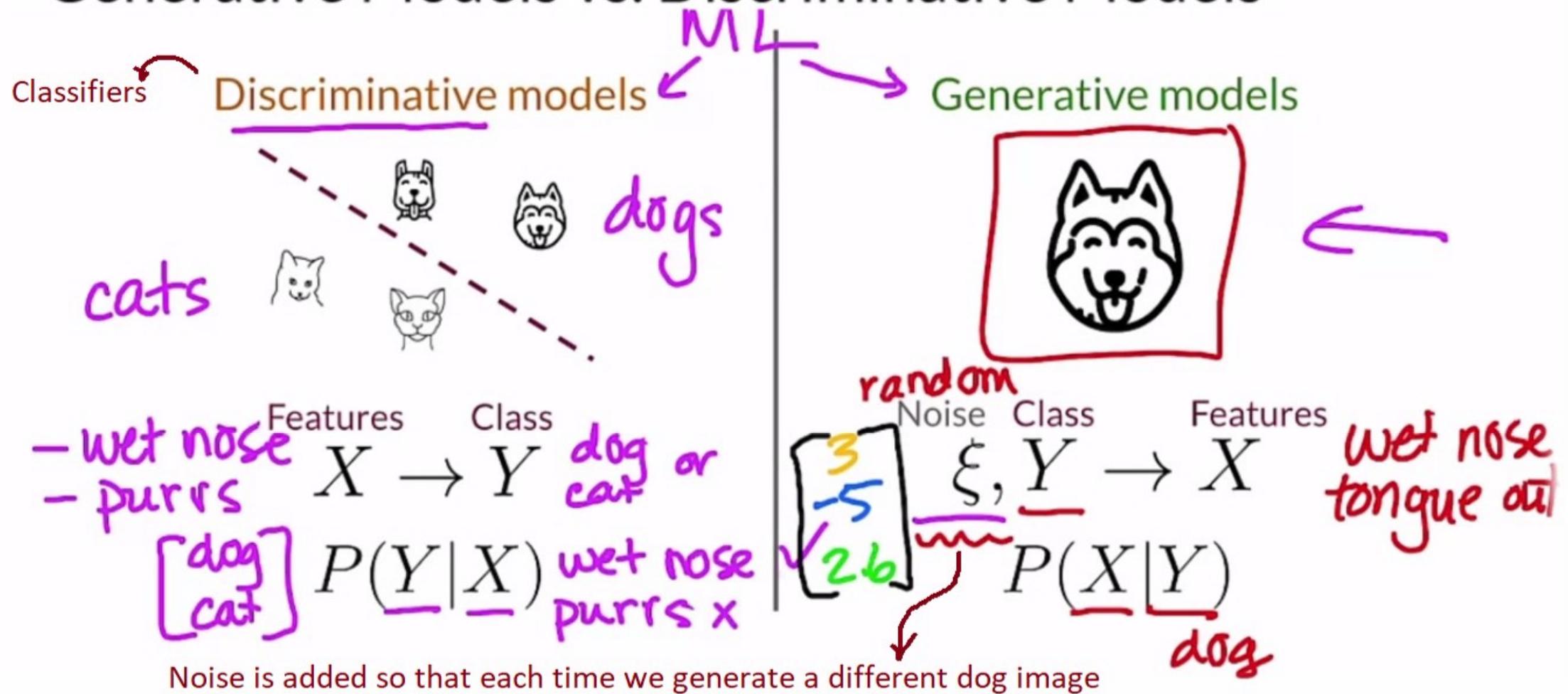




deeplearning.ai

Generative Models

Generative Models vs. Discriminative Models



Generative Models vs. Discriminative Models



Generative models



Noise Class Features
 $\xi, Y \rightarrow X$

$P(X|Y)$

Each time we run our generative model, we end generating a new image of dog

Generative Models vs. Discriminative Models



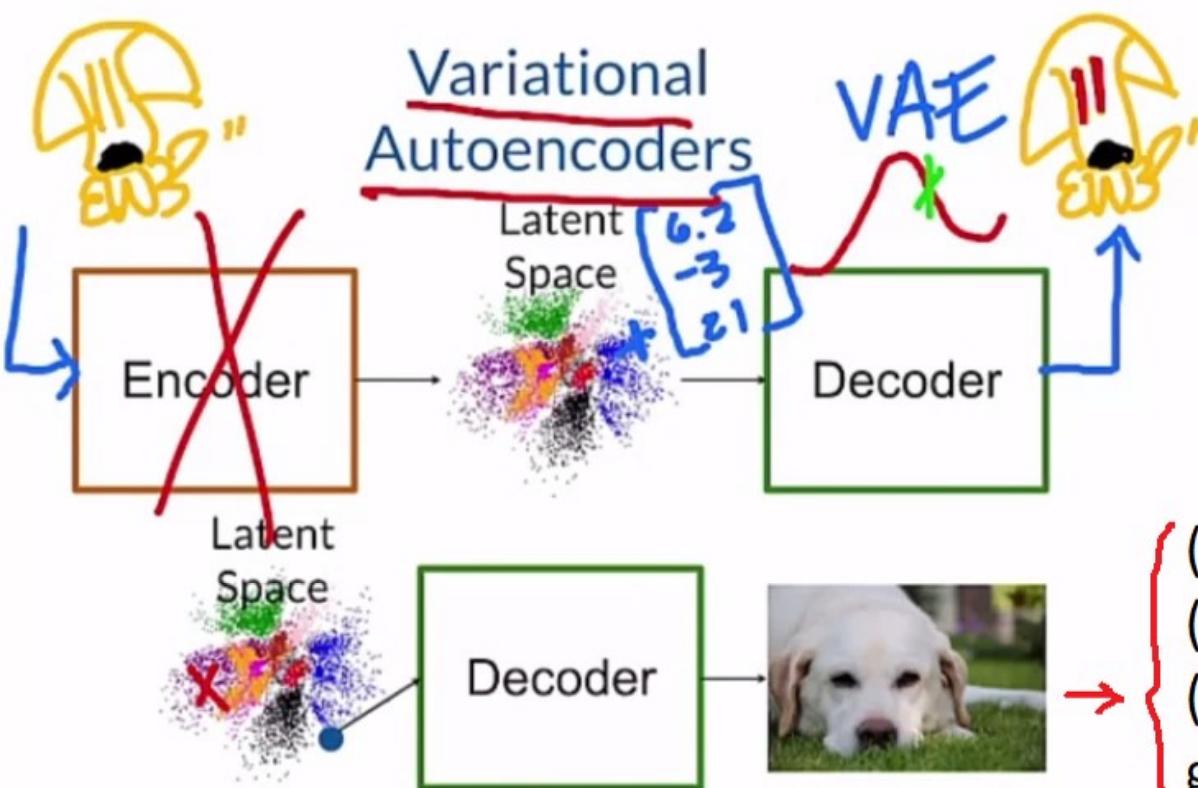
Generative models



Noise Class Features
 $\xi, Y \rightarrow X$

$P(X|Y)$

Generative Models



Generative Adversarial
Networks

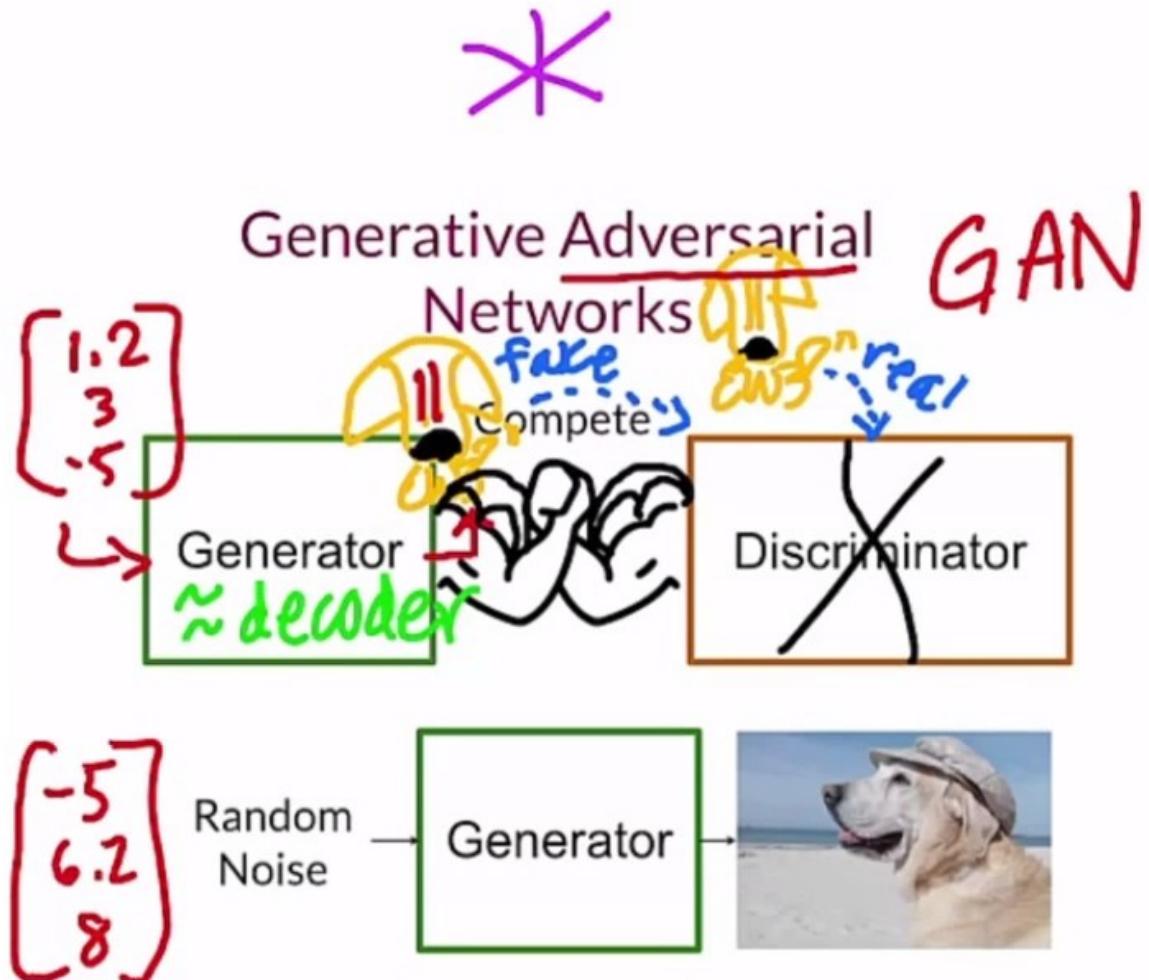
GAN

- (1) Encoders and decoders are NN
- (2) After training, the encoders is locked
- (3) Then we randomly sample encodings to generate features/images.
- (4) This may contain some noise

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

Generative Models

