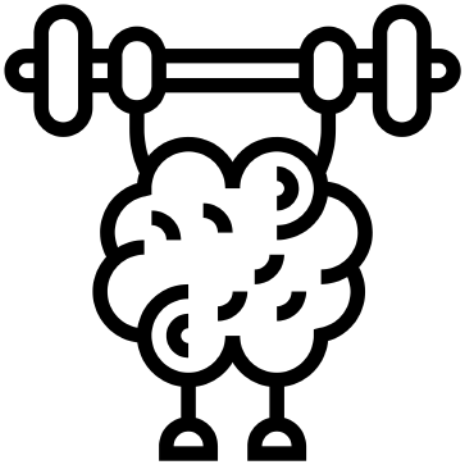


deeplearning.ai

Real Life GANs

Outline

- Cool applications of GANs
- Major companies using them



GANs Over Time

2014 Black and white

↓

2018 Colored photo-realistic

GANs Over Time

Face Generation
StyleGAN2

These people do not exist!

Karras, Tero, et al. "Analyzing and improving the image quality of stylegan." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020.

GANs Over Time

generated img has feat

StyleGAN2

Mimics the distribution of the training data

Karras, Tero, et al. "Analyzing and improving the image quality of stylegan." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020.

https://9gag.com/gag/aWYZKWx

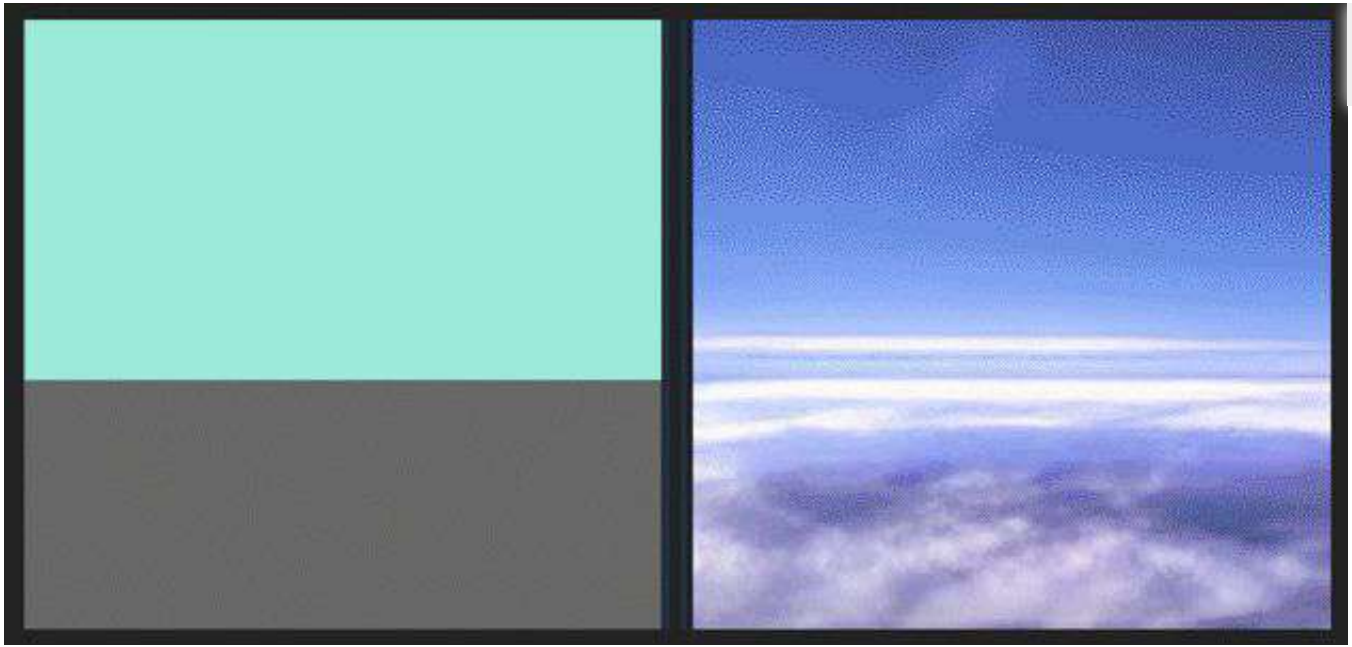
GANs for Image Translation

From one domain to another

CycleGAN

Park, Taesung, et al. "Semantic image synthesis with spatially-adaptive normalization." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019.

GANs for Image Translation



GauGAN

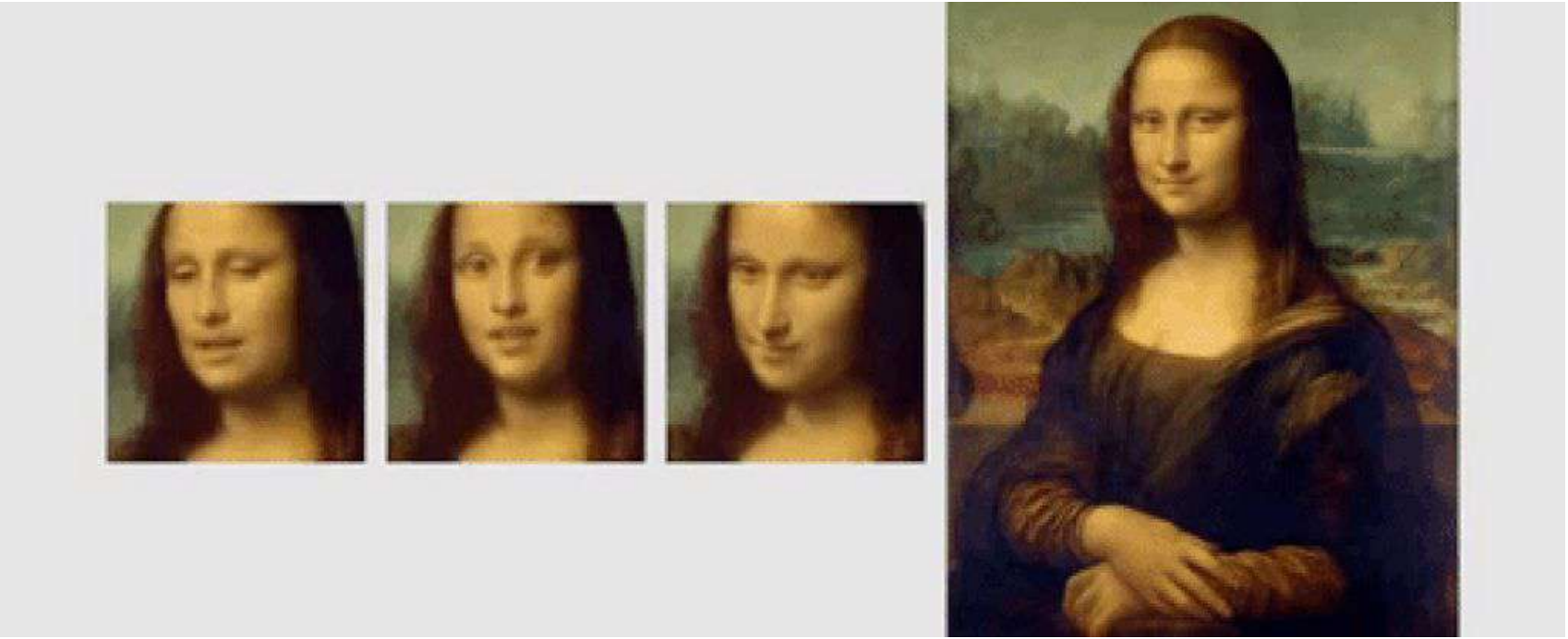
Doodles

↓

Pictures

Park, Taesung, et al. "Semantic image synthesis with spatially-adaptive normalization." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019.

GANs are Magic!



Zakharov, Egor, et al. "Few-shot adversarial learning of realistic neural talking head models." *Proceedings of the IEEE International Conference on Computer Vision*. 2019.

GANs for 3D Objects

3D-GAN

Generative Design

Wu, Jiajun, et al. "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling." *Advances in neural information processing systems*. 2016.

Companies Using GANs

Next-gen Photoshop

Text Generation

Data Augmentation

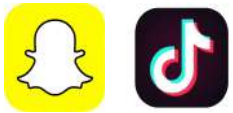
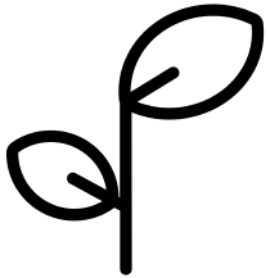


Image Filters

Super-resolution

Summary

- GANs' performance is rapidly improving
- Huge opportunity to work in this space!
- Major companies are using them



deeplearning.ai

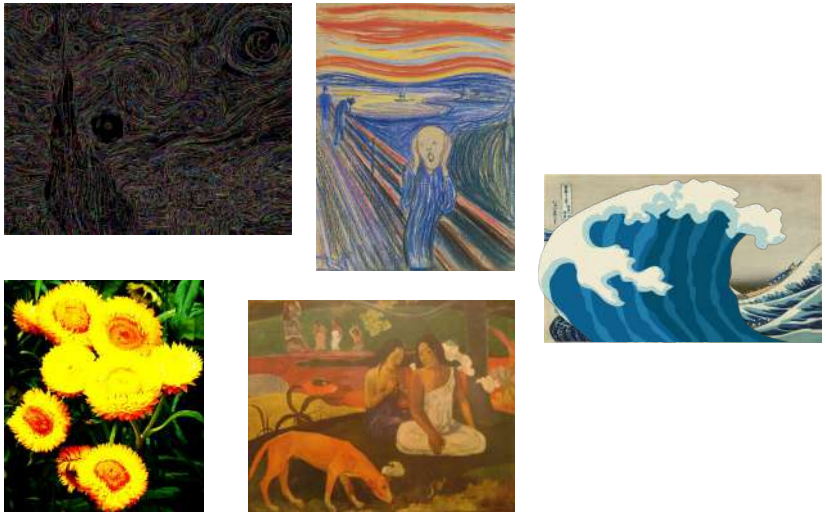
Intuition Behind GANs

Outline

- The goal of the generator and the discriminator
- The competition between them

Generative Adversarial Network

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



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

Generative Adversarial Network

Fake

Real

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

Generative Adversarial Network

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

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

Generative Adversarial Network

(don't look at real)

