

Copyright Notice

These slides are distributed under the Creative Commons License.

DeepLearning.AI makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite DeepLearning.AI as the source of the slides.

For the rest of the details of the license, see

<https://creativecommons.org/licenses/by-sa/2.0/legalcode>



deeplearning.ai

Generative Models

Outline

- What are generative models
- Types of generative models



Generative Models vs. Discriminative Models

Discriminative models

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

mirror each other

Generative Models vs. Discriminative Models



Generative models



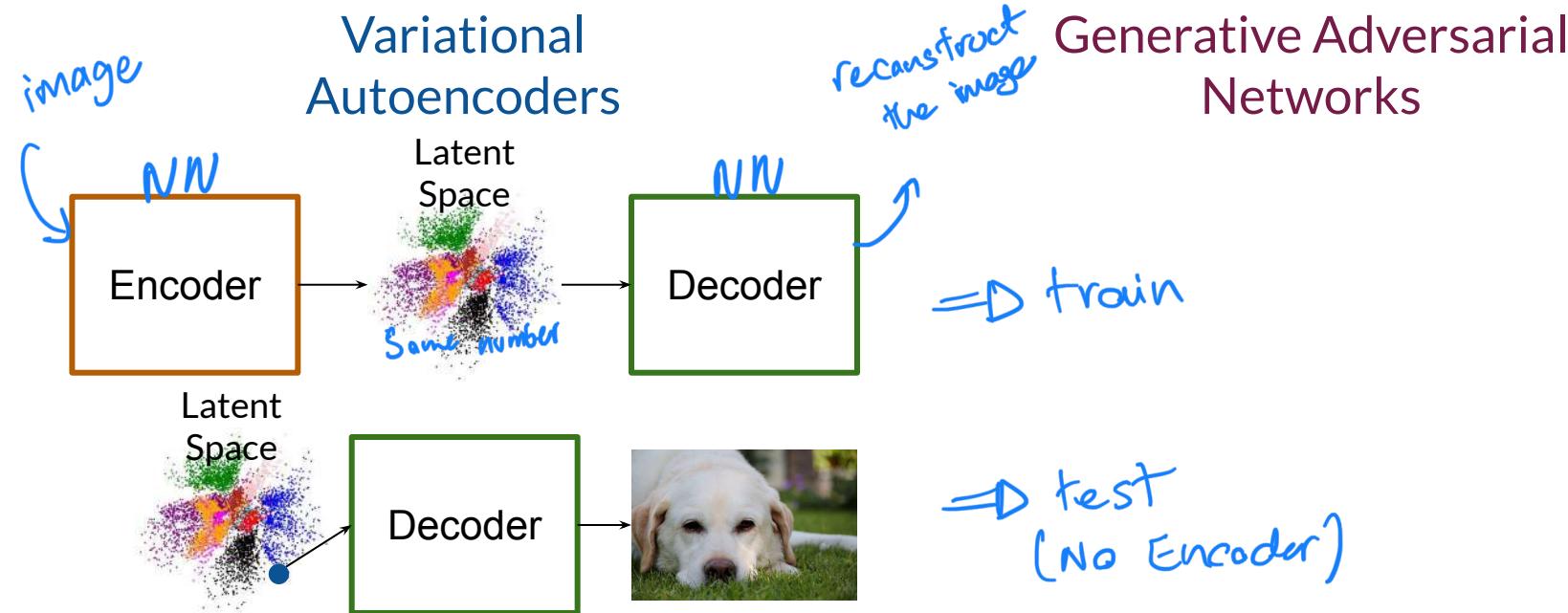
Noise Class Features

$$\xi, Y \rightarrow X$$

$$P(X|Y)$$

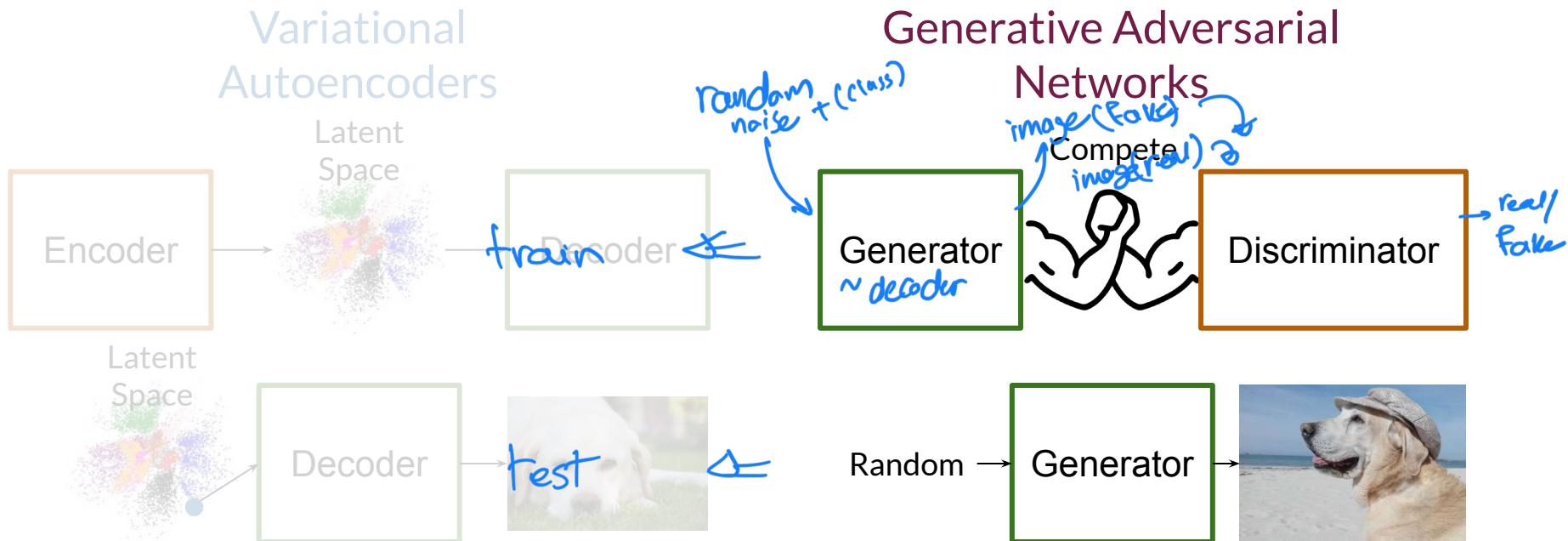
NN (neural network)

Generative Models



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

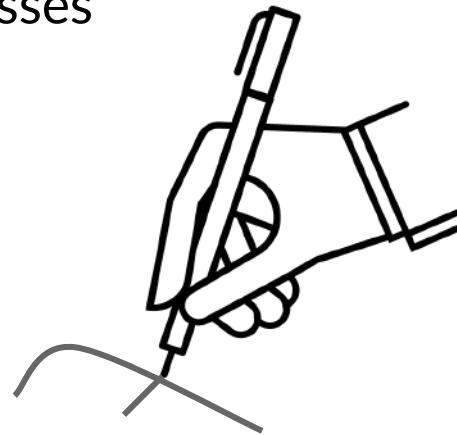
Generative Models



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

Summary

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



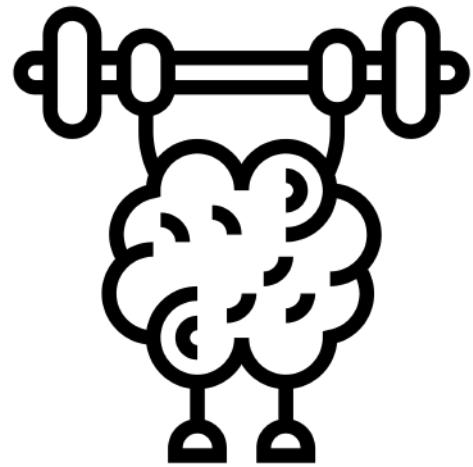


deeplearning.ai

Real Life GANs

Outline

- Cool applications of GANs
- Major companies using them



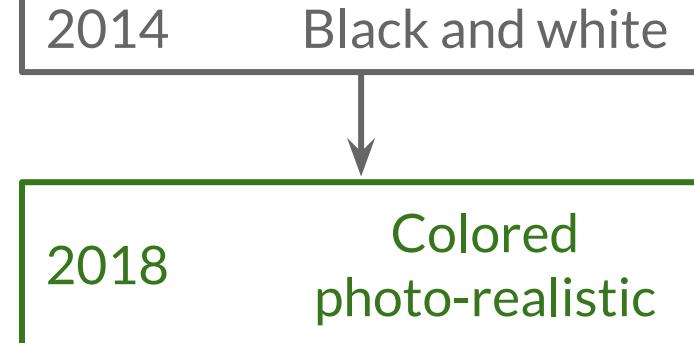
GANs Over Time



Ian Goodfellow
@goodfellow_ian

4.5 years of GAN progress on face generation.

arxiv.org/abs/1406.2661 arxiv.org/abs/1511.06434
arxiv.org/abs/1606.07536 arxiv.org/abs/1710.10196
arxiv.org/abs/1812.04948