- (1) Generator takes random noise vector and an optional class to generate
- (2) Discriminator try to figures out real and fake images.
- (3) They will compete against each other and learn from each other
- (4) After training we do not need discrimator.
- (5) Now model can generate realistic images using random noise



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

Summary

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

realistic
discriminator
real *fake*



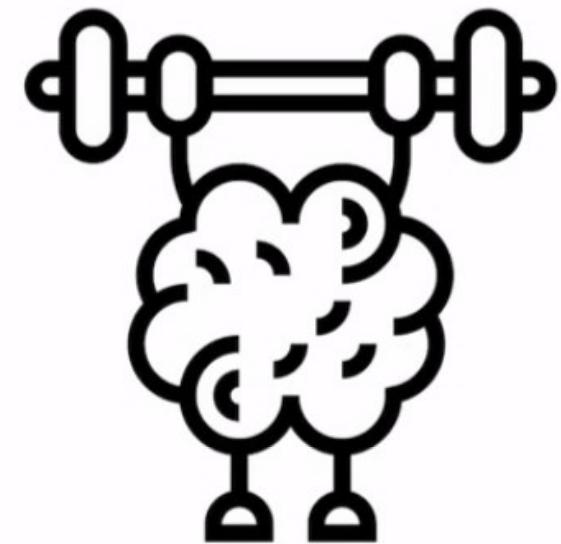


deeplearning.ai

Real Life GANs

Outline

- Cool applications of GANs
- Major companies using them



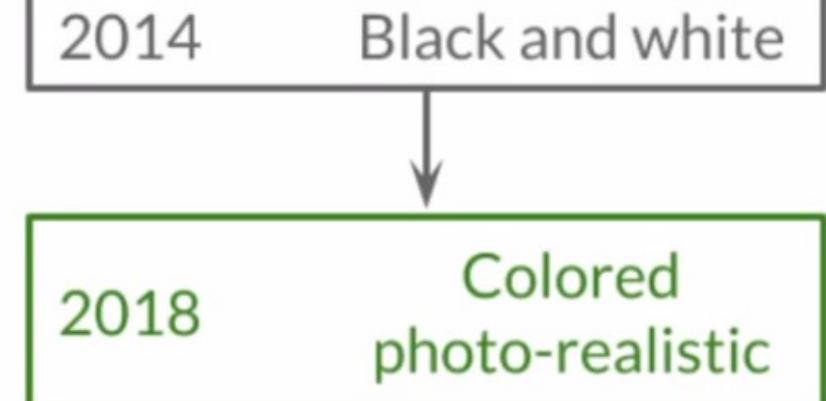
GANs Over Time



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

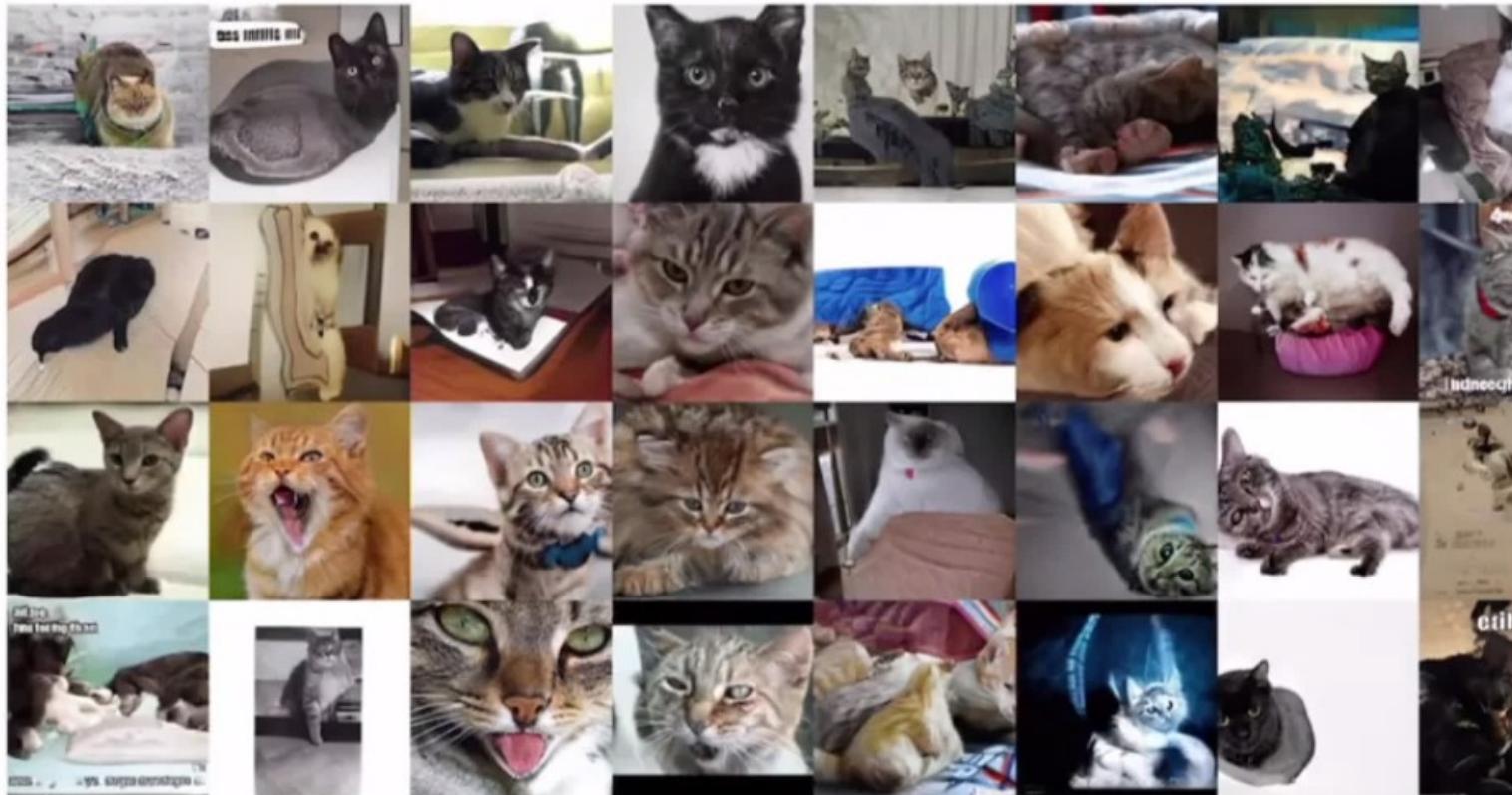


Face Generation
StyleGAN2

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.