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

at beginning (it doesn't know what to generate)

Doesn't know how it should look

Generative Adversarial Network

(Look at real)

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

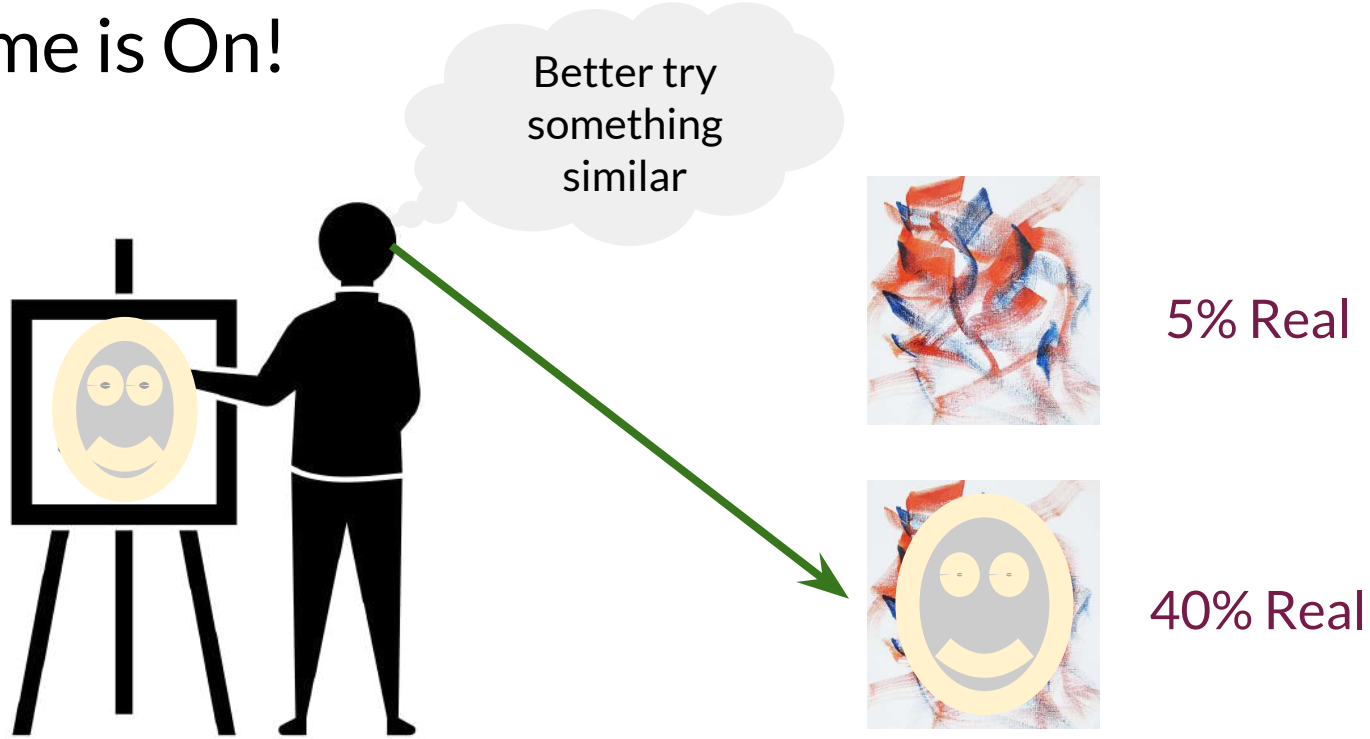
The Game Is On!

5% Real

40% Real

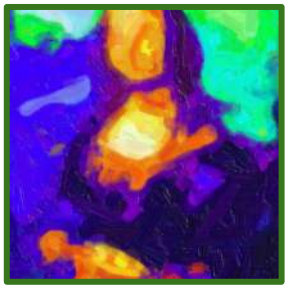
80% Real

The Game is On!



The Game Is On!

30% Real

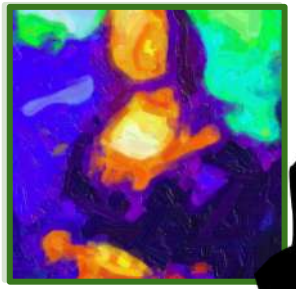


60% Real

95% Real

The Game Is On!

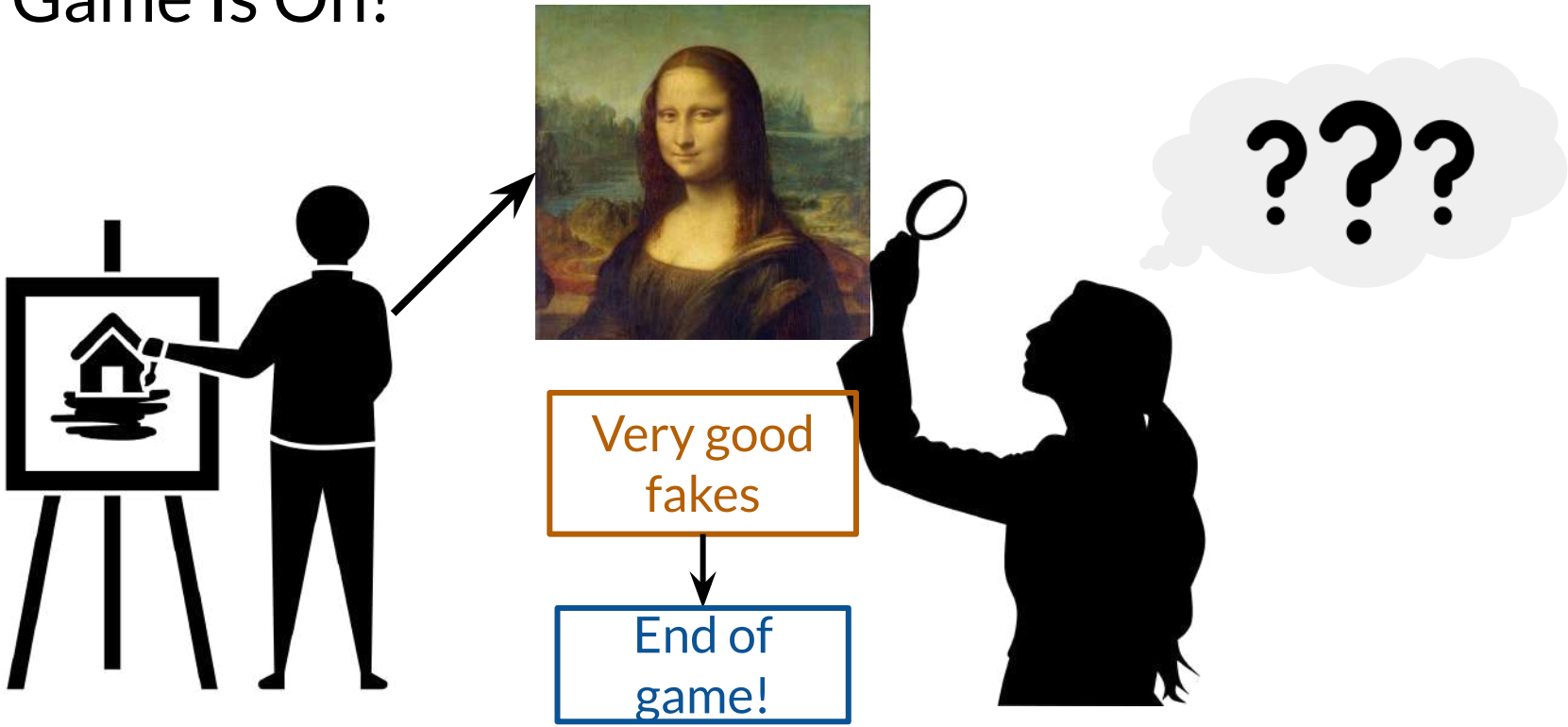
30% Real



60% Real

Wrong!

The Game Is On!