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 image
has heat

StyleGAN2

RAPID APPEARANCE AND
DISAPPEARANCE OF RED DOT

ALIENS

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

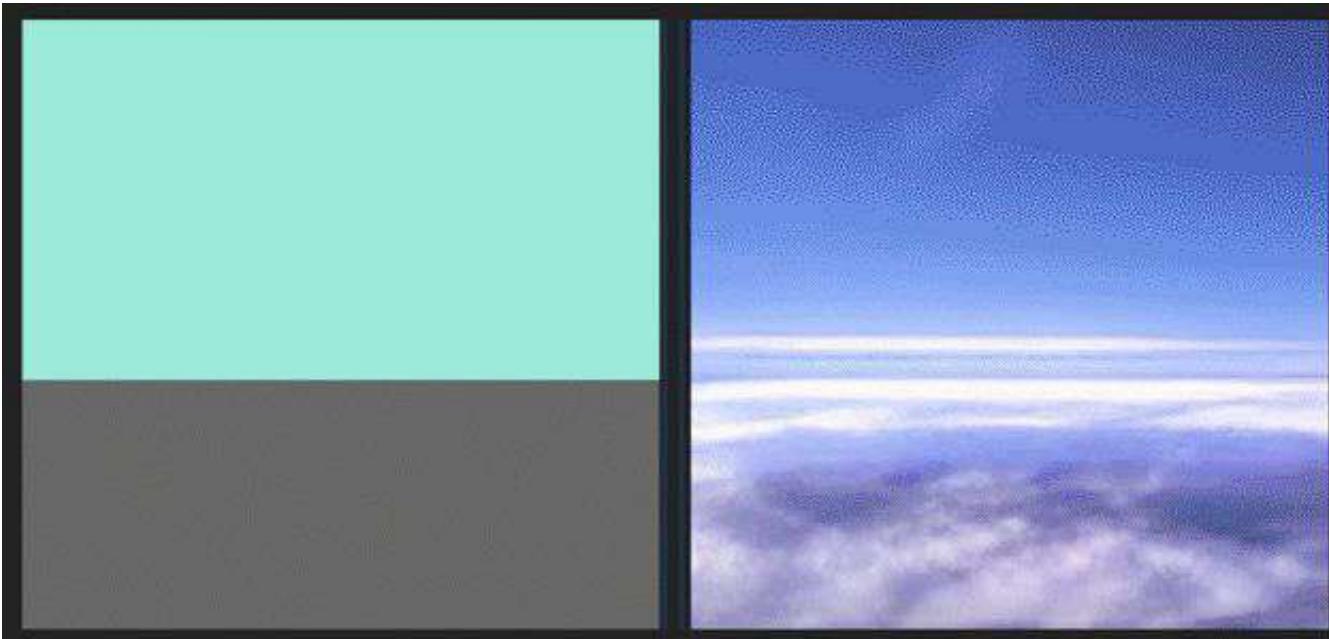
From one domain to another

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

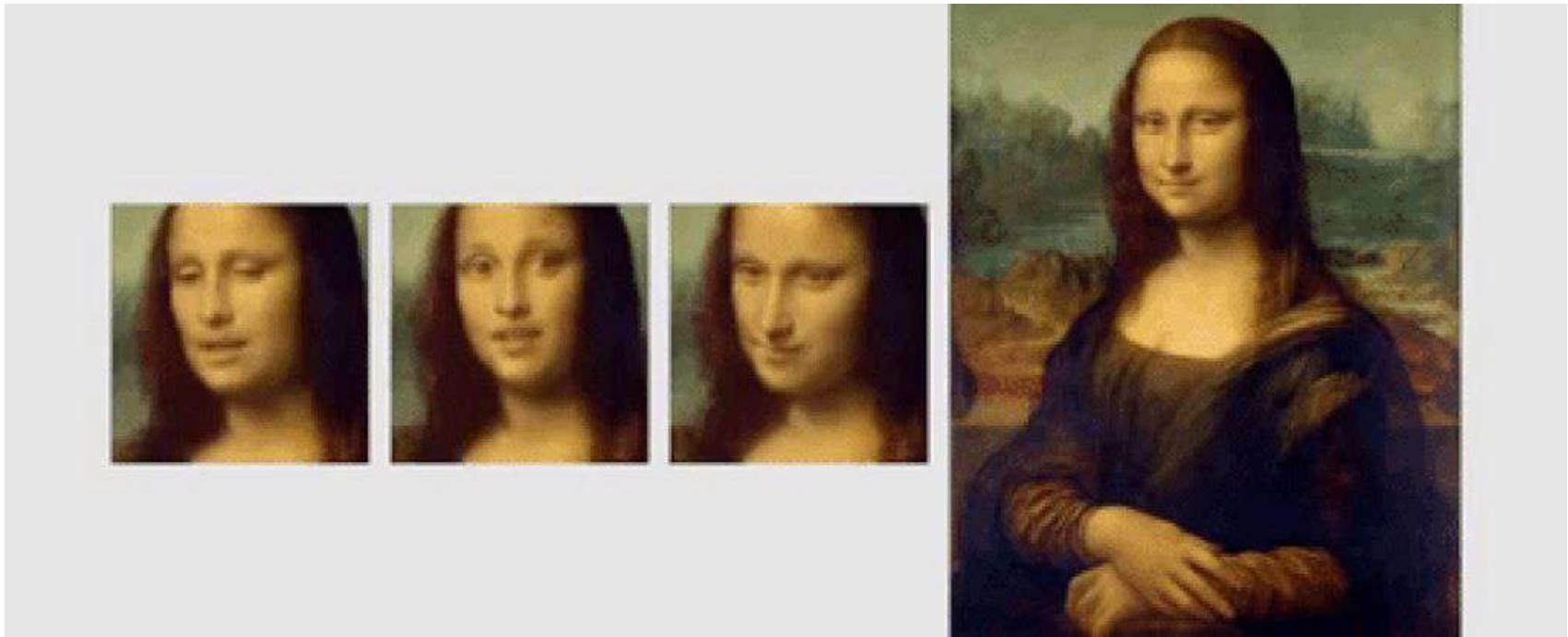


GauGAN

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.

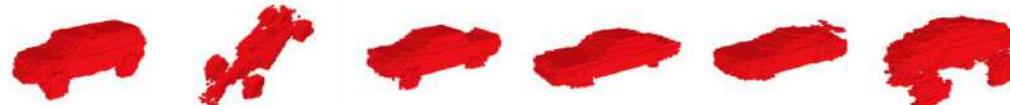
GANs for 3D Objects

Chair

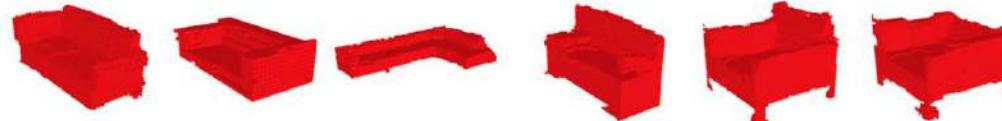


3D-GAN

Car



Sofa



Table



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



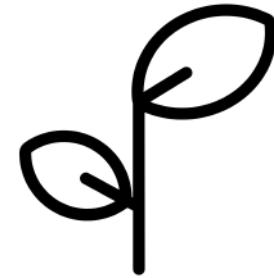
Image Filters



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

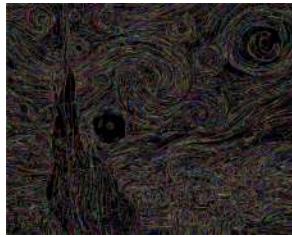
Outline

- 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

Discriminator learns to distinguish
real from *fake*

Fake

Real



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 me!)

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



I don't know what I'm doing

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*



I don't know what I'm doing, either

The Game Is On!