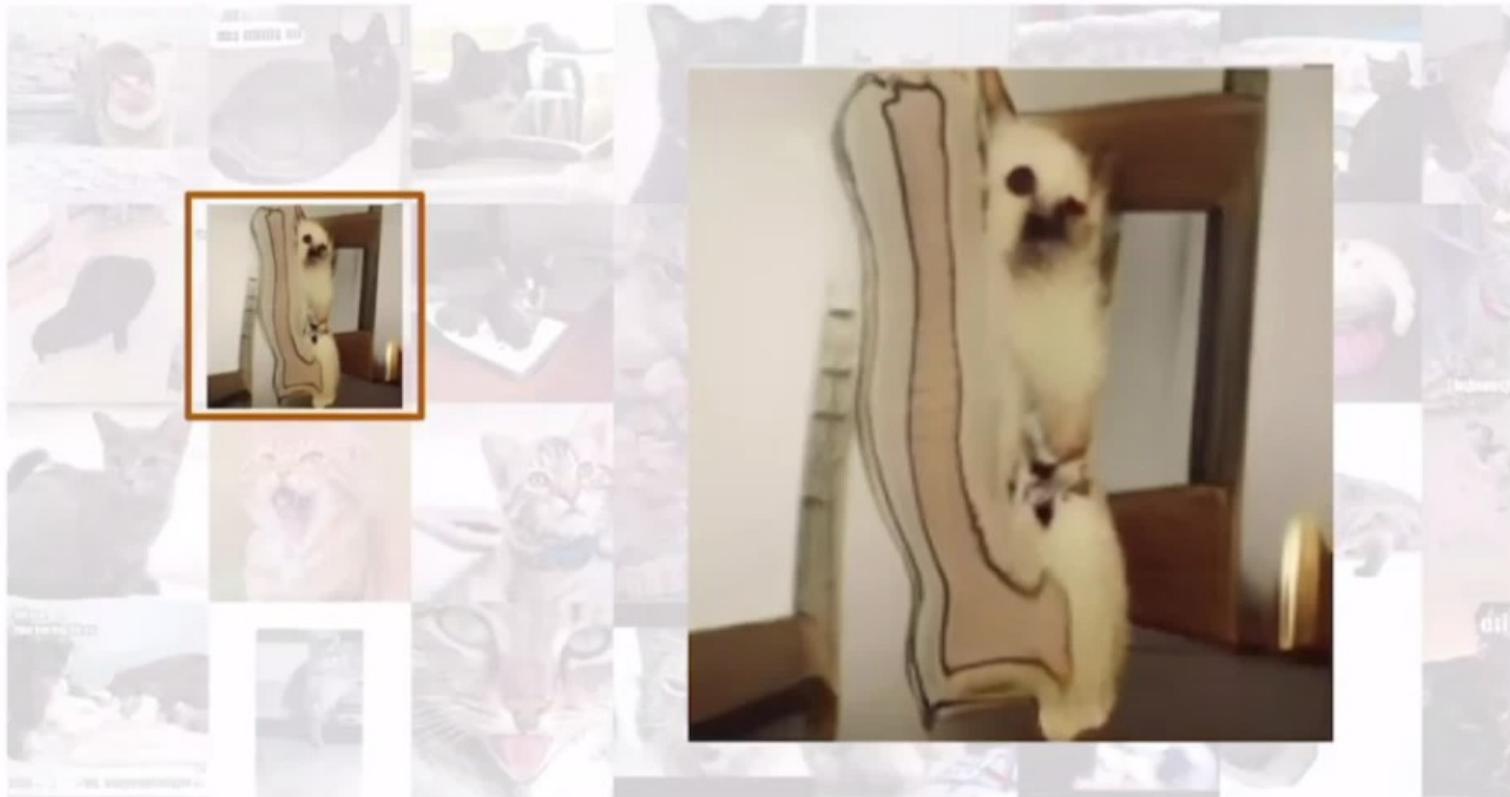
GANs Over Time



StyleGAN2

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



StyleGAN2

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 can only generate visualization (images) and not text

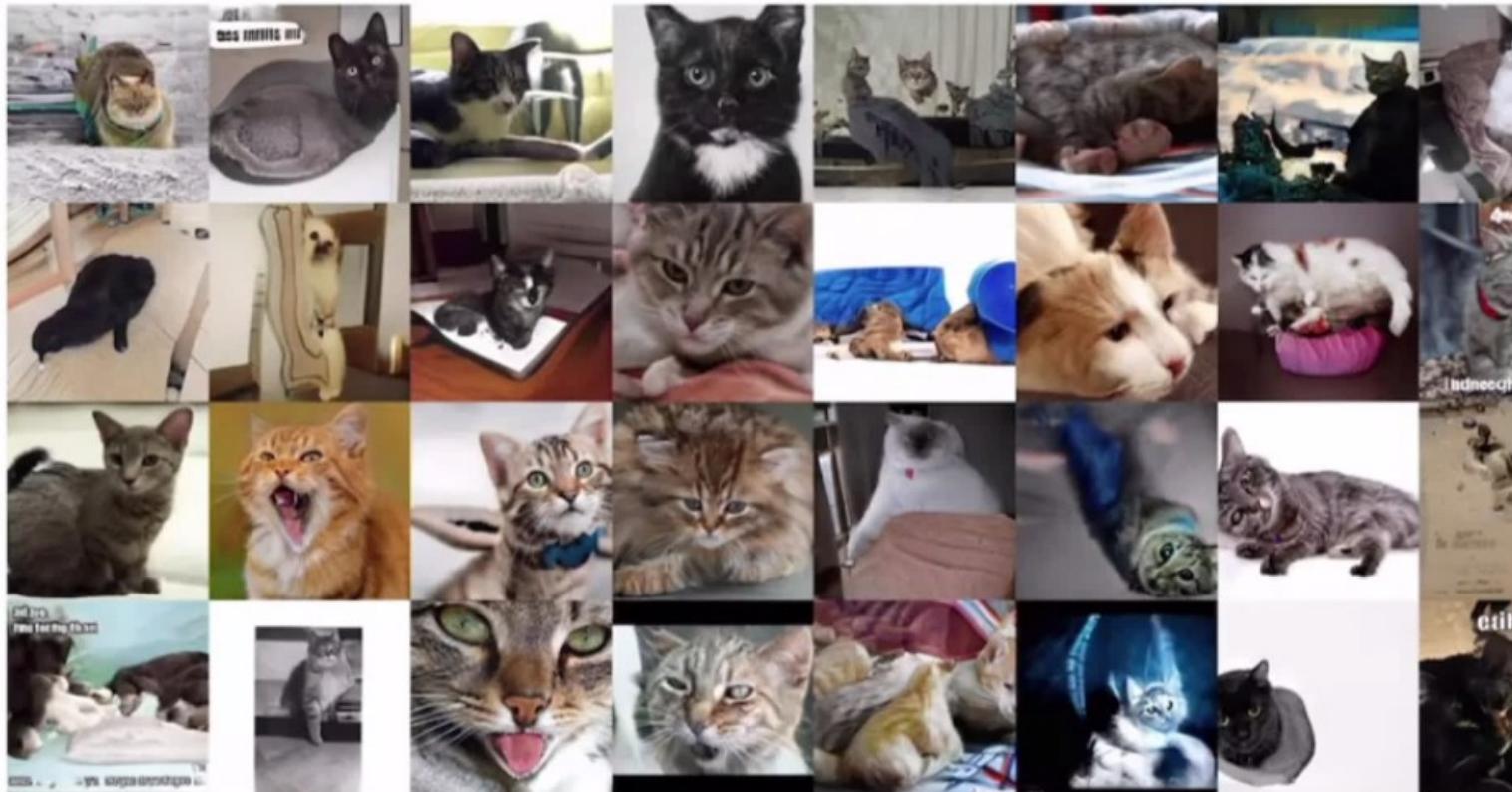
GANs Over Time



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 Over Time



StyleGAN2



Mimics the distribution of the training data

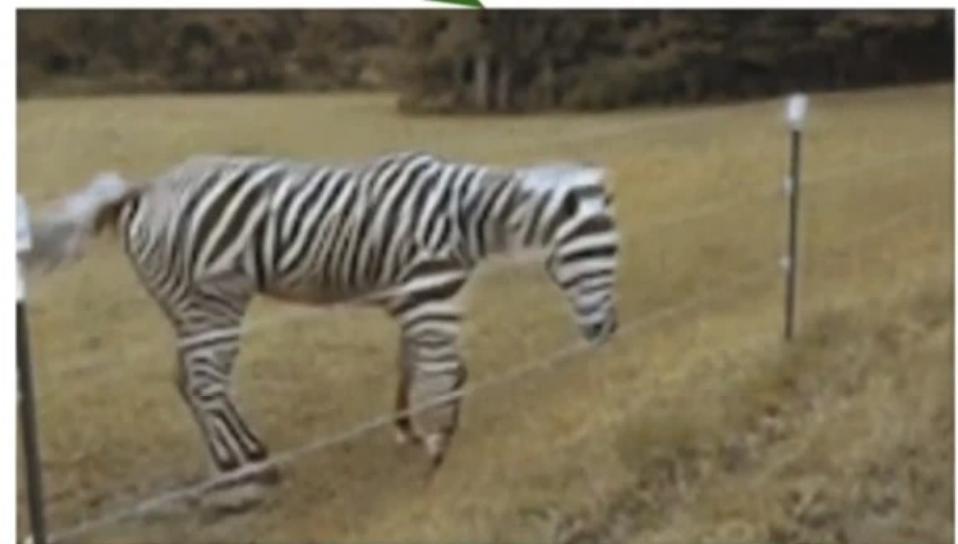
Karras, Tero, et al. "Analyzing and improving the image quality of stylegan." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020.