Summary

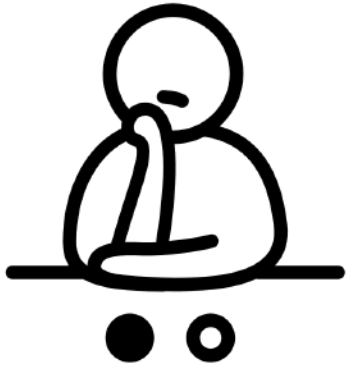
- The generator's goal is to fool the discriminator
- The discriminator's goal is to distinguish between real and fake
- They learn from the competition with each other
- At the end, *fakes* look real

deeplearning.ai

Discriminator

Outline

- Review of classifiers
- The role of classifiers in terms of probability
- Discriminator



Classifiers

Distinguish between different classes

Turtle

Bird

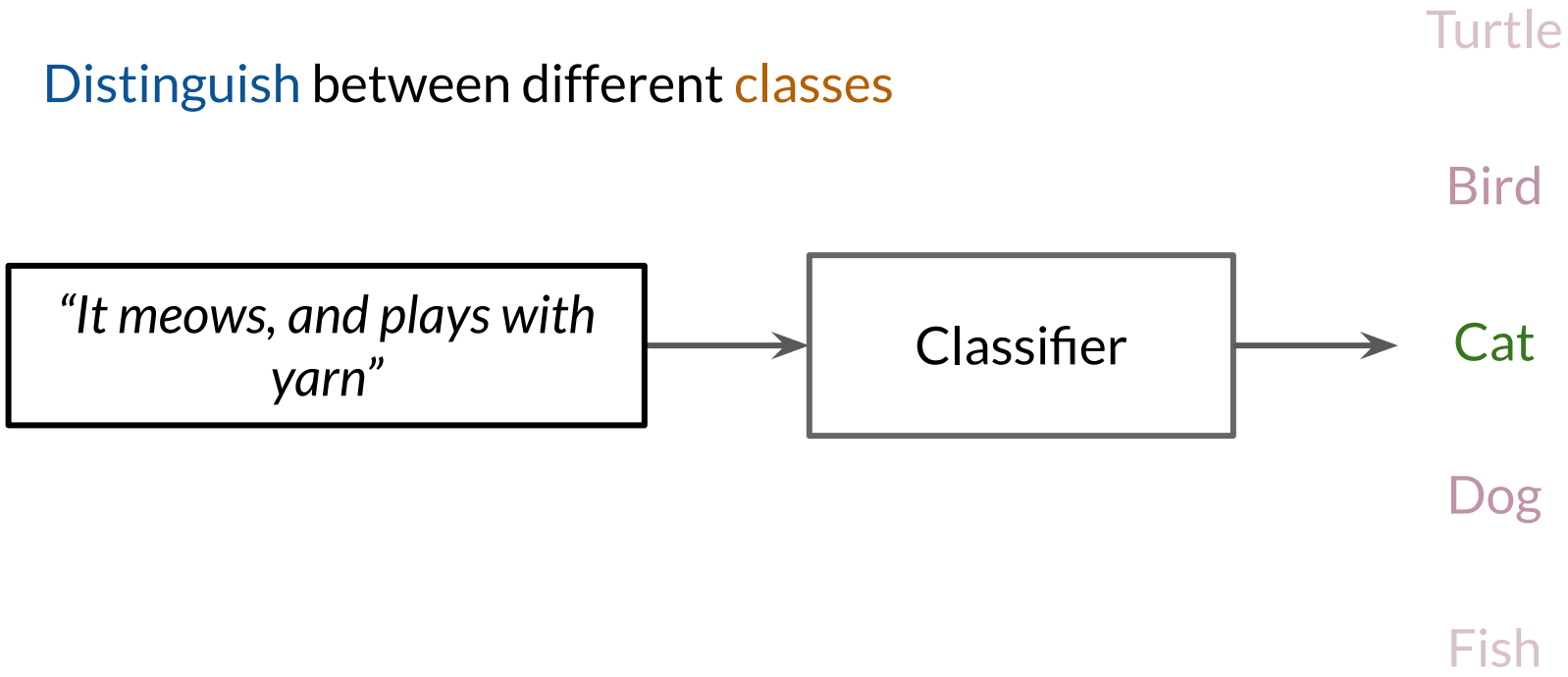
Cat

Dog

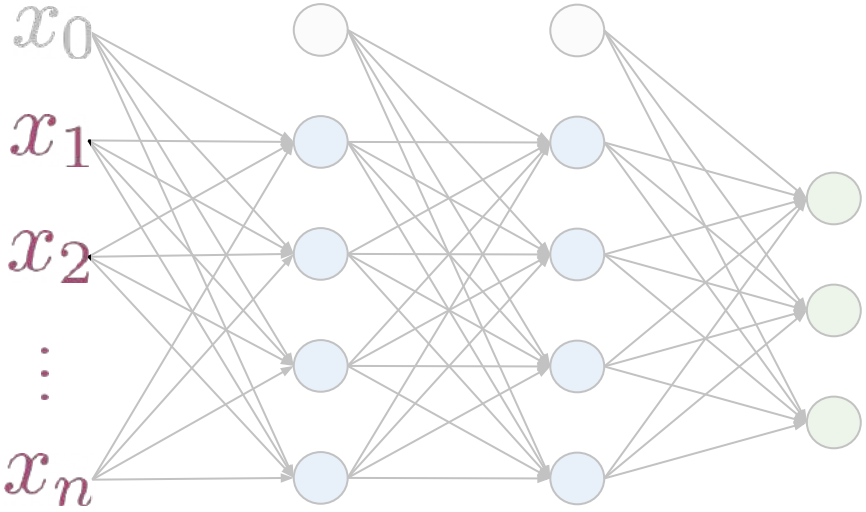
Fish

Classifiers

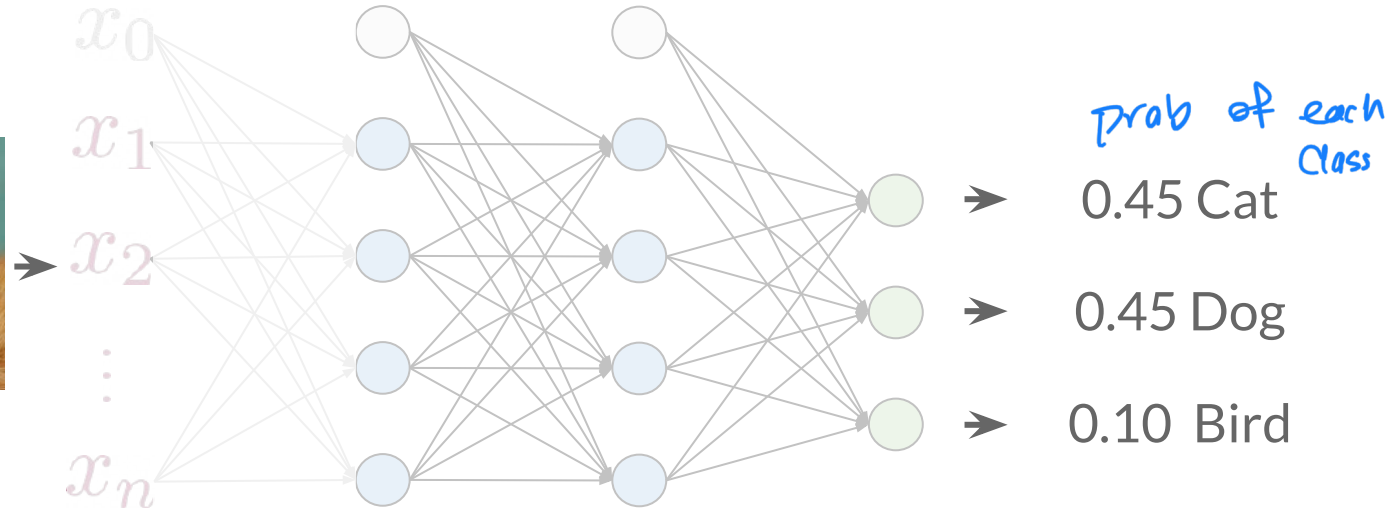
Distinguish between different classes



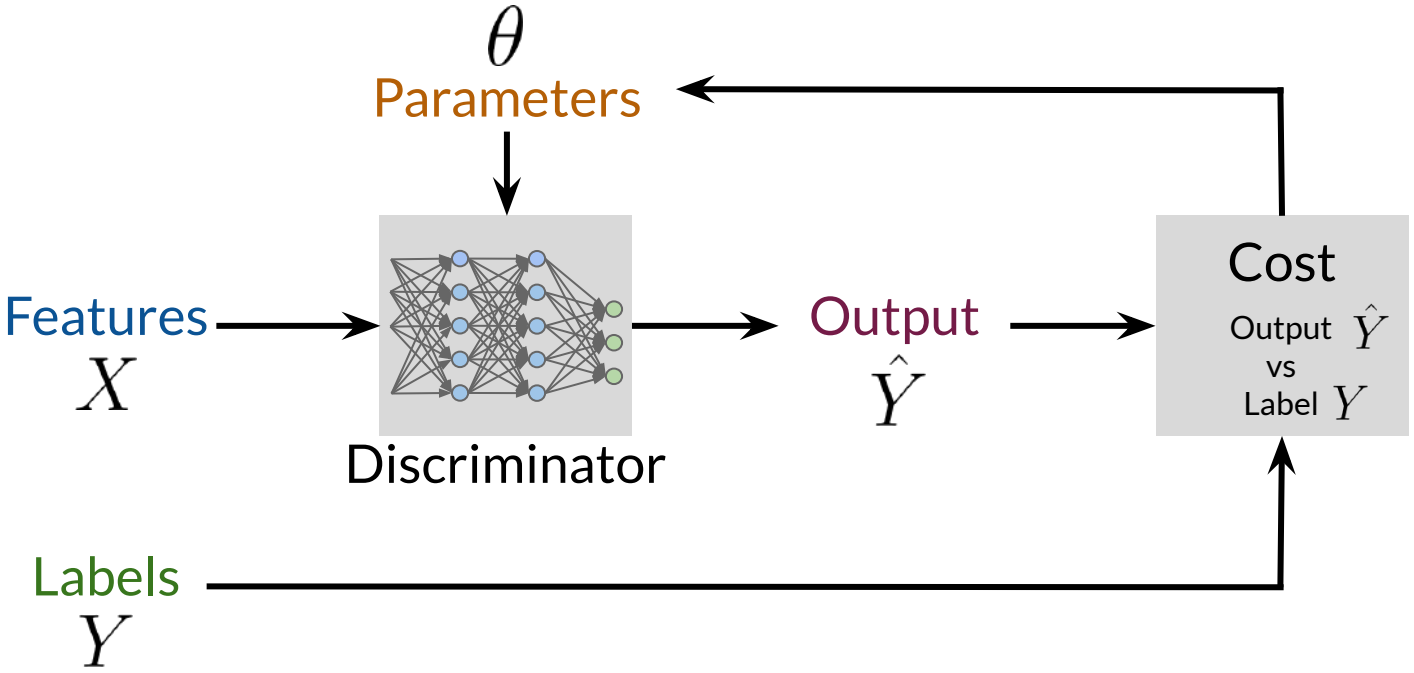
Neural Networks



Neural Networks



Classifiers (training)