5% Real



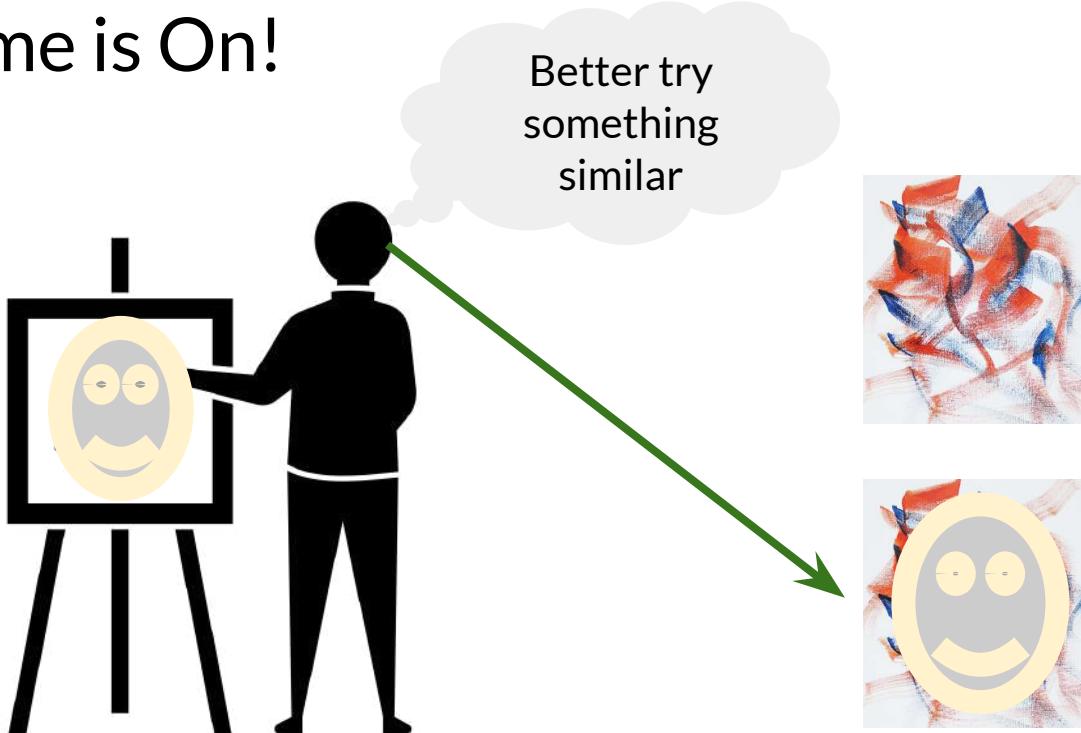
40% Real



80% Real



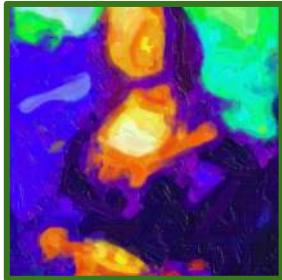
The Game is On!



The Game Is On!



30% Real



60% Real

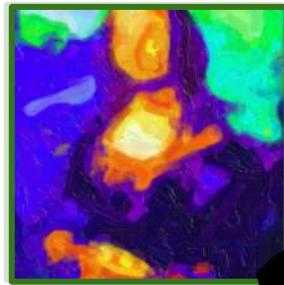


95% Real

The Game Is On!

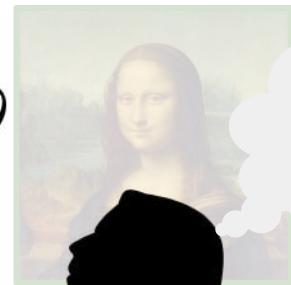


30% Real



60% Real

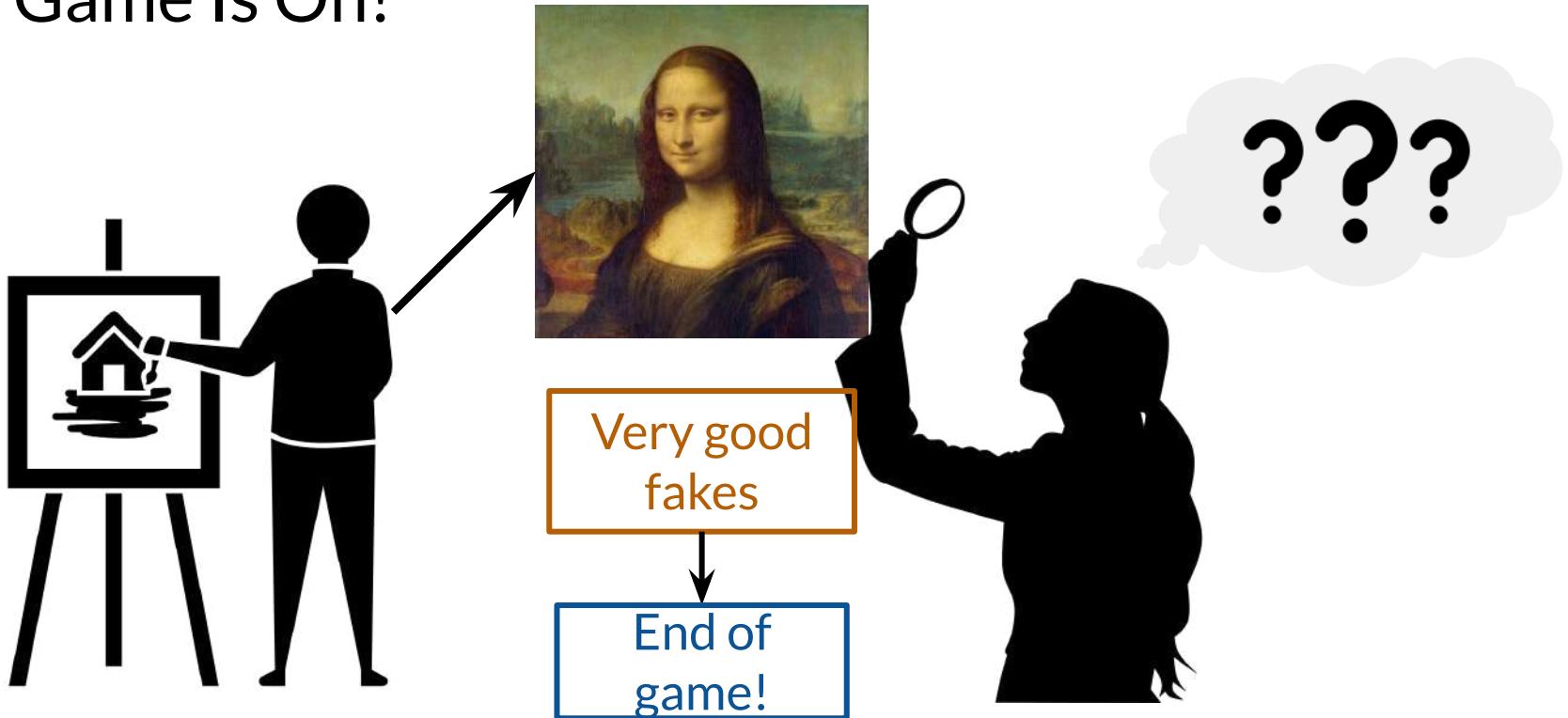
Wrong!