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

GANs for Image Translation

From one domain to
another

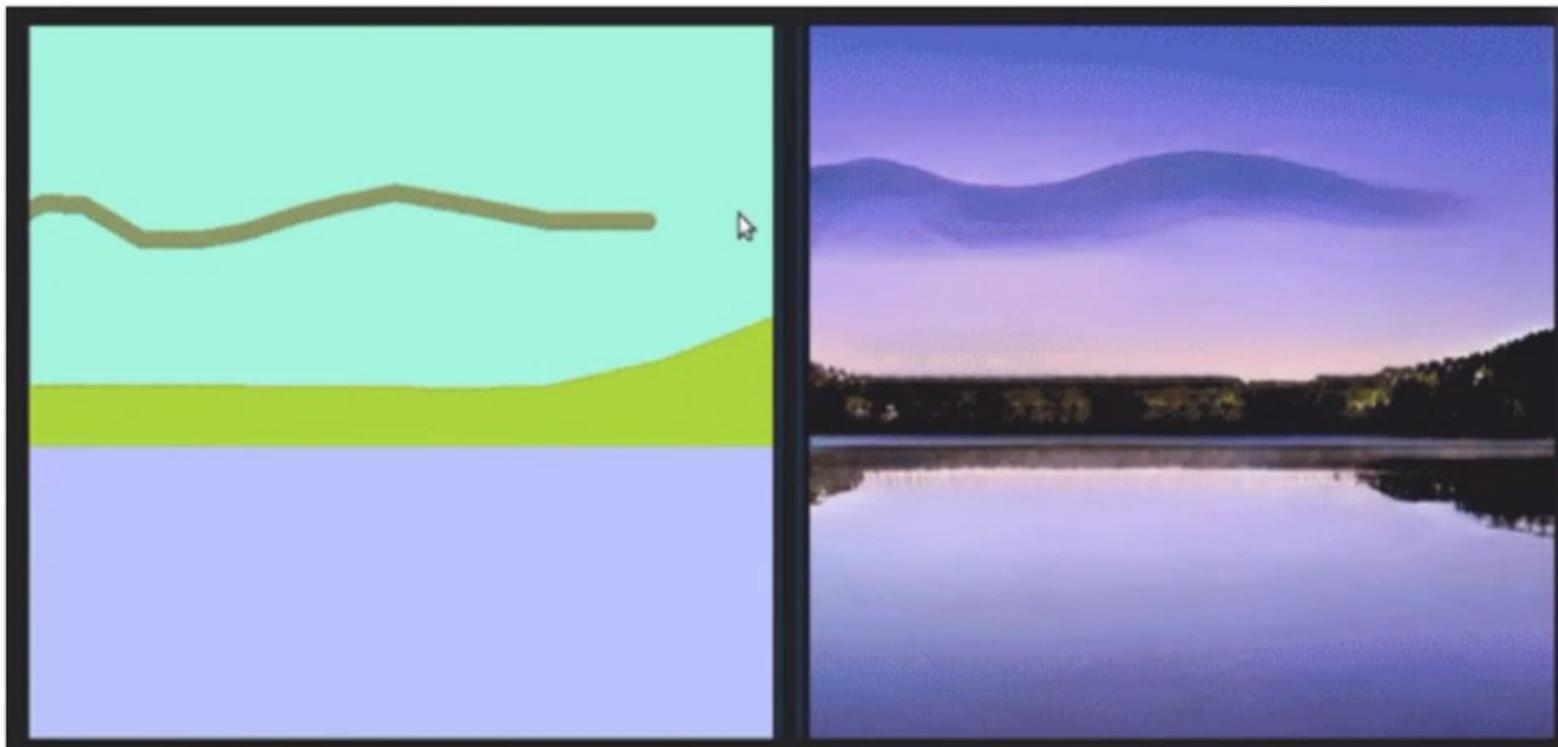
CycleGAN



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

From rough sketch drawing as input, GANs can produce realistic images

GANs for Image Translation

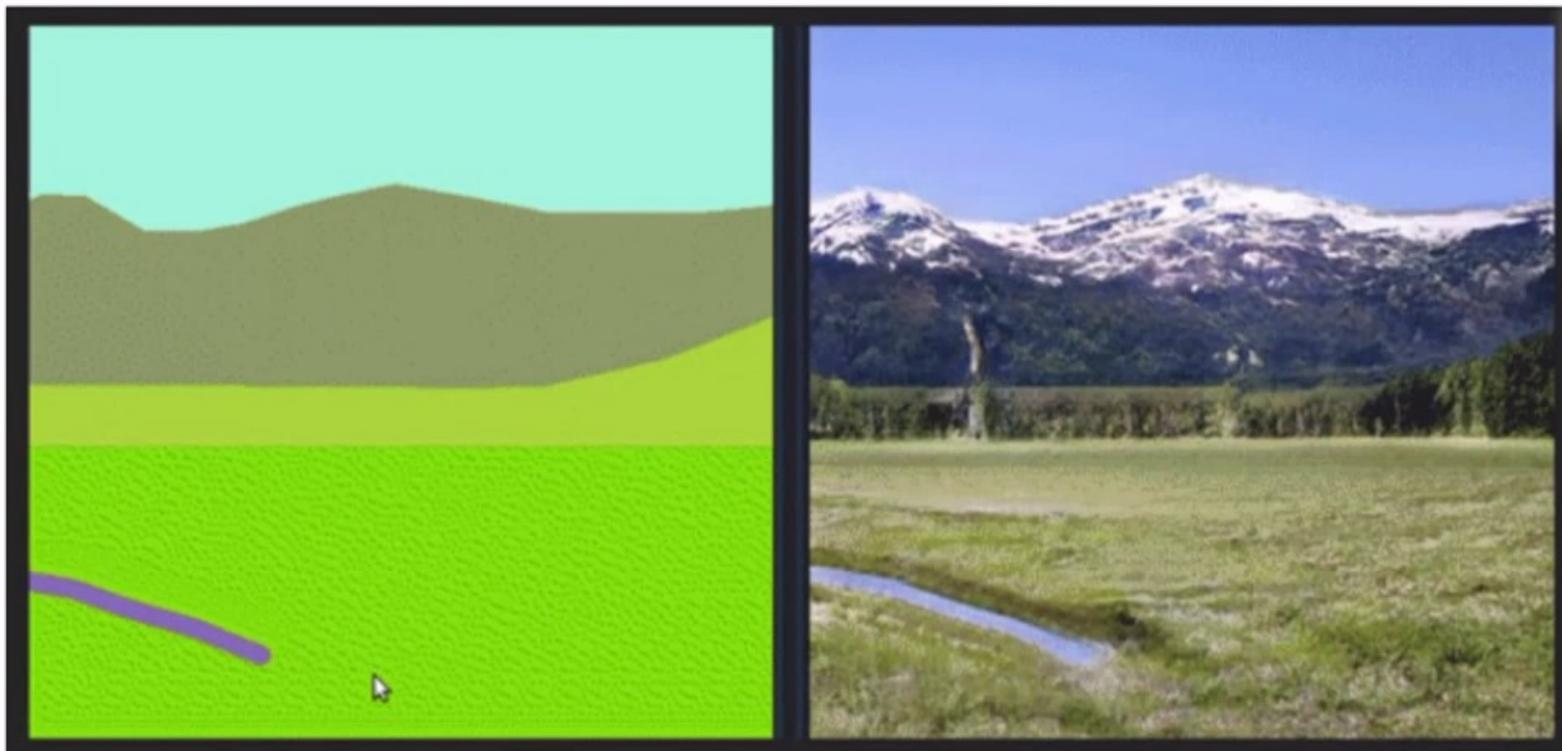


GauGAN

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 for Image Translation



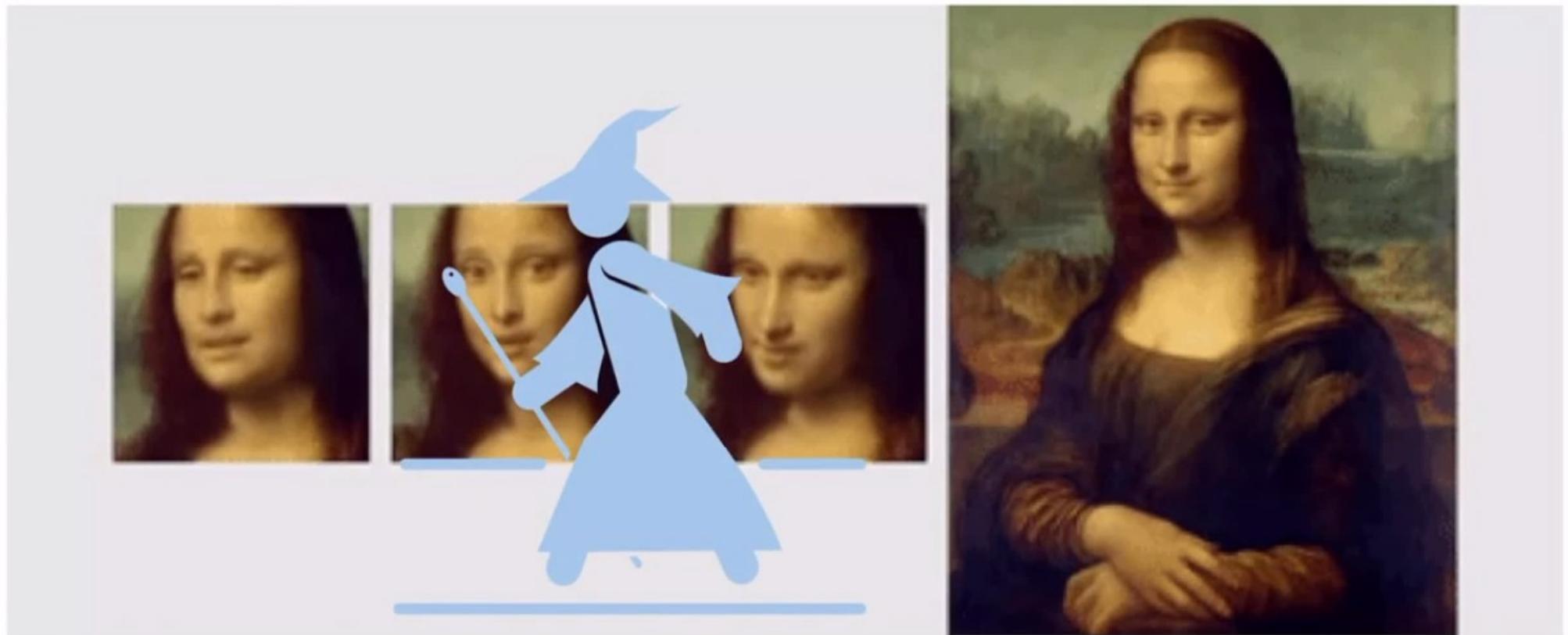
GauGAN

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 can make an Monalisa image animate!!!

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

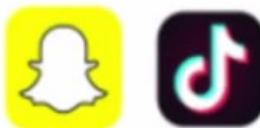


Image Filters



Super-resolution

Summary

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



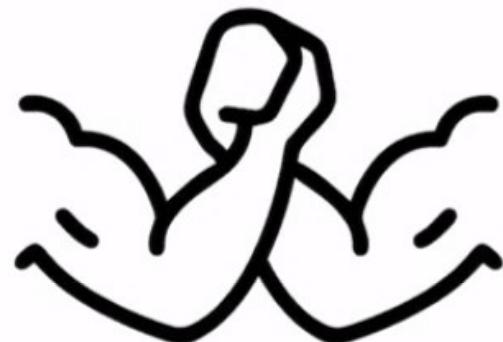


deeplearning.ai

Intuition Behind GANs

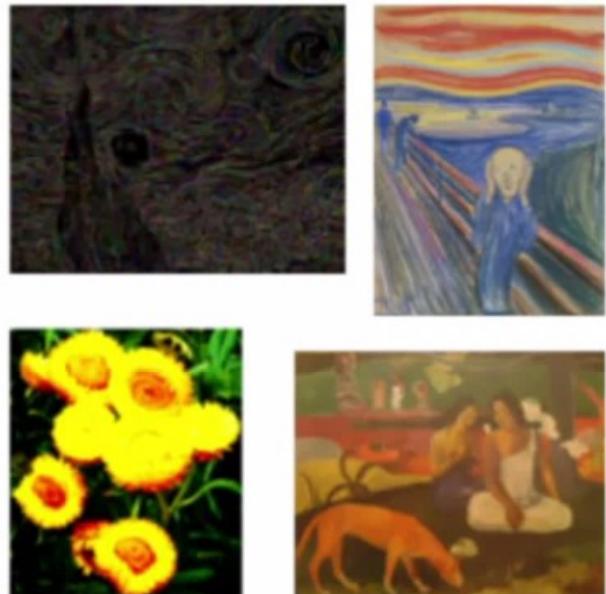