Classifiers

Turtle

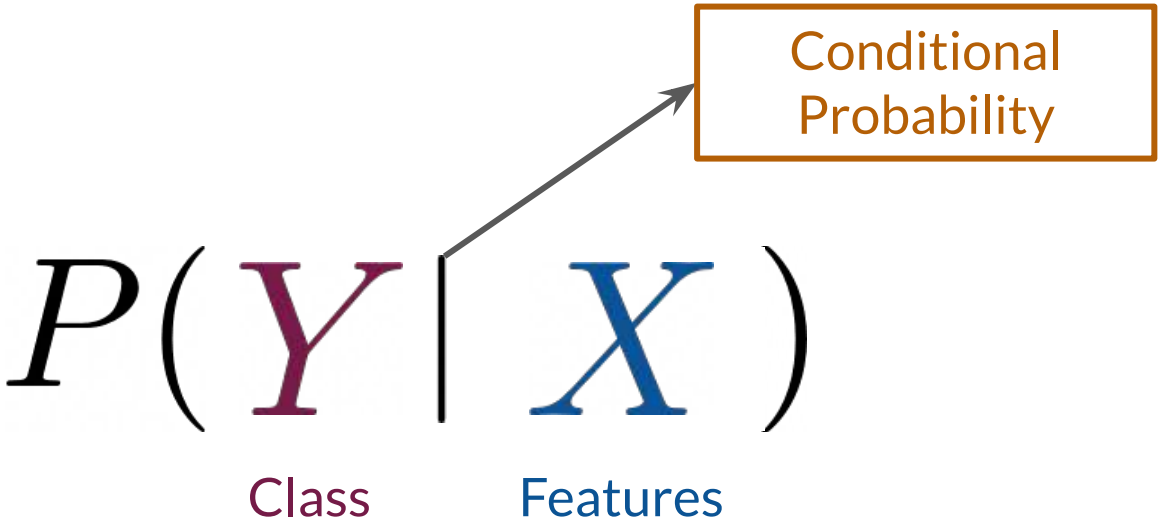
Bird

$$P(\text{Cat} \mid \text{  })$$

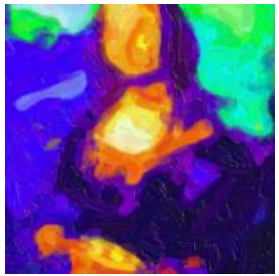
Dog

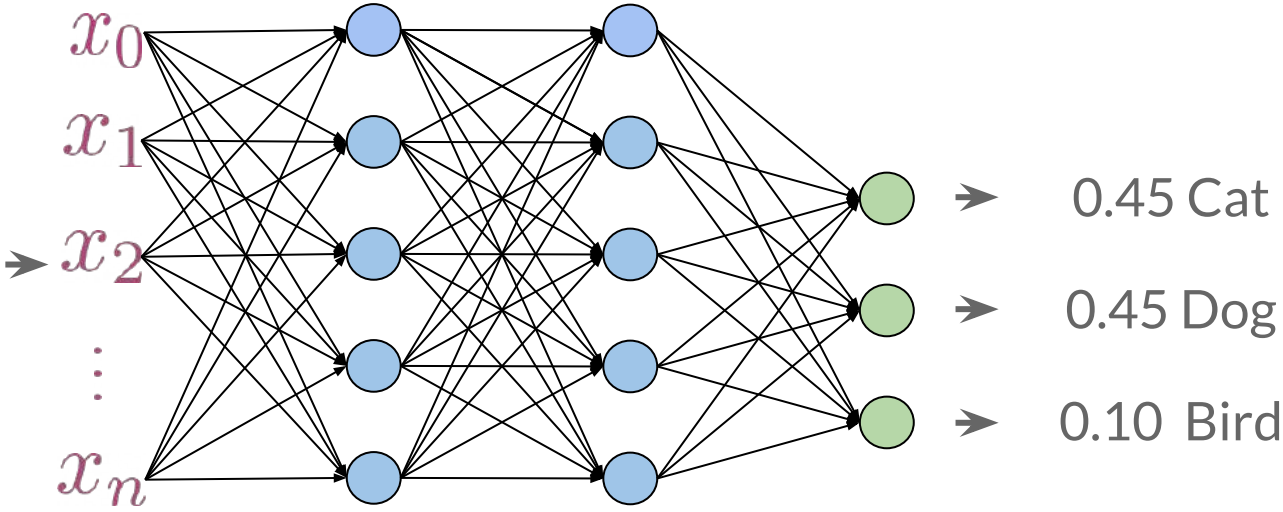
Fish

Classifiers

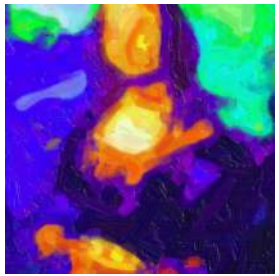


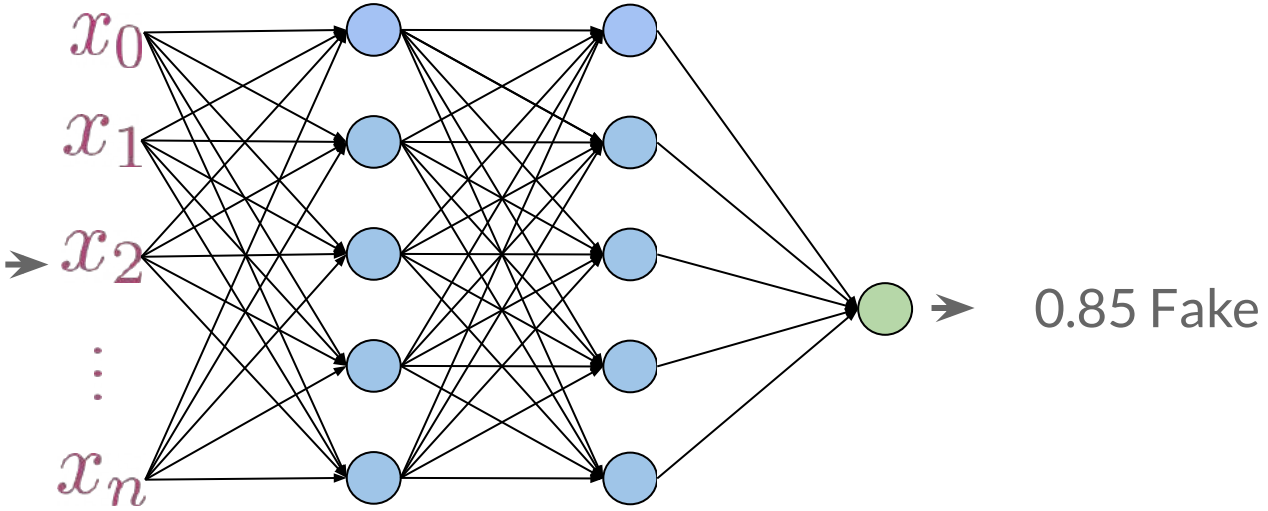
Discriminator





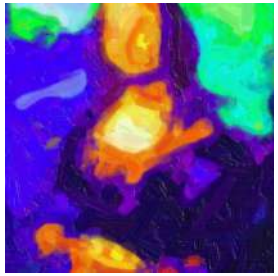
Discriminator





Discriminator

Discriminator

$$P(\text{Fake} \mid \text{  }) = 0.85 \rightarrow \boxed{\text{Fake}}$$

Class

Features

Summary

- 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

deeplearning.ai

Generator

Outline

- What the generator does
- How it improves its performance
- Generator in terms of probability

the IMP one !