Won't fool me
again



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



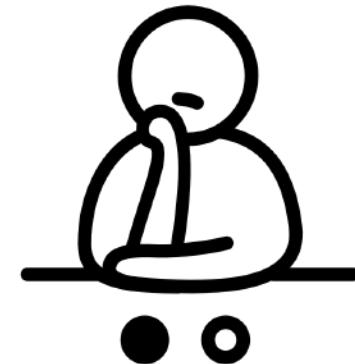


deeplearning.ai

Discriminator

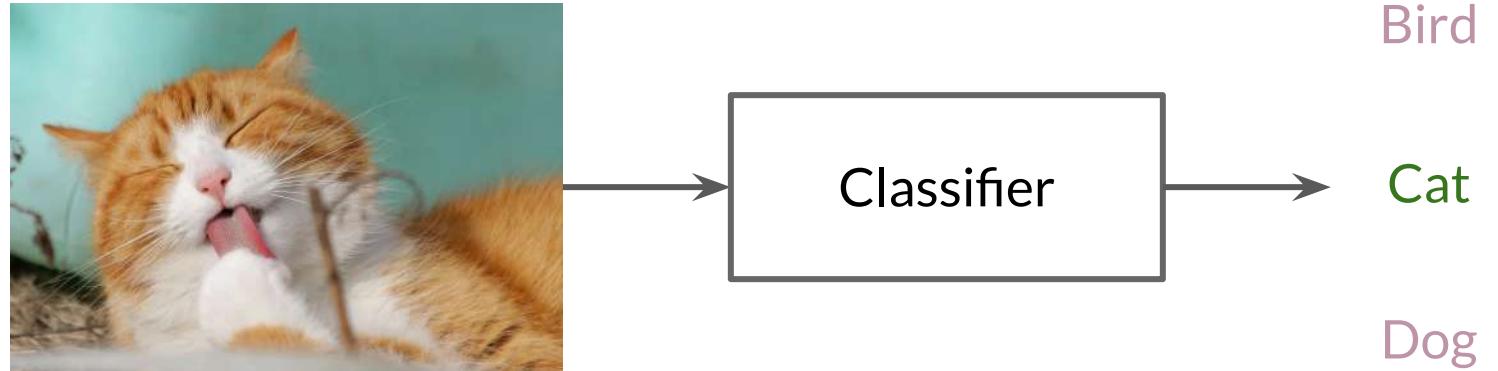
Outline

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



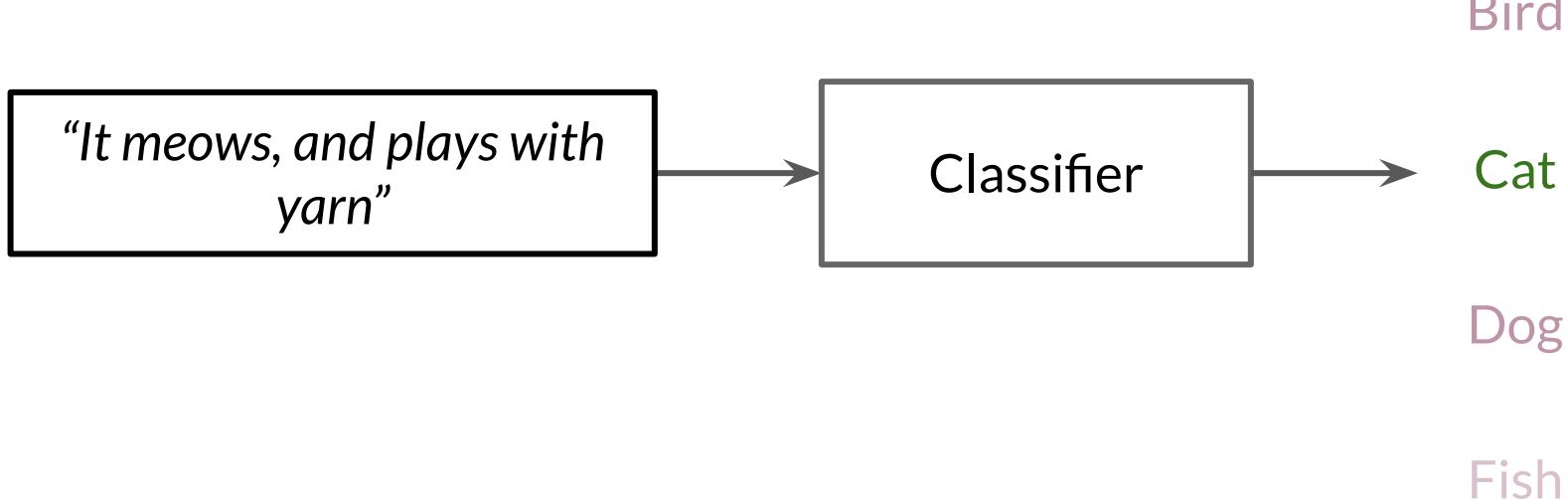
Classifiers

Distinguish between different classes

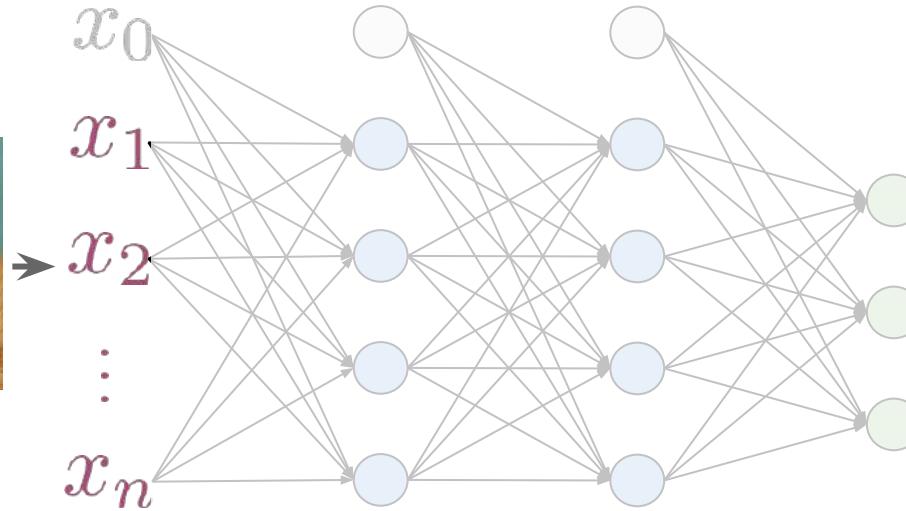


Classifiers

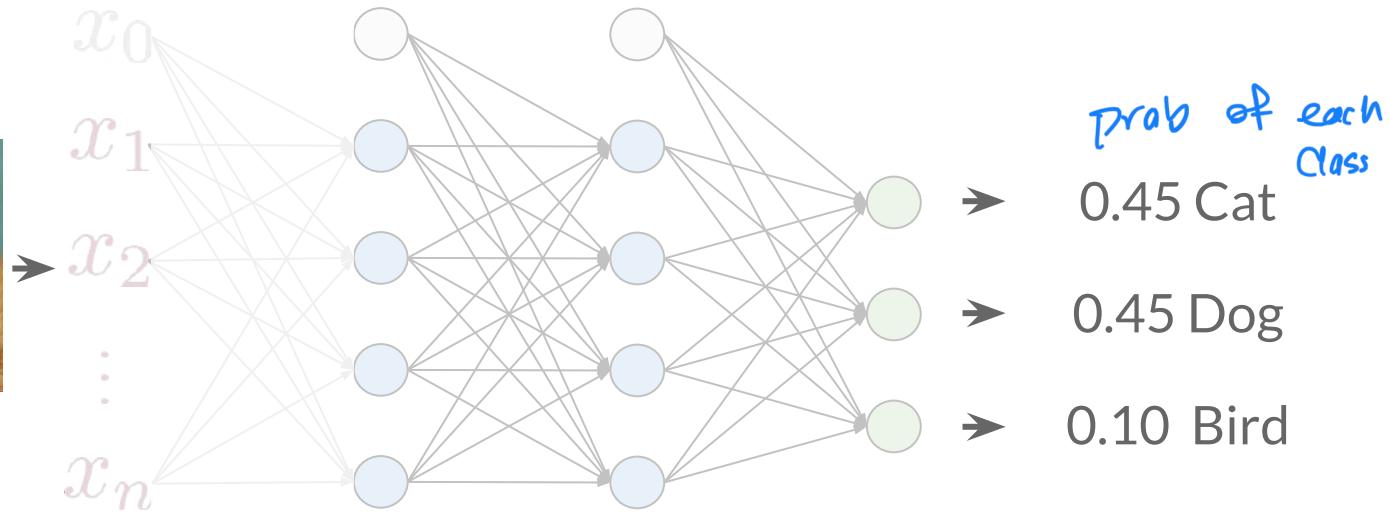
Distinguish between different classes



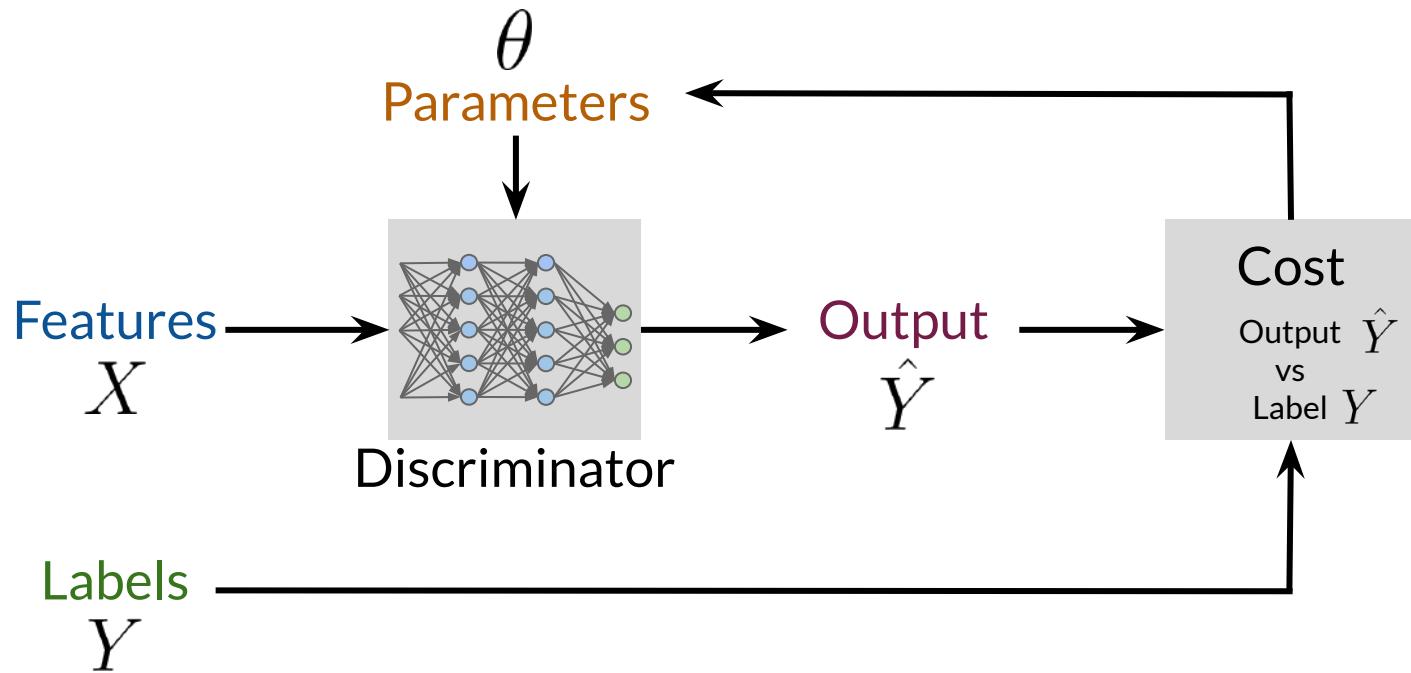
Neural Networks



Neural Networks



Classifiers (training)