Outline

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

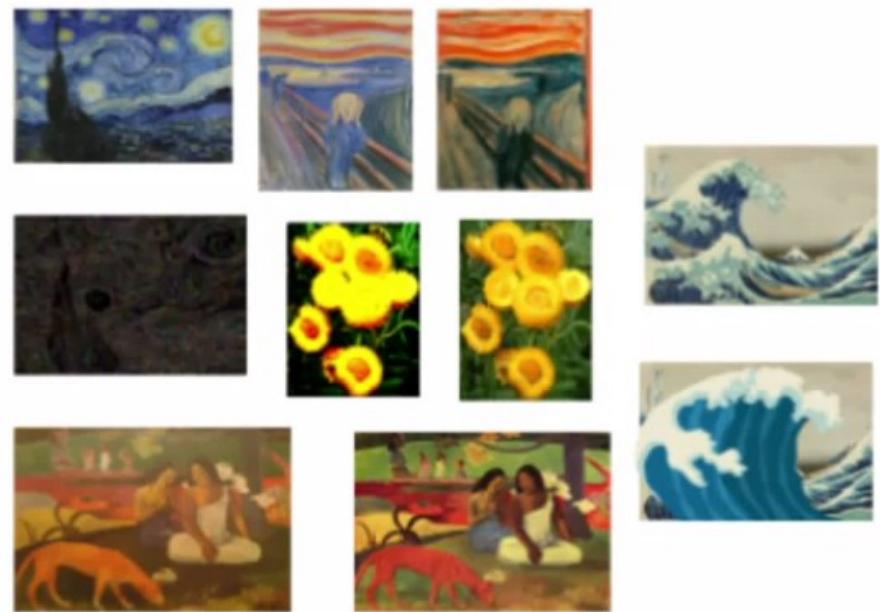


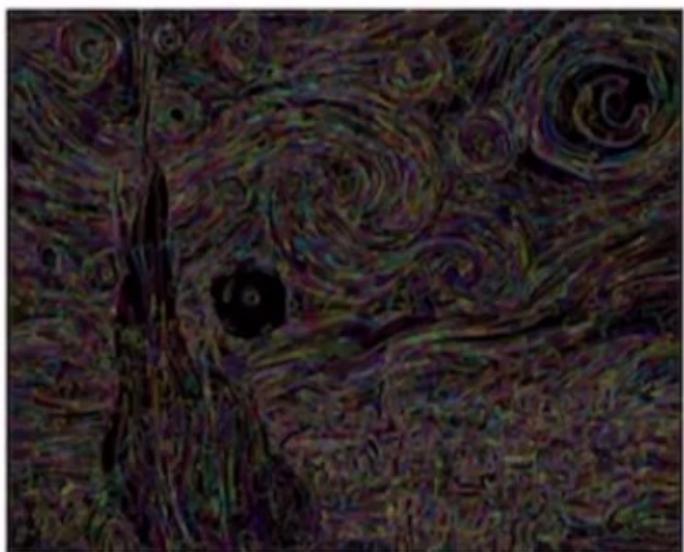
Generative Adversarial Network

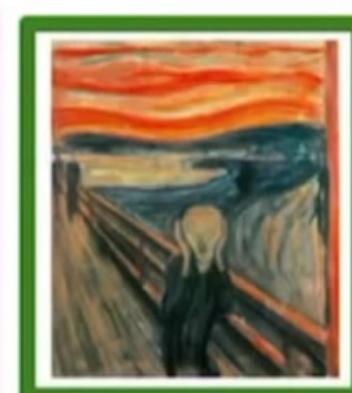
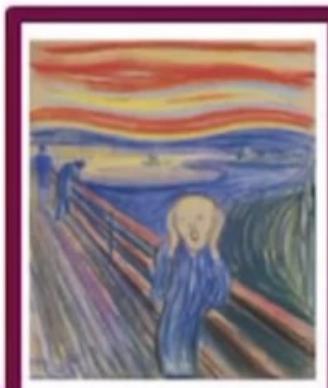
Generator learns to make *fakes* that look *real*



Discriminator learns to distinguish *real* from *fake*







Generative Adversarial Network

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*



- (1) Generator will learn to make more realistic fakes
- (2) Discriminator will make sure to not be fooled by real looking fakes

Generative Adversarial Network

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



Doesn't know how it should look



Generative Adversarial Network



Discriminator learns to distinguish
real from *fake*



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

The Game Is On!