Generator

Turtle

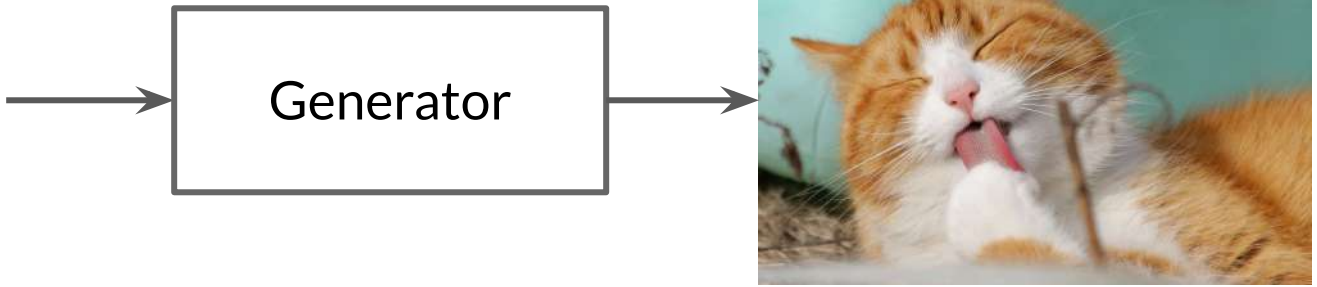
Bird

Cat

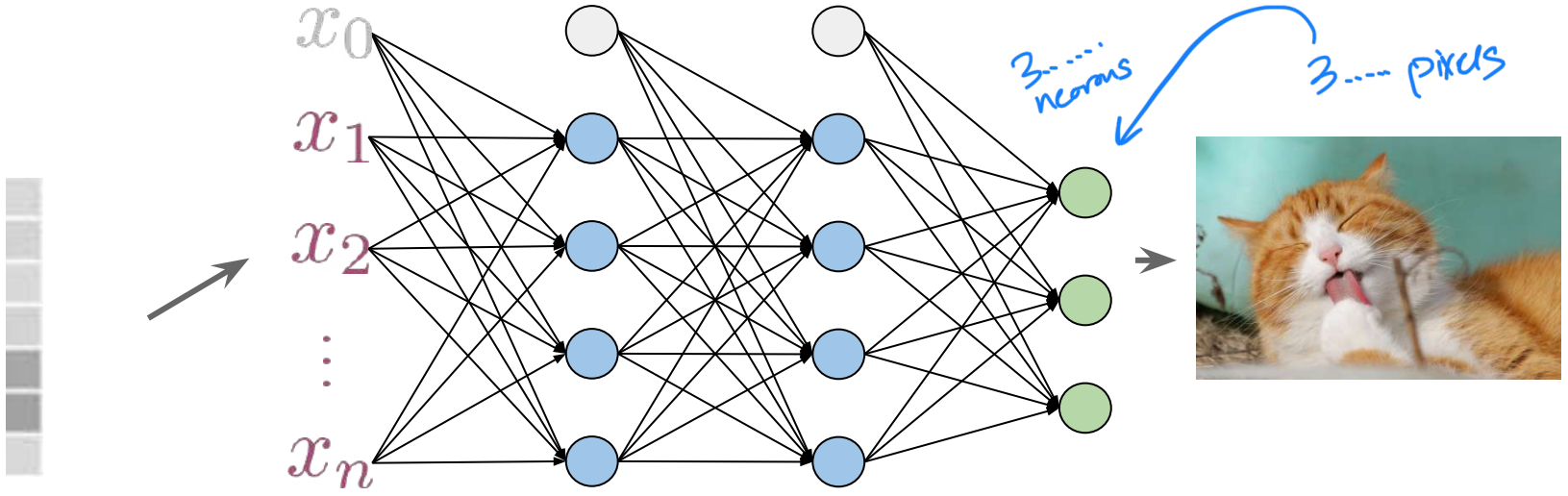
Dog

Fish

Generates examples of the class



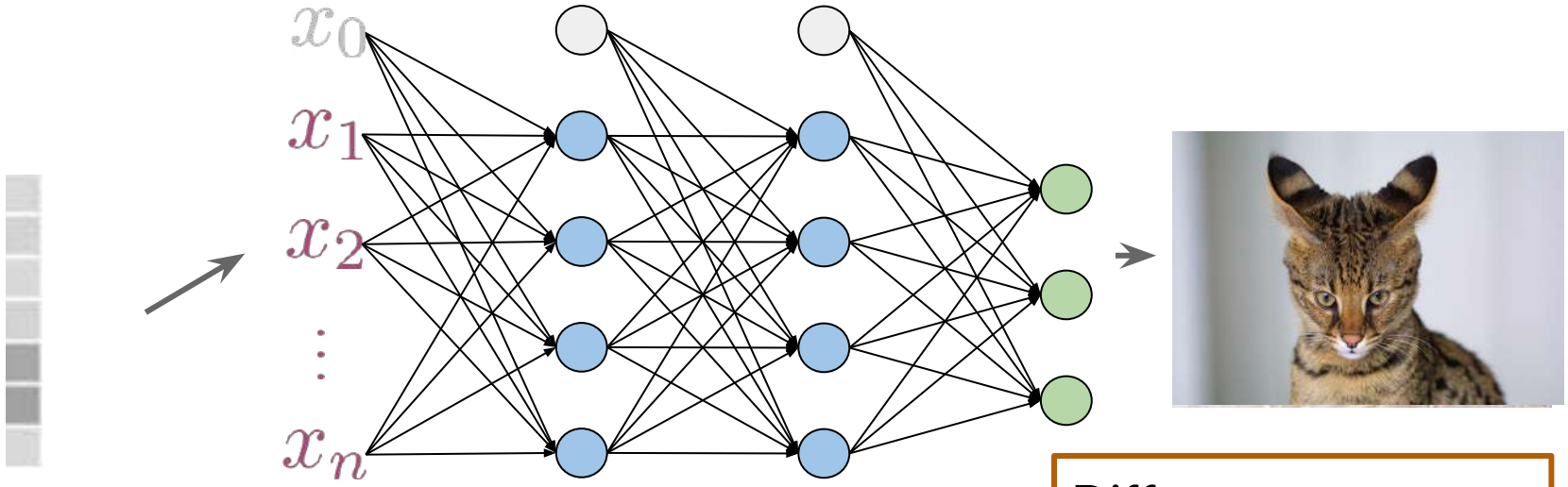
Neural Networks



Noise
(random features)

↳ For diff output every time

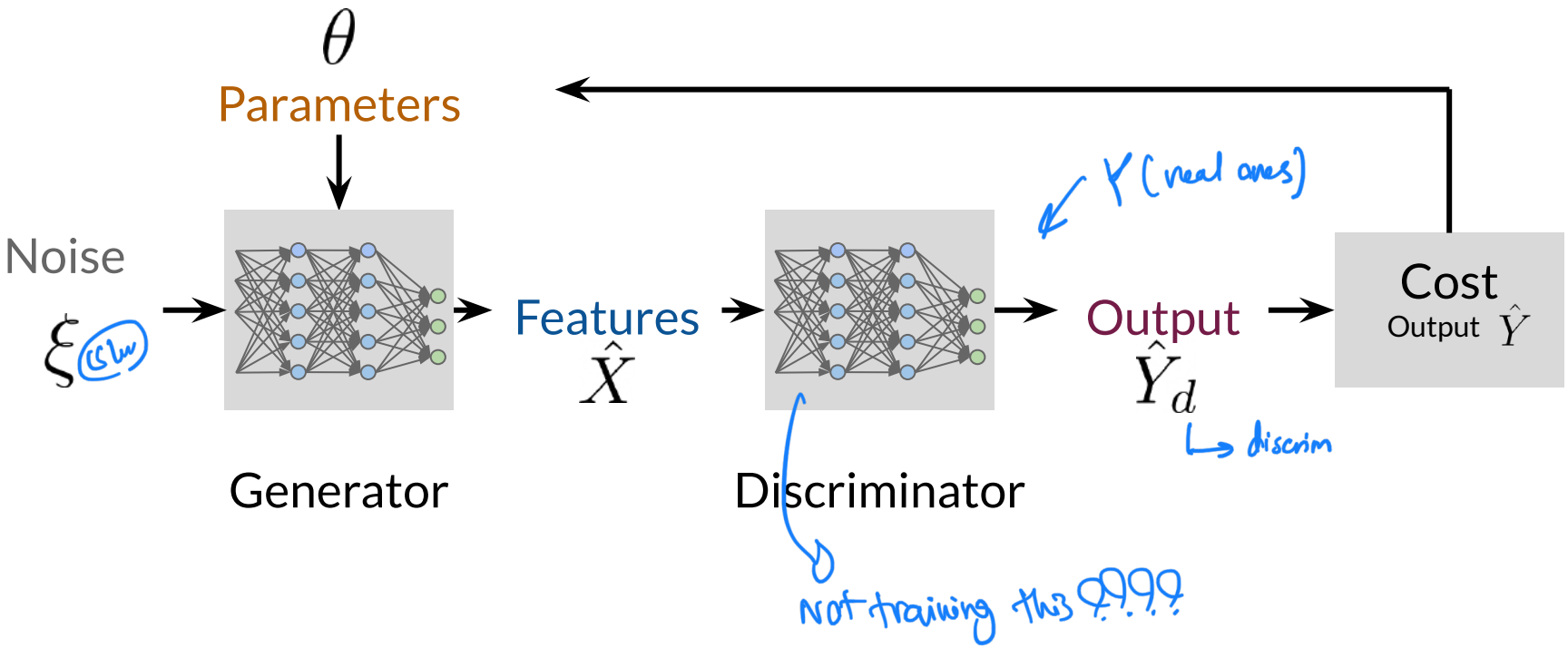
Neural Networks



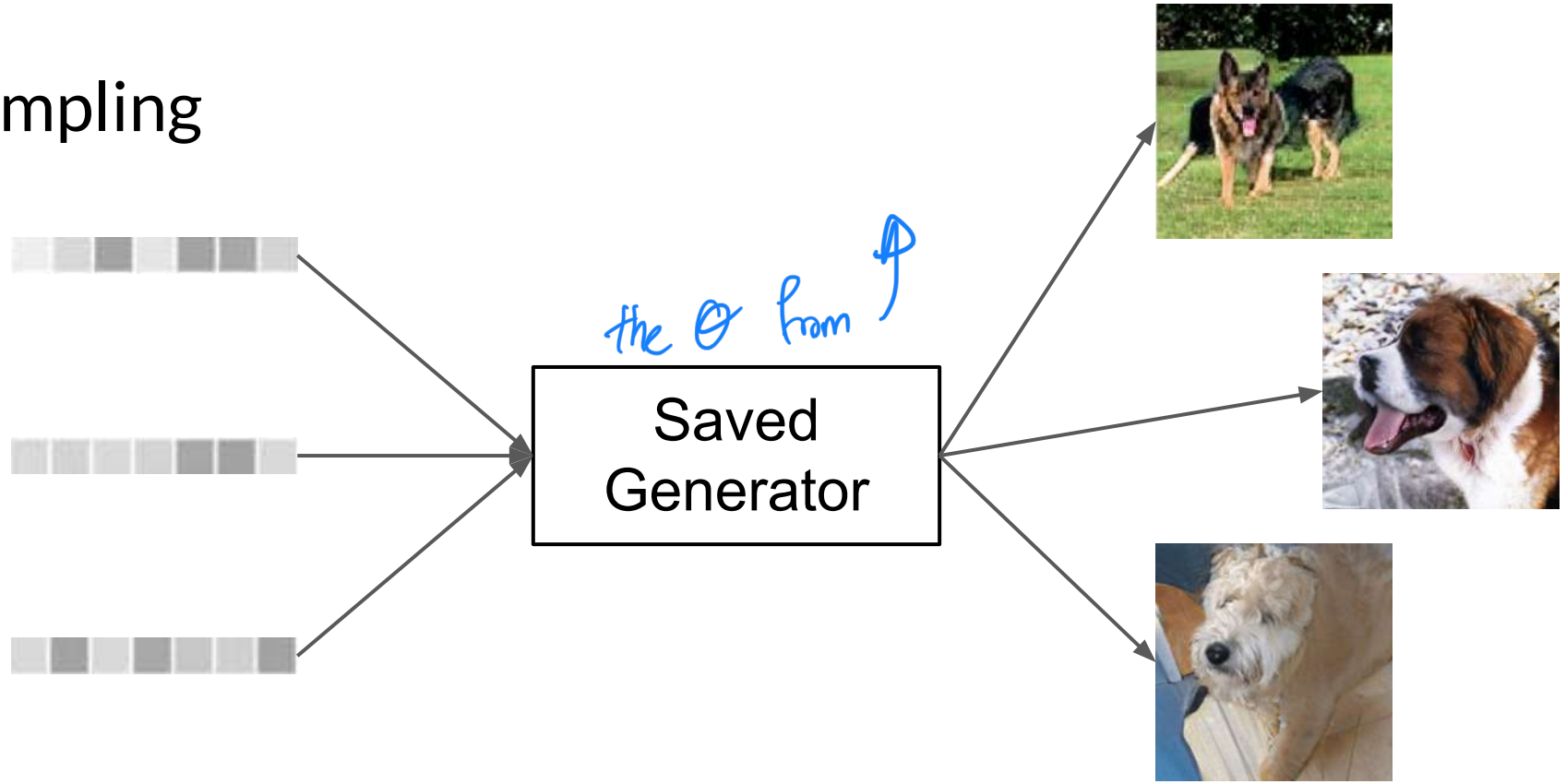
Noise
(random features)