Classifiers

Turtle

Bird

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



Dog

Fish

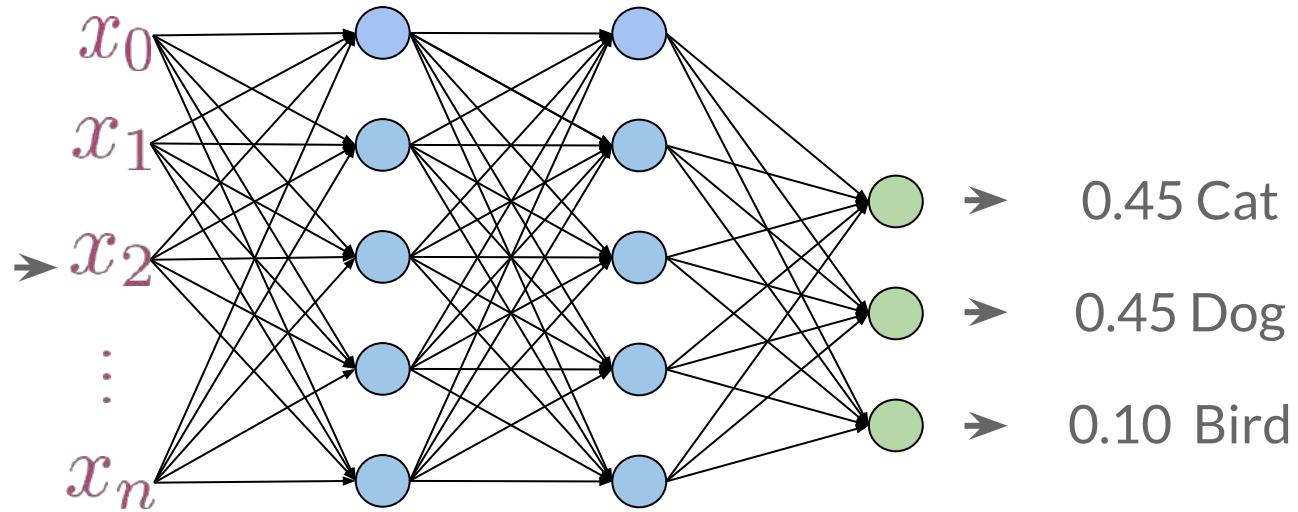
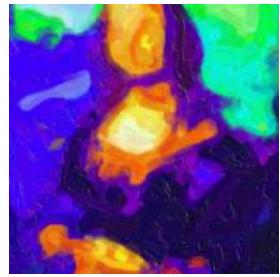
Classifiers

$$P(Y | X)$$

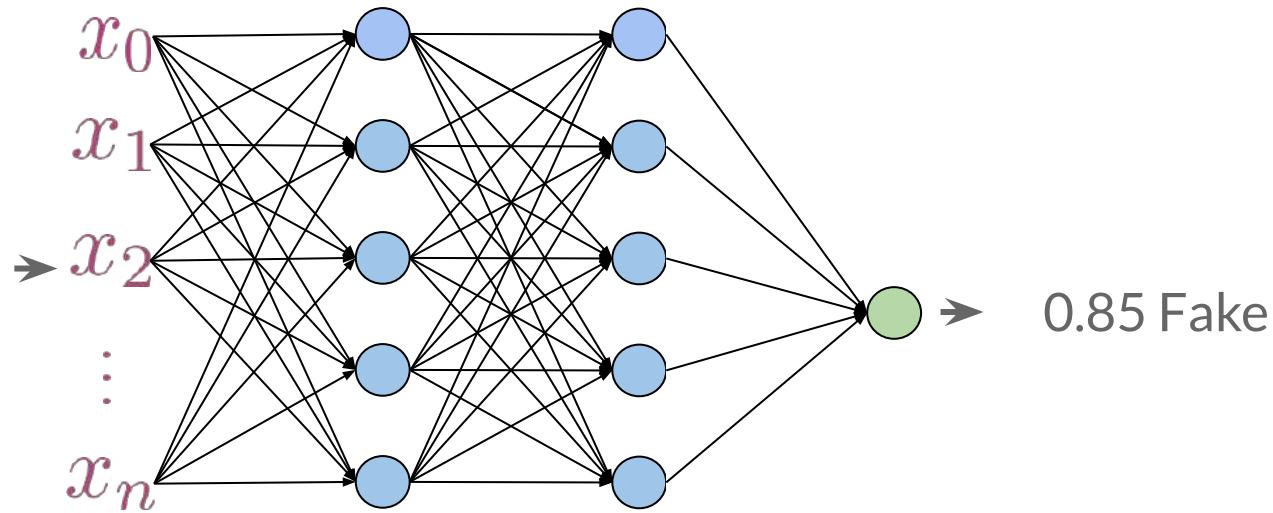
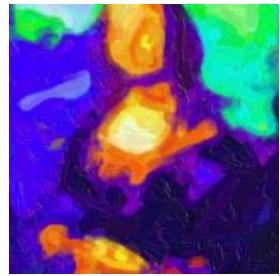
Class Features

A diagram illustrating the components of a classifier. The formula $P(Y | X)$ is shown, where Y is colored purple and X is colored blue. Below the formula, the word "Class" is aligned under Y and the word "Features" is aligned under X . An arrow points from the vertical bar separating Y and X to a rectangular box with a brown border. Inside the box, the text "Conditional Probability" is written in brown.