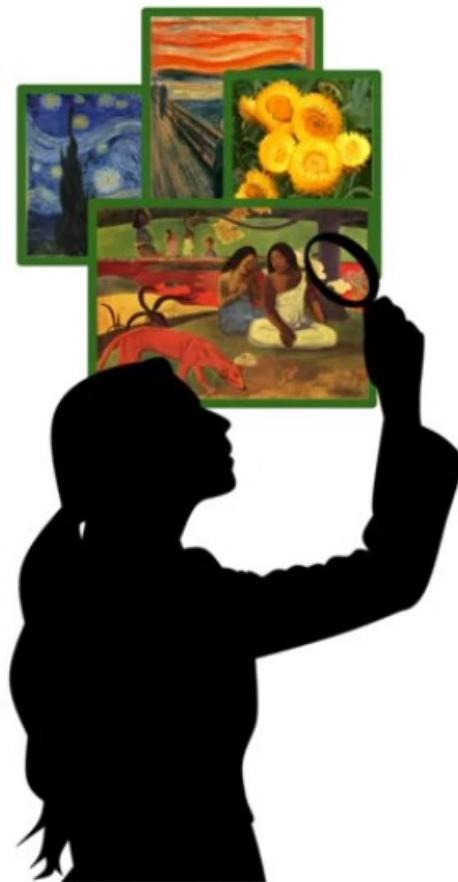
5% Real



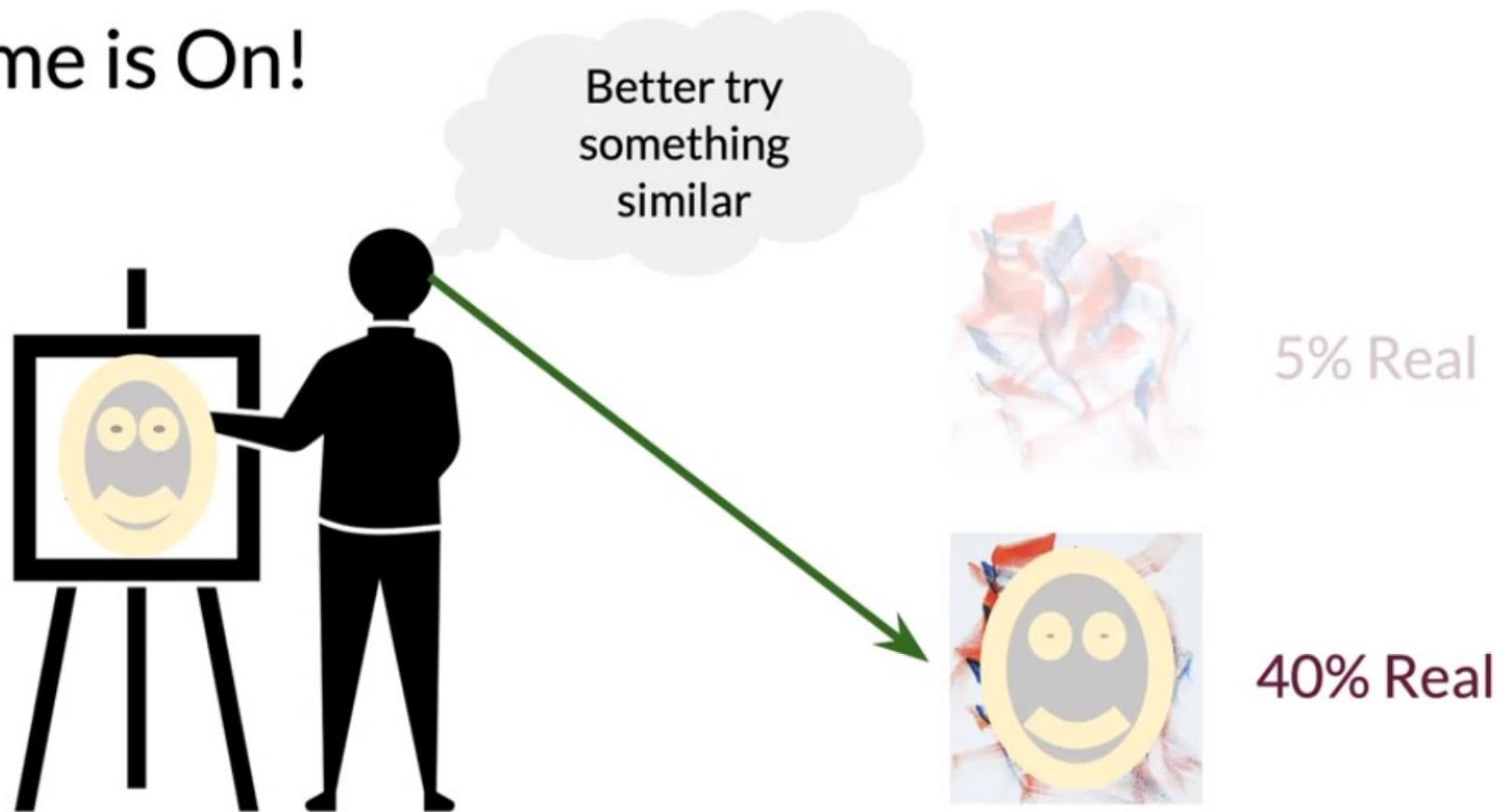
40% Real



80%
Real



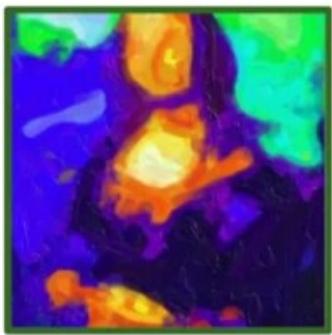
The Game is On!



The Game Is On!



30% Real



60% Real

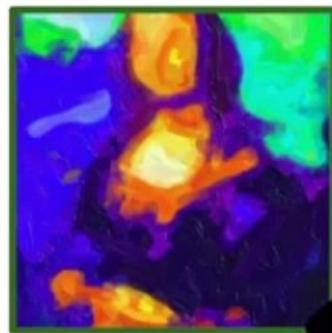


95% Real

The Game Is On!

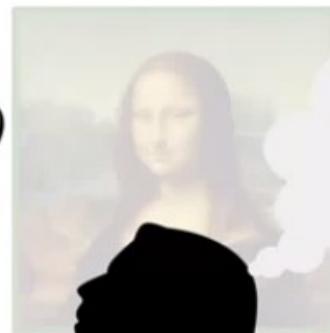


30% Real



60% Real

Wrong!