Different outputs at every run!

Generator: Learning



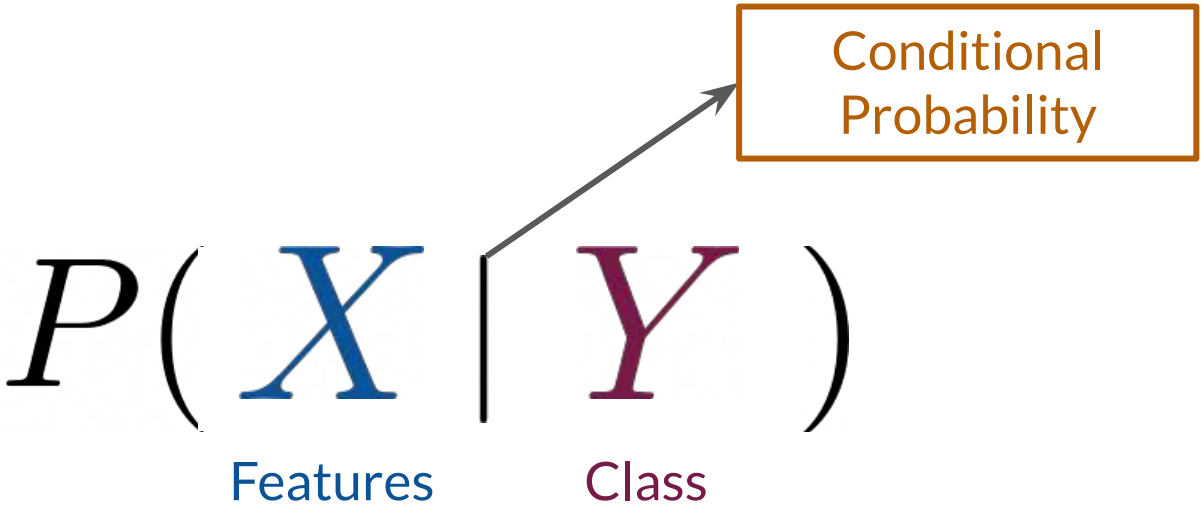
Sampling



Generator

$$P\left(\text{  }\middle|\ \begin{array}{l}\text{Turtle}\\ \text{Bird}\\ \text{Cat}\\ \text{Dog}\\ \text{Fish}\end{array}\right)$$

Generator

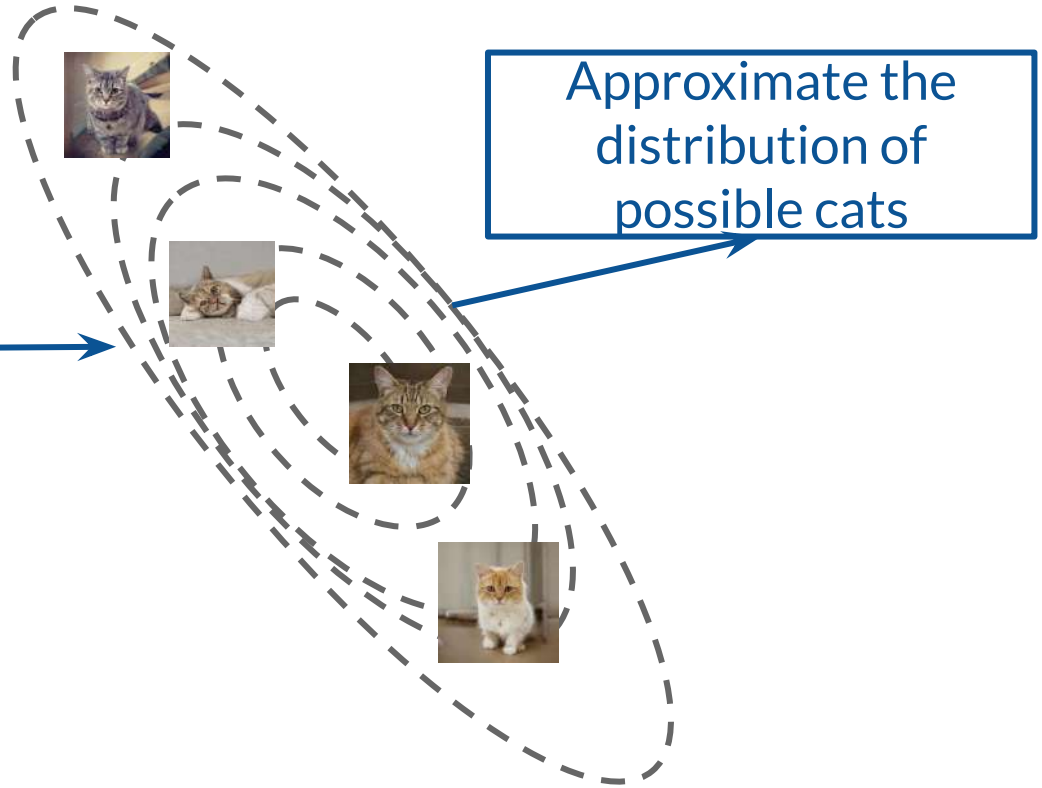


if only one class
→ P(X)
just X

Generator

$$P(X)$$

Features



Images available from: http://thesecatsdonotexist.com/

Summary

- The generator produces fake data
- It learns the probability of features X
- The generator takes as input noise (random features)

deeplearning.ai

BCE Cost Function

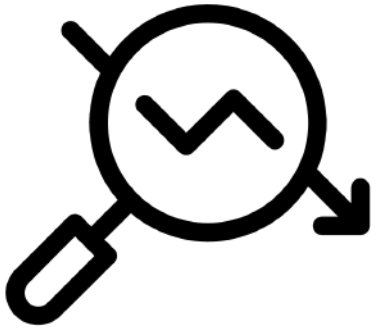
Outline

for train GAN (Binary)

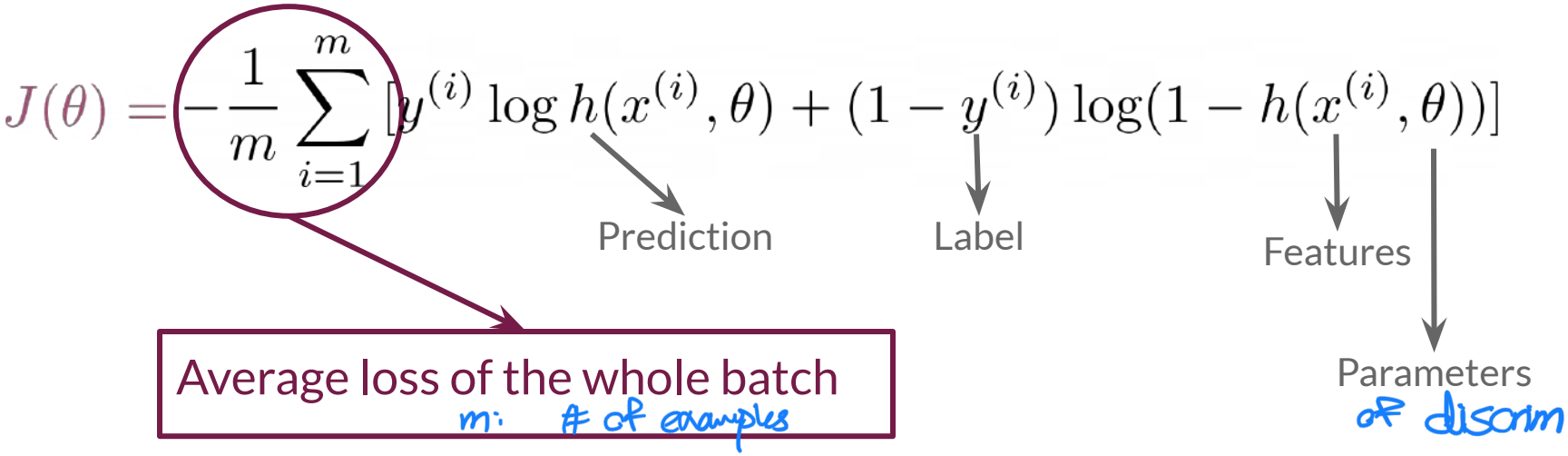
- Binary Cross Entropy (BCE) Loss equation by parts