Discriminator



Discriminator



Discriminator

$$P(\text{Fake} \mid X)$$

Class Features

Discriminator

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

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

Generates examples of the class

Bird

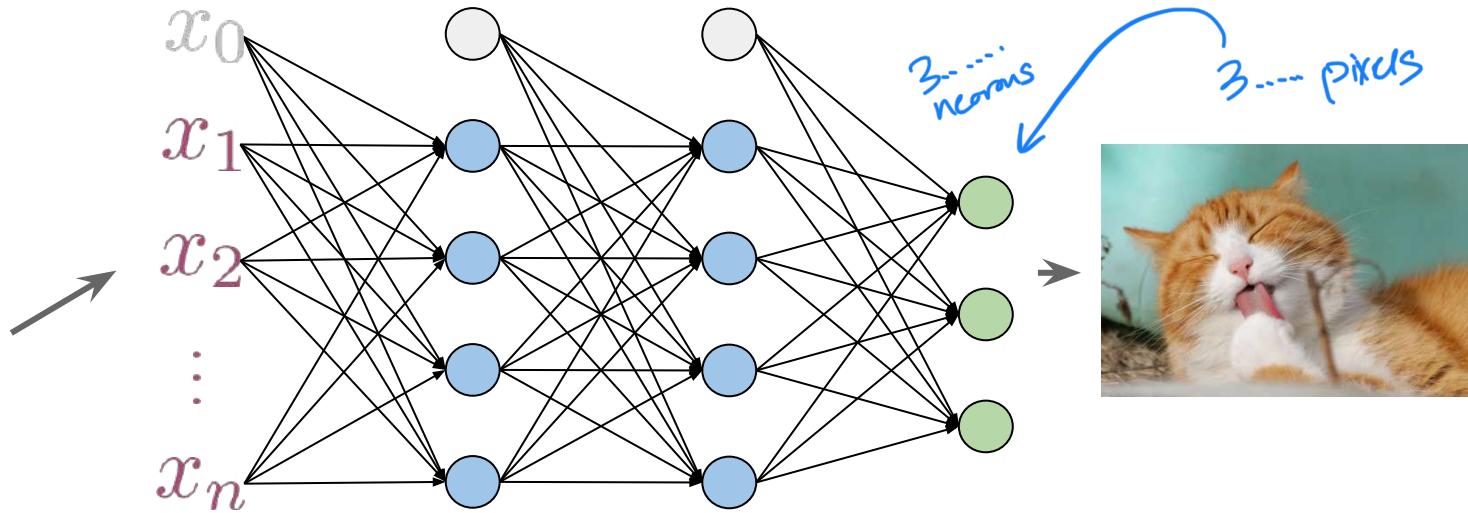
Cat

Dog

Fish



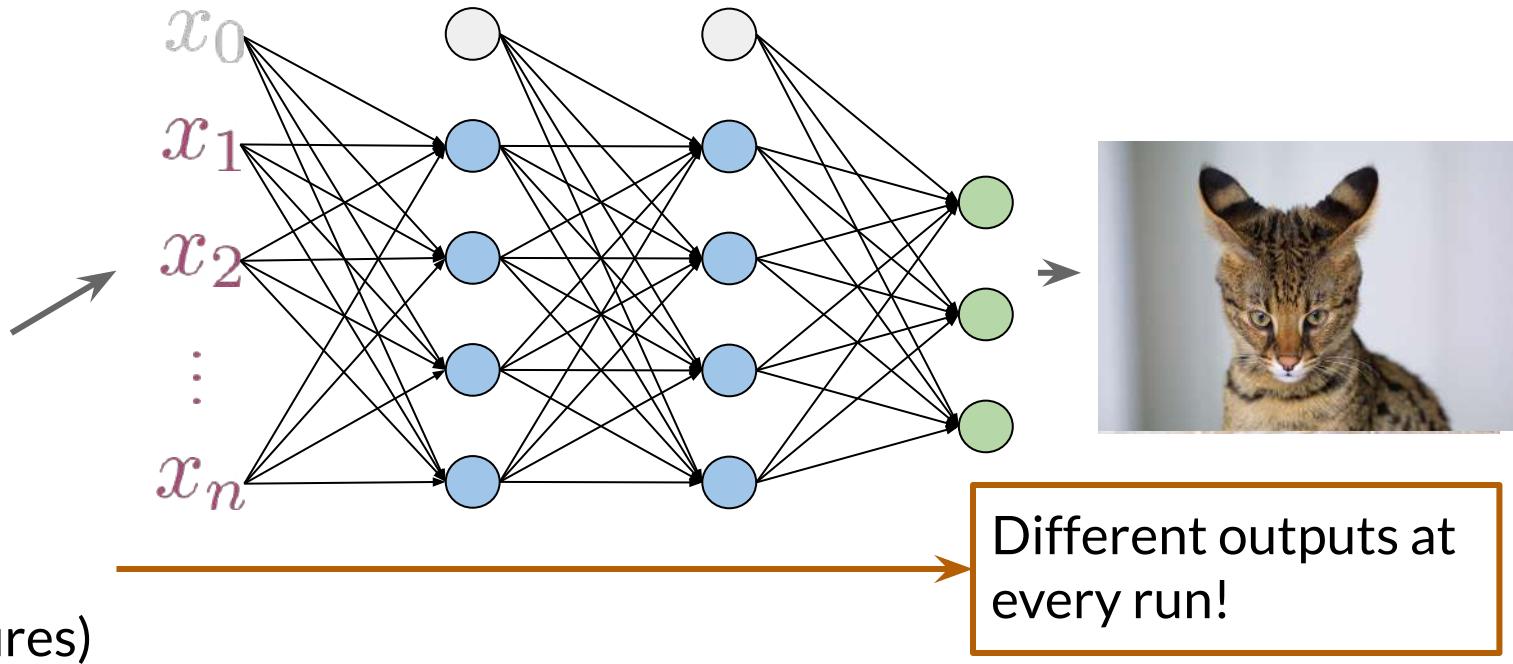
Neural Networks



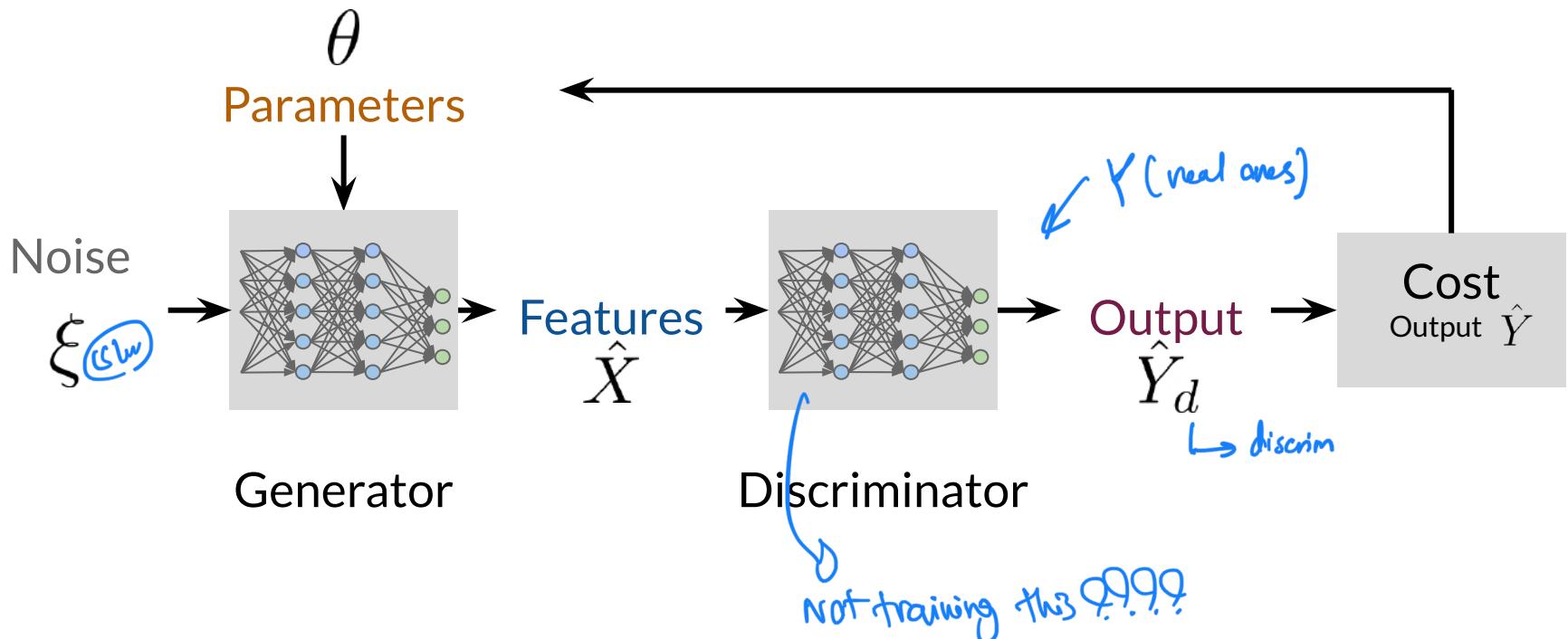
Noise
(random features)

For diff output every time

Neural Networks



Generator: Learning



Sampling



the θ from f



Generator

$$P\left(\begin{array}{c|c} \text{Image} & \text{Label} \\ \hline \text{Cat} & \text{Cat} \\ \text{Dog} & \text{Dog} \\ \text{Bird} & \text{Bird} \\ \text{Turtle} & \text{Turtle} \end{array}\right)$$

Generator

$$P(X \mid Y)$$

Features Class

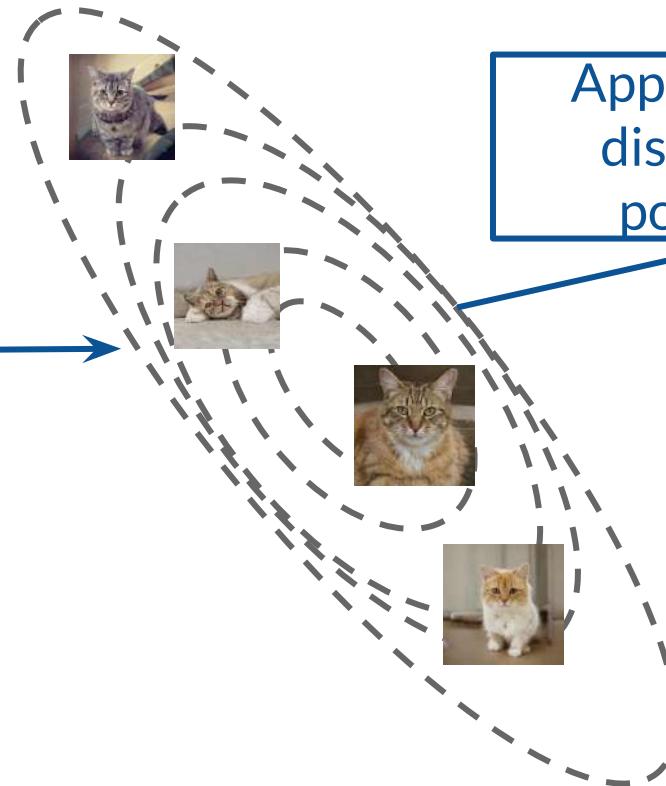
A diagram illustrating a conditional probability distribution. The expression $P(X \mid Y)$ is shown, with X in blue and Y in purple. Below X is the label "Features" in blue, and below Y is the label "Class" in purple. A black arrow points from the right side of the \mid symbol to a rectangular box with a brown border. Inside the box, the text "Conditional Probability" is written in brown.

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

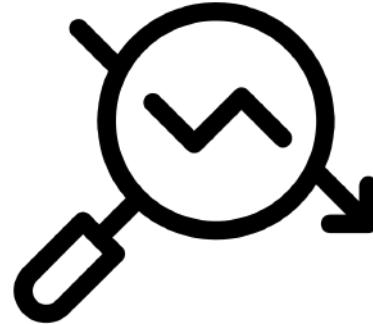
BCE Cost Function

Outline

for train GAN (Binary)

- Binary Cross Entropy (BCE) Loss equation by parts
- How it looks graphically

Fake/real



BCE Cost Function

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

The diagram illustrates the components of the BCE Cost Function. A large purple circle highlights the term $\frac{1}{m} \sum_{i=1}^m$. Arrows point from this term to four labels: 'Prediction' (under $y^{(i)}$), 'Label' (under $1 - y^{(i)}$), 'Features' (under $x^{(i)}$), and 'Parameters of discrim' (under θ). Below the highlighted term is a box containing the text 'Average loss of the whole batch' and 'm: # of examples'.