95% Real

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

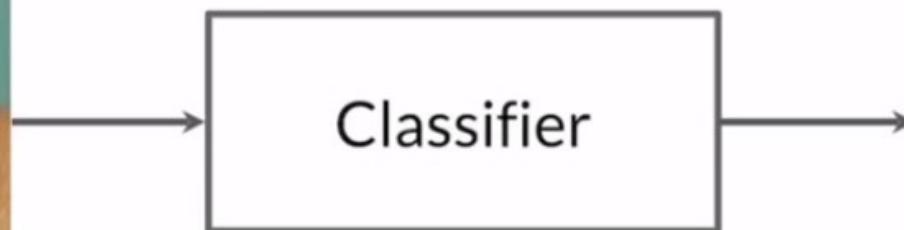
Outline

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



Classifiers

Distinguish between different classes



Turtle

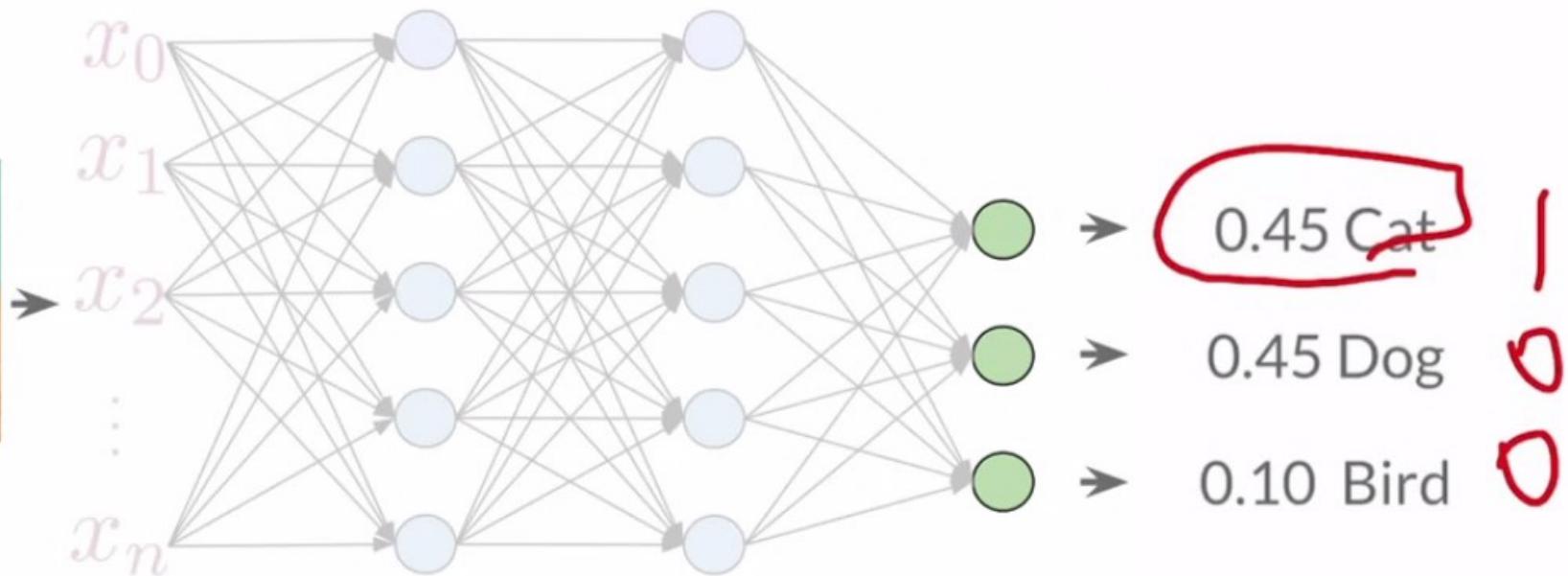
Bird

Cat

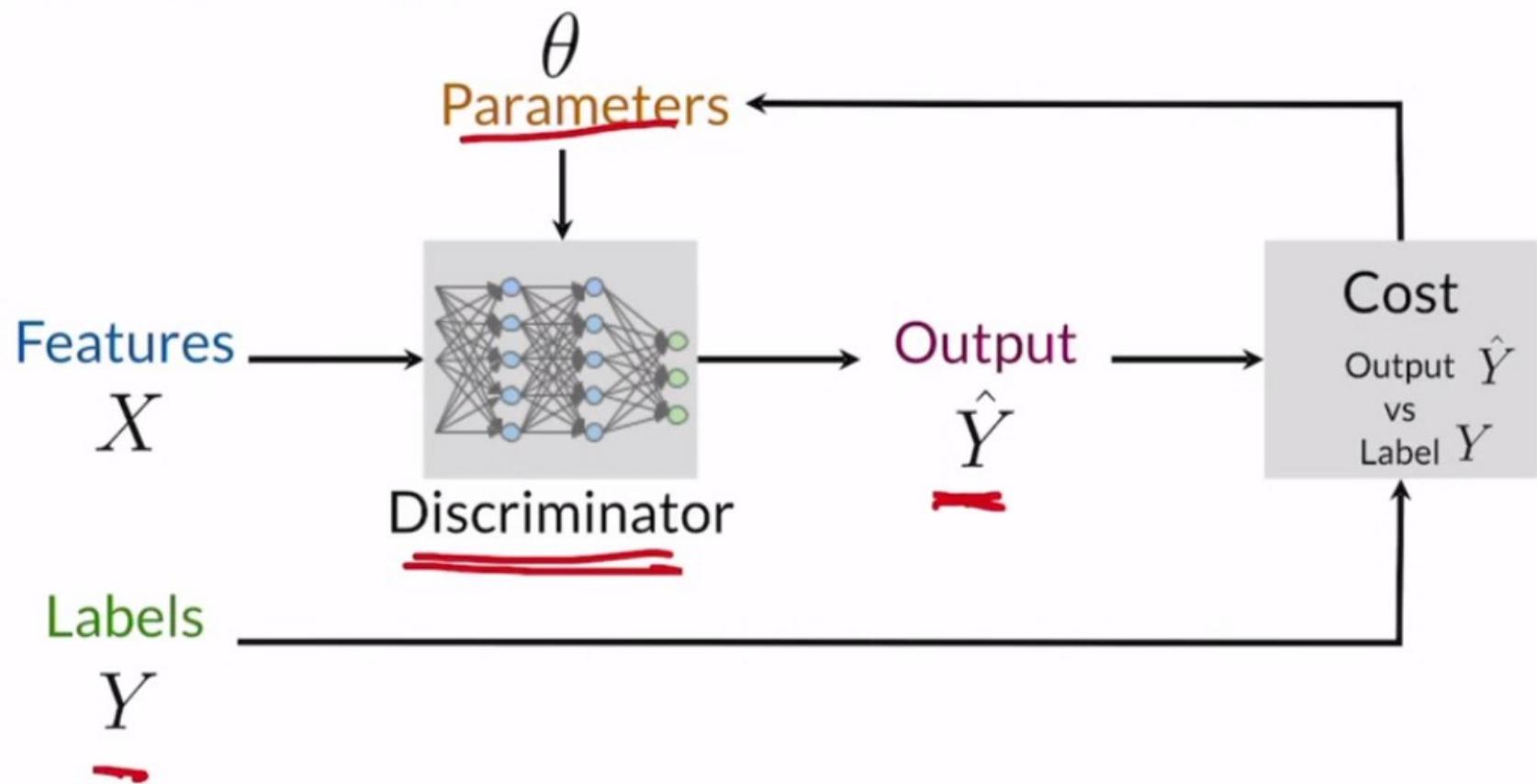
Dog

Fish

Neural Networks



Classifiers (training)