fake/real

- How it looks graphically



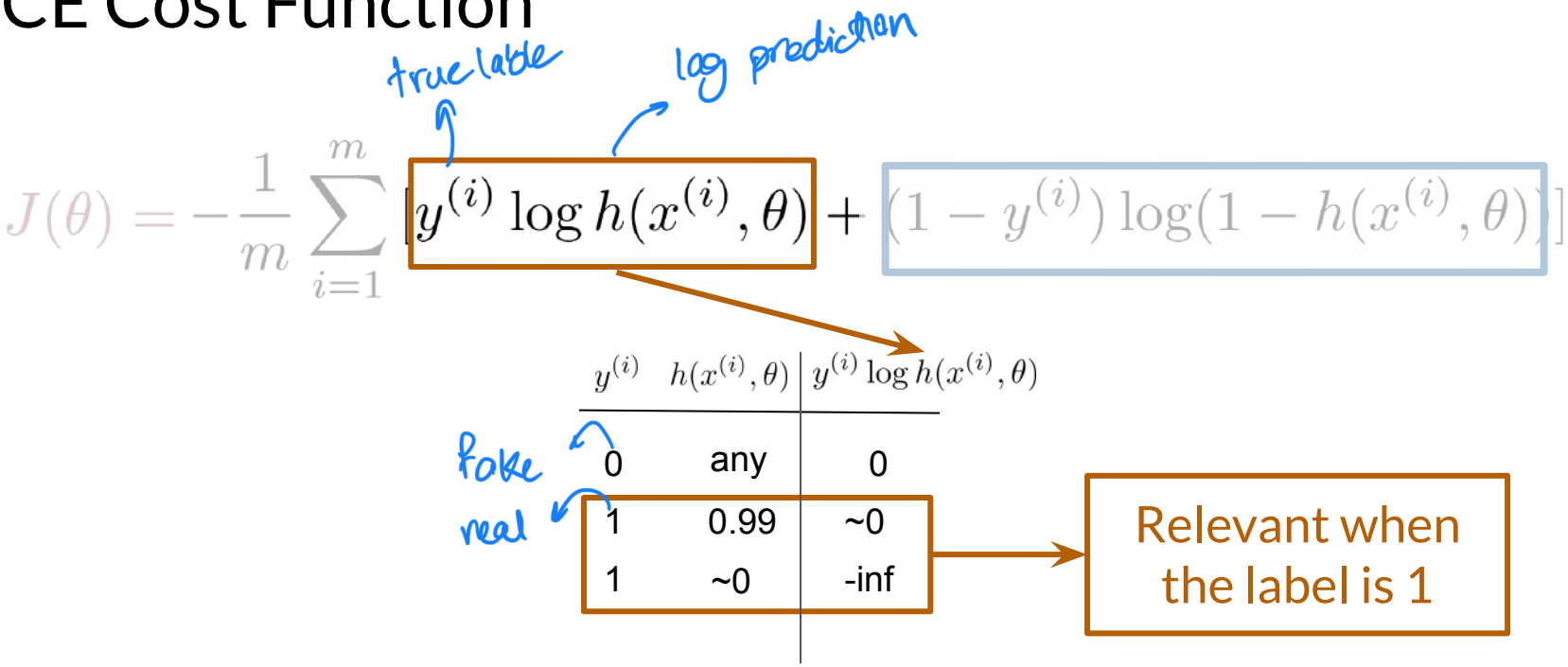
BCE Cost Function



BCE Cost Function

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

BCE Cost Function



BCE Cost Function

so J is + (because of $-\infty$ s)

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

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

Relevant when the label is 0

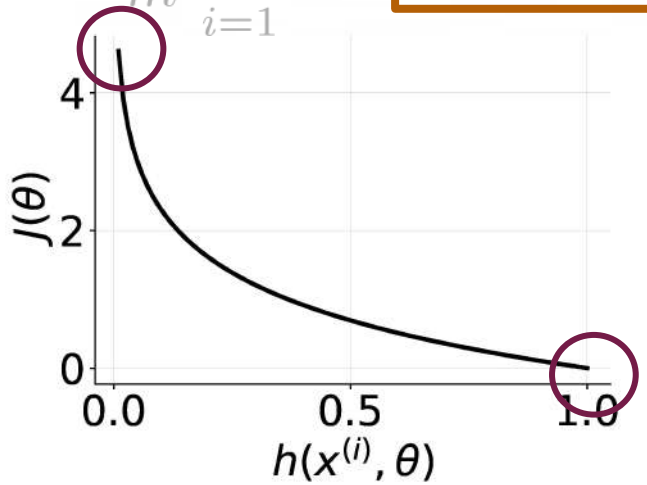
BCE Cost Function

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

Ensures that the cost is always greater or equal to 0

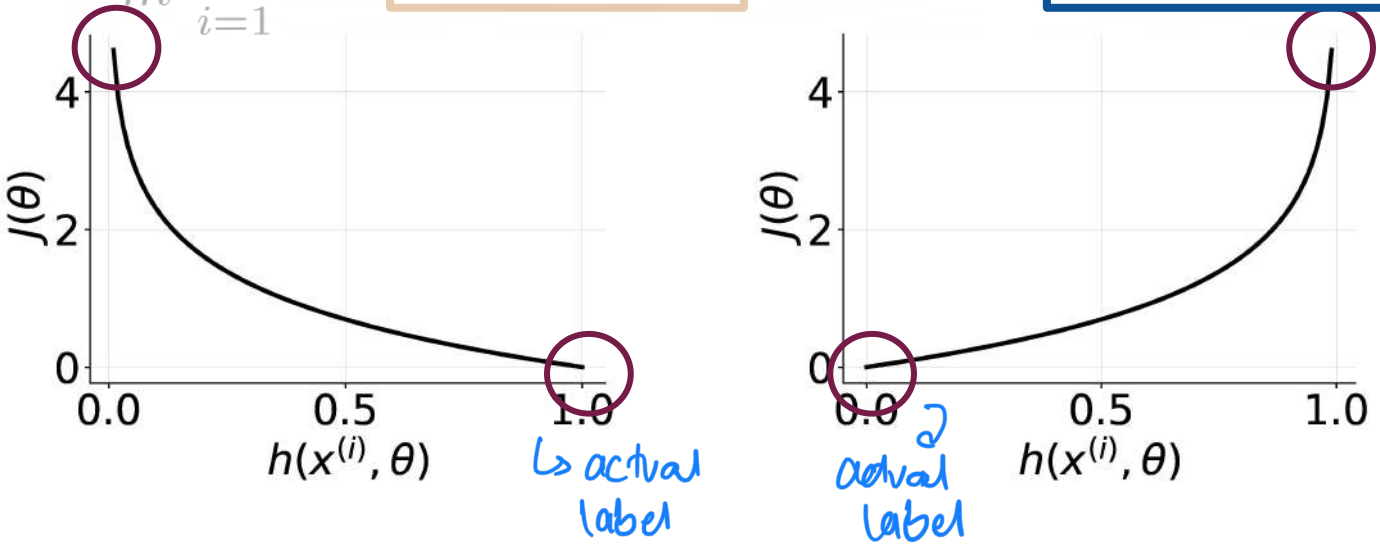
BCE Cost Function

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



BCE Cost Function

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



Summary

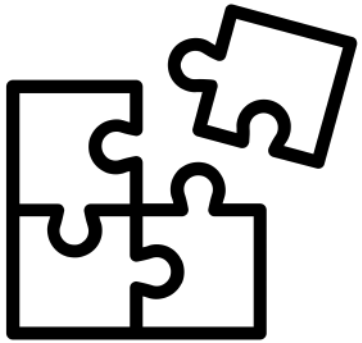
- The BCE cost function has two parts (one relevant for each class)
- Close to zero when the label and the prediction are similar
- Approaches infinity when the label and the prediction are different