Average loss of the whole batch
m: # of examples

Prediction

Label

Features

Parameters of discrim

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

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

true label *log prediction*

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

Fake
real

Relevant when
the label is 1

BCE Cost Function

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

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

$$\begin{array}{c|c} y^{(i)} & h(x^{(i)}, \theta) \\ \hline (1 - y^{(i)}) \log(1 - h(x^{(i)}, \theta)) & \end{array}$$

1	any	0
0	0.01	~0
0	~1	-inf

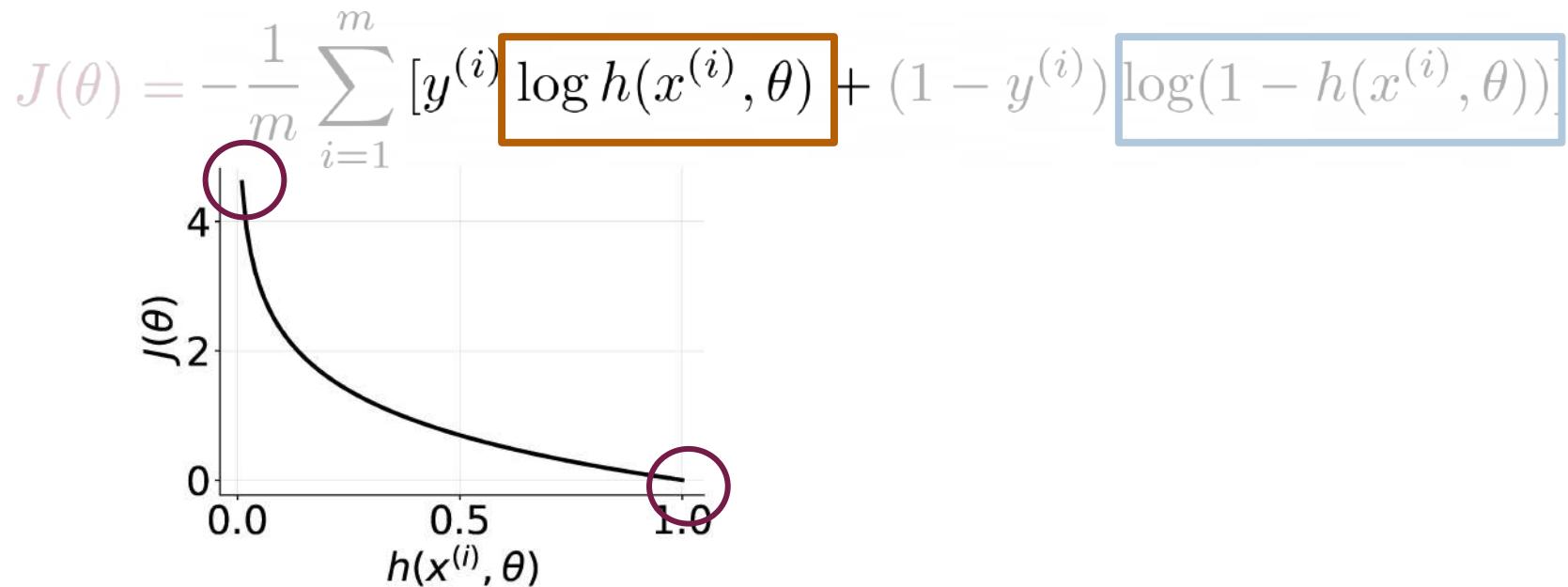
Relevant when
the label is 0

BCE Cost Function

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

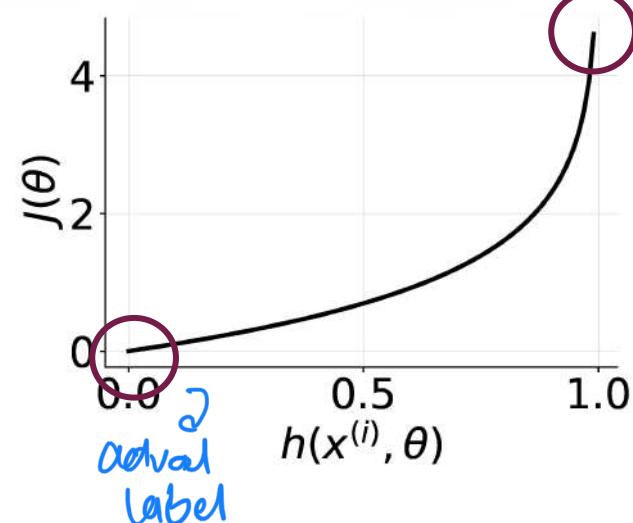
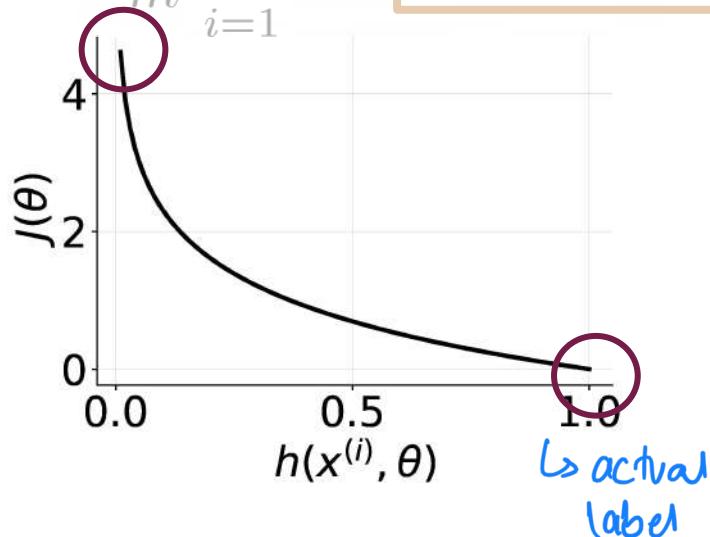
Ensures that the cost is always greater or equal to 0

BCE Cost Function



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



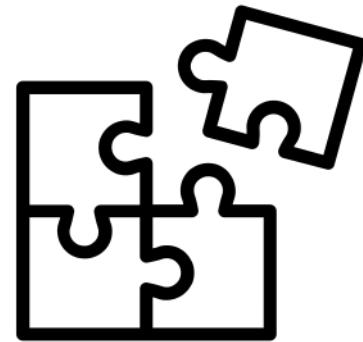


deeplearning.ai

Putting It All Together

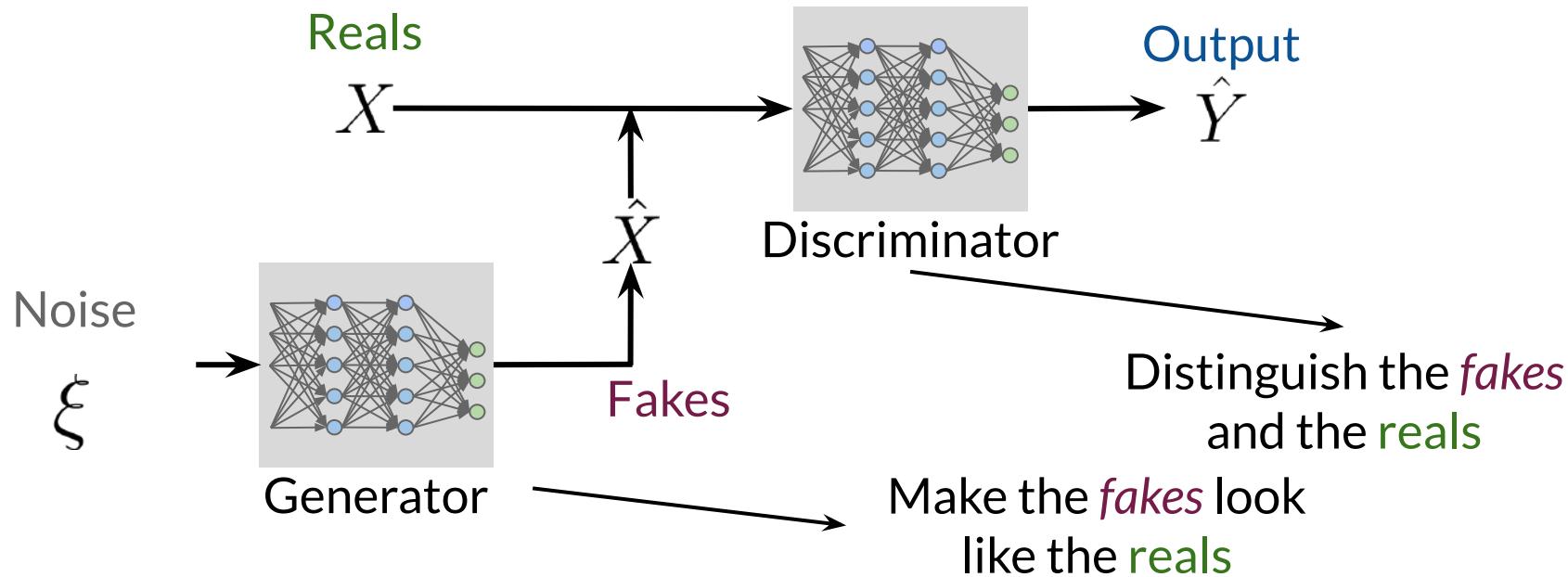
Outline

- How the whole architecture looks
- How to train GANs

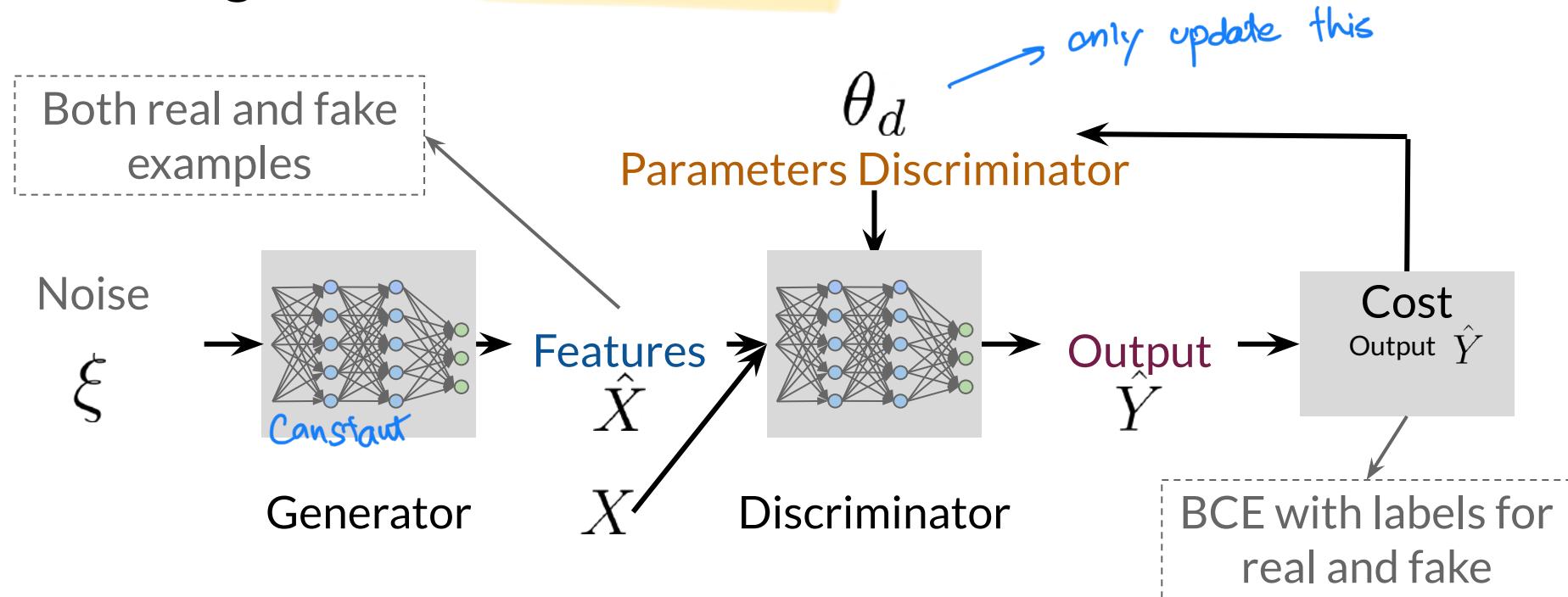


We alter the training of gener & discrim

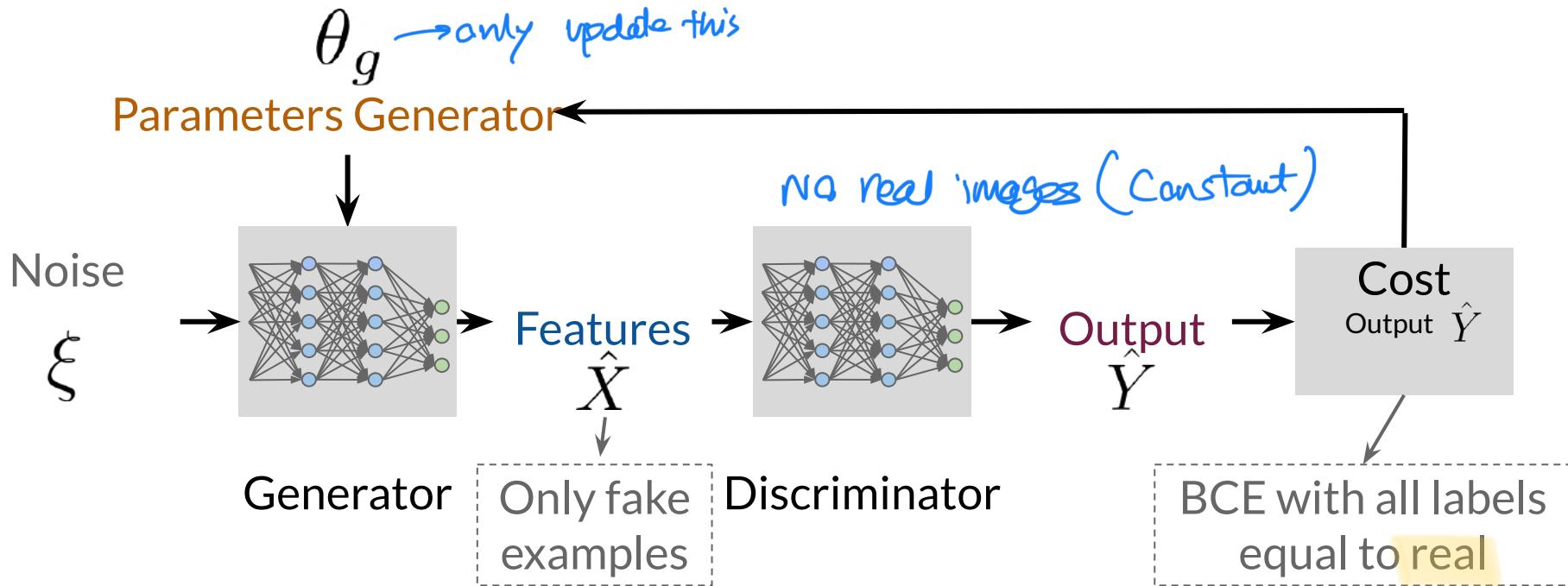
GANs Model



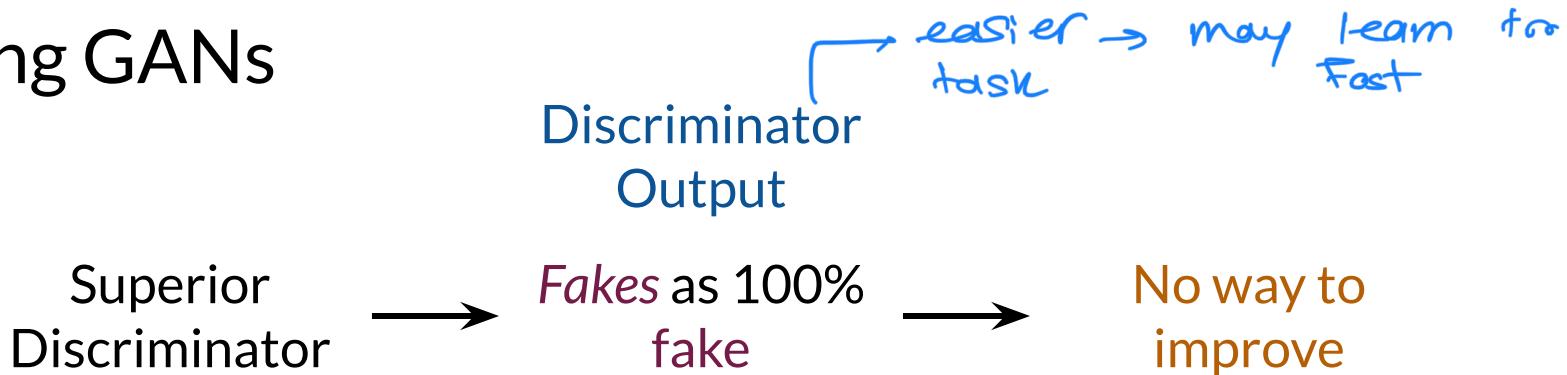
Training GANs: Discriminator *phase*



Training GANs: Generator



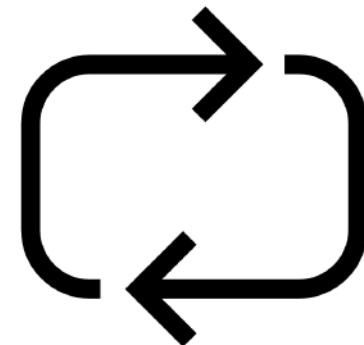
Training GANs



discrim & genor 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



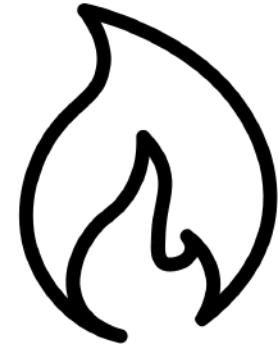


deeplearning.ai

Intro to PyTorch (Optional)

Outline

- Comparison with TensorFlow
- Defining Models
- Training



PyTorch vs TensorFlow

PyTorch	TensorFlow
Imperative, computations on the go	Symbolic, first define and then compile
<pre>A, B = 1, 2 C = A + B print(C)</pre> <p>3</p> <p>Dynamic Computational Graphs</p>	<pre>C = A + B f = compile(C) print(f(A = 1, B = 2))</pre> <p>3</p> <p>Static Computational Graphs</p>
<p>Tensorflow > 2.0 moves toward PyTorch by including Eager Execution</p>	

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):
```

Training loop for number of epochs

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

Forward propagation

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

Optimization step

Summary

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