deeplearning.ai

Putting It All Together

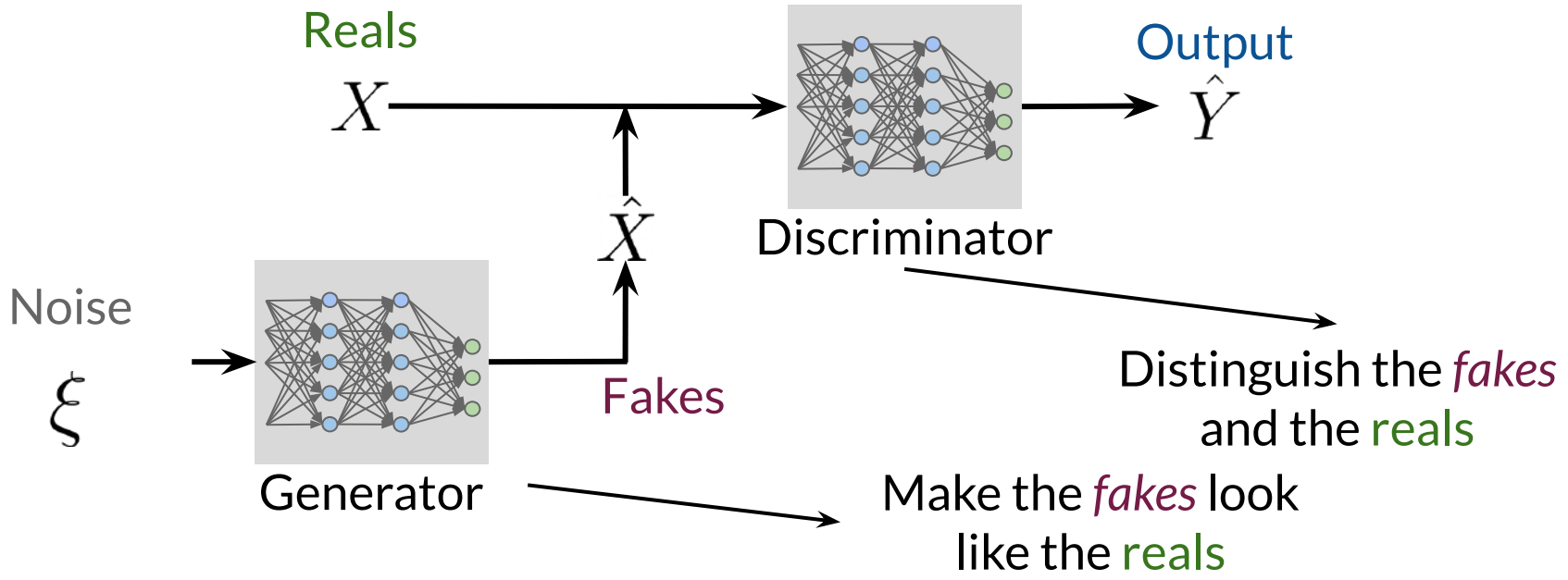
Outline

- How the whole architecture looks
- How to train GANs



We alter the training of gener & discrim

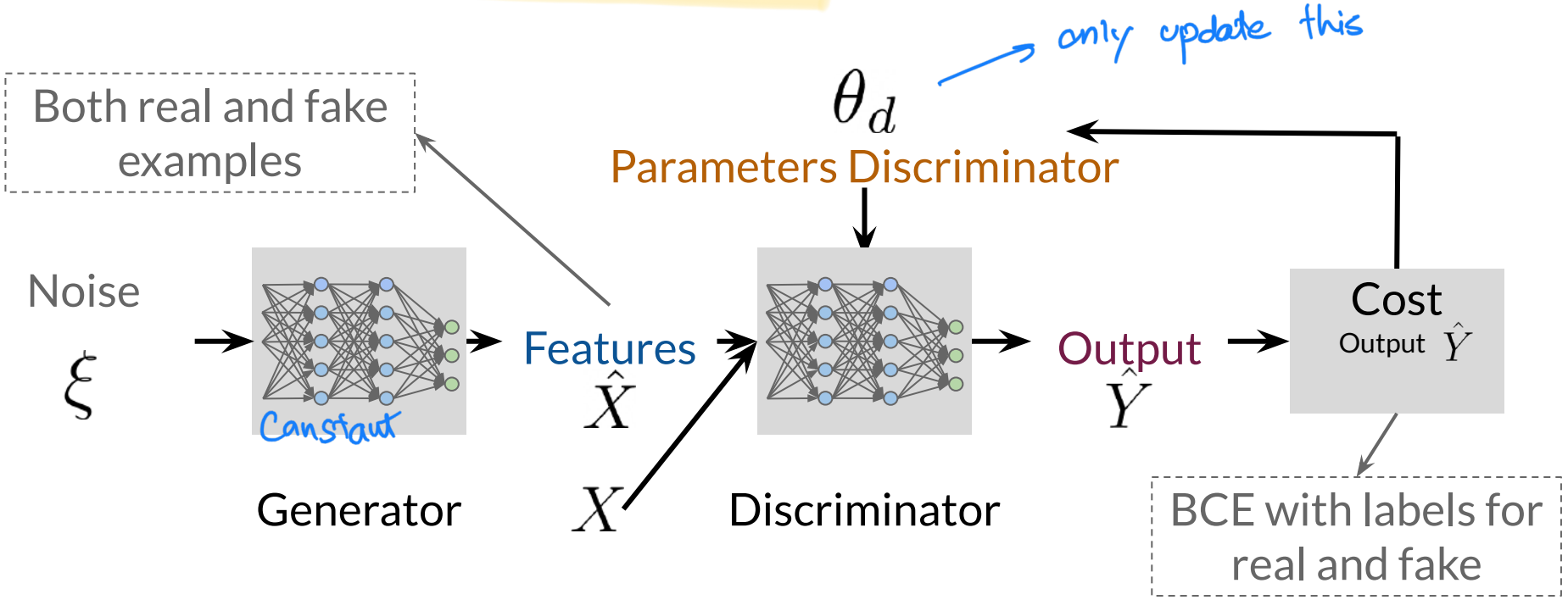
GANs Model



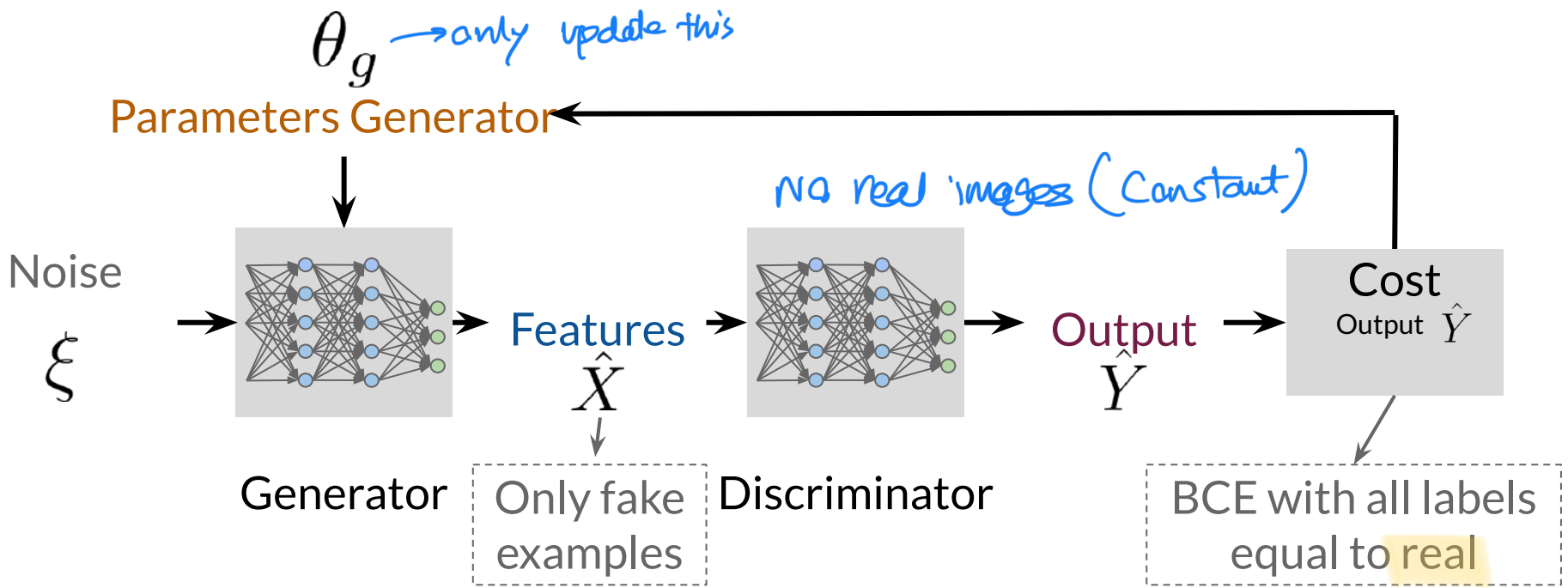
Training GANs: Discriminator

phase

only update this



Training GANs: Generator



Training GANs

easier → may learn too fast

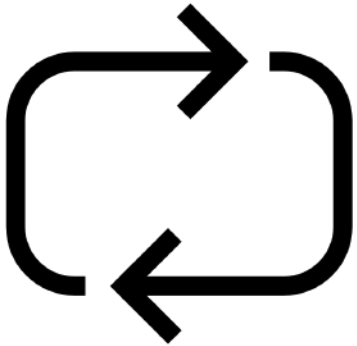
Discriminator Output

Superior Discriminator → *Fakes* as 100% fake → No way to improve

discrim & gener should be improved together & be in sync. be in same scale.

Summary

- GANs train in an alternating fashion
- The two models should always be at a similar “skill” level



deeplearning.ai

Intro to PyTorch (Optional)

Outline

- Comparison with TensorFlow
- Defining Models
- Training

PyTorch vs TensorFlow

PyTorch

Imperative, computations on the go

```
A, B = 1, 2
C = A + B
print(C)
```

```
3
```

Dynamic Computational Graphs

TensorFlow

Symbolic, first define and then compile

```
C = A + B
f = compile(C)
print(f(A = 1, B = 2))
```

```
3
```

Static Computational Graphs

Faster

Tensorflow > 2.0 moves toward PyTorch by including Eager Execution

PyTorch vs TensorFlow

PyTorch | TensorFlow

Currently very similar frameworks!

Tensorflow > 2.0 moves toward PyTorch by including Eager Execution

Defining Models in PyTorch

```
import torch
from torch import nn
```

Custom layers for DL

```
class LogisticRegression(nn.Module):
    def __init__(self, in):
        super().__init__()
        self.log_reg = nn.Sequential(
            nn.Linear(in,  1),
            nn.Sigmoid()
        )
    def forward(self, x):
        return self.log_reg(x)
```

Define the model as a class

Initialization method with parameters

Definition of the architecture

Forward computation of the model with inputs x

Training Models In PyTorch

```
model = LogisticRegression(16)
```

Initialization of the model

```
criterion = nn.BCELoss()
```

Cost function

```
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)
```

Optimizer

```
for t in range(n_epochs):

    y_pred = model(x)
    loss = criterion(y_pred, y)

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
```

Training loop for number of epochs

Forward propagation

Optimization step

Summary

- PyTorch makes computations on the run
- Dynamic computational graphs in Pytorch
- Just another framework, and similar to Tensorflow!