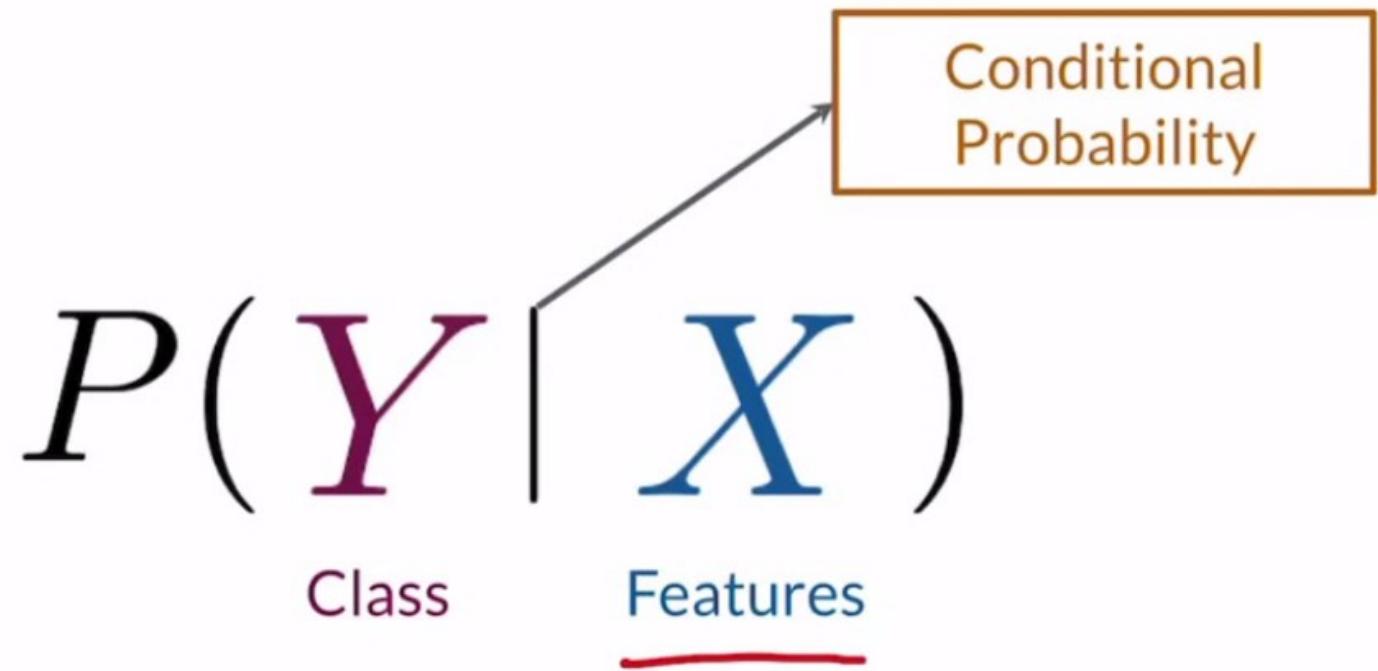


Classifiers

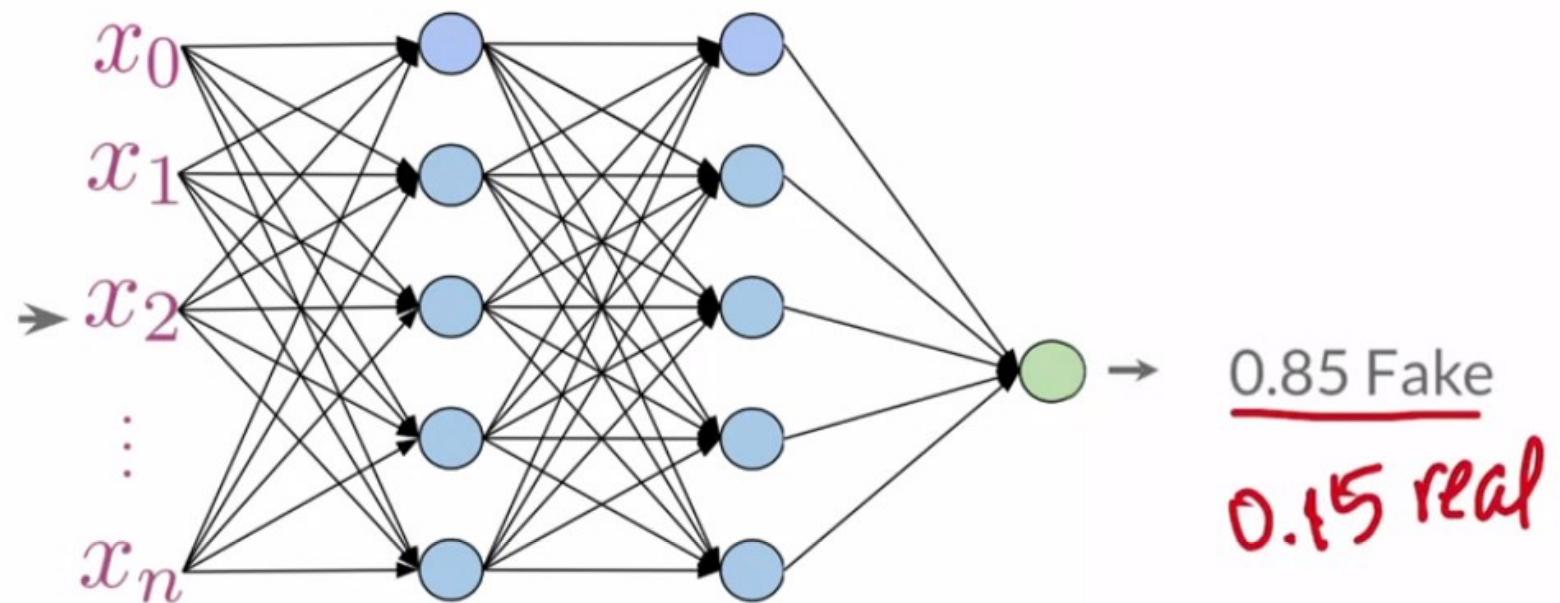
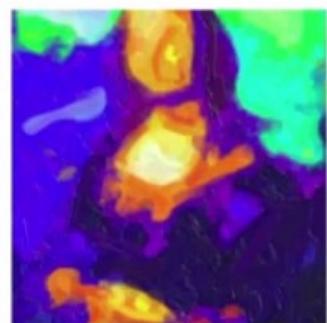
$$P(Y \mid \underline{X})$$

Class Features

Conditional Probability



Discriminator



Discriminator

$$P(\text{Fake} \mid \begin{matrix} \text{Class} \\ | \\ \text{Features} \end{matrix}) = 0.85 \rightarrow \boxed{\text{Fake}}$$

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



Generator

Turtle

Generates examples of the class

Bird

Cat

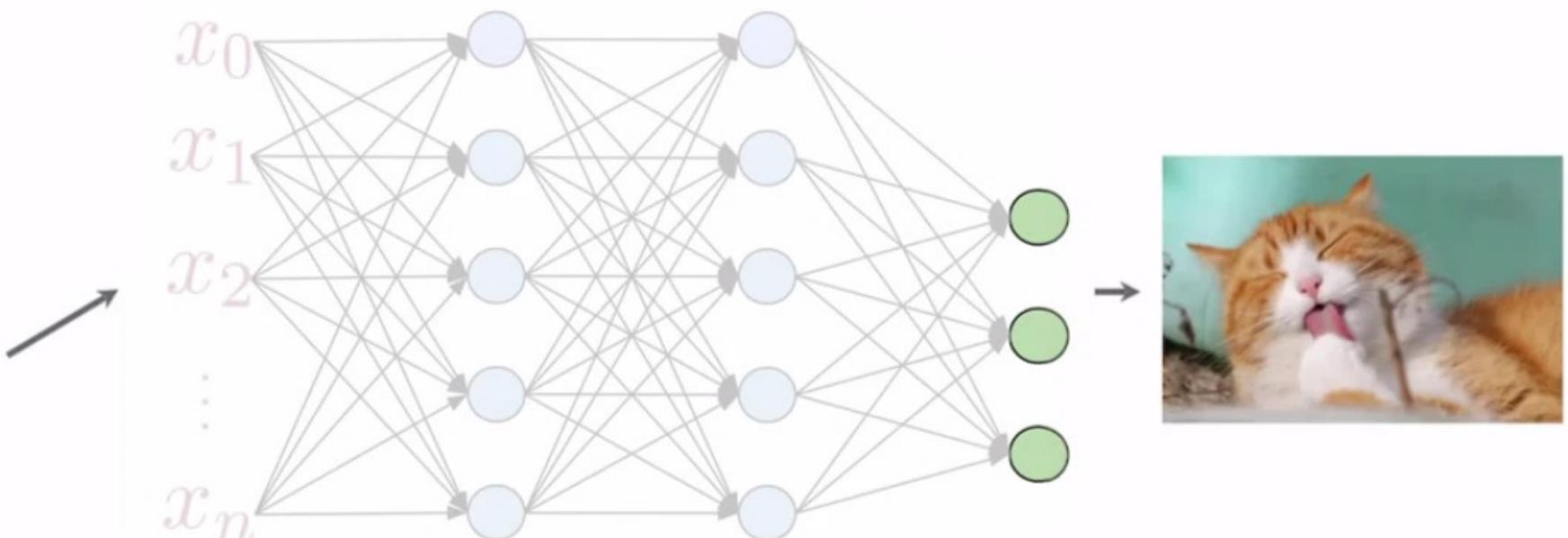
Dog

Fish



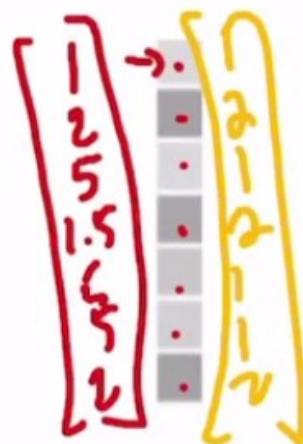
Neural Networks

$$\begin{bmatrix} 1 \\ 2 \\ 5 \\ 1.5 \\ \vdots \\ 2 \end{bmatrix} \rightarrow \dots$$

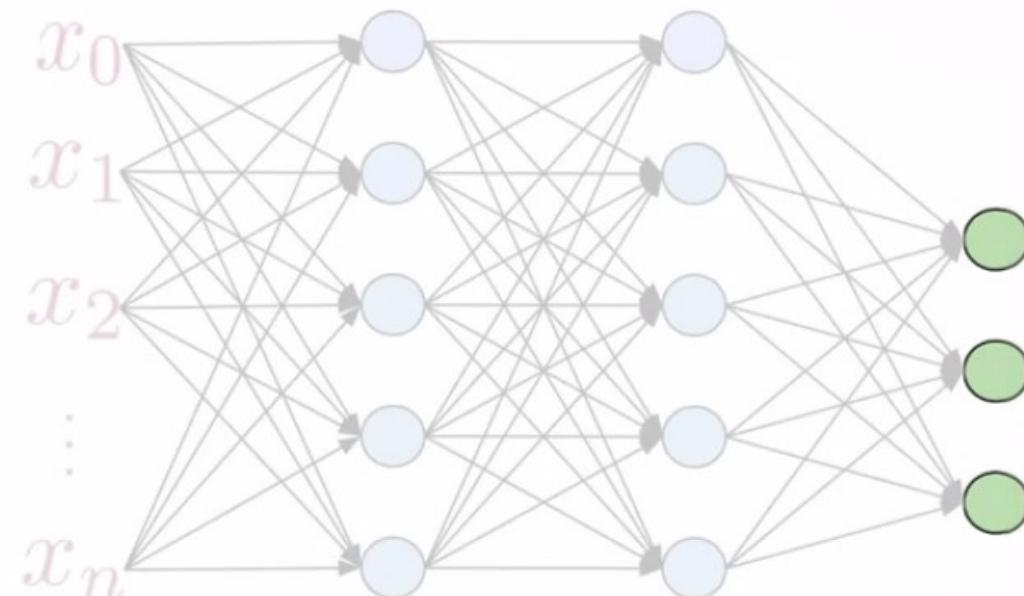


Noise
vector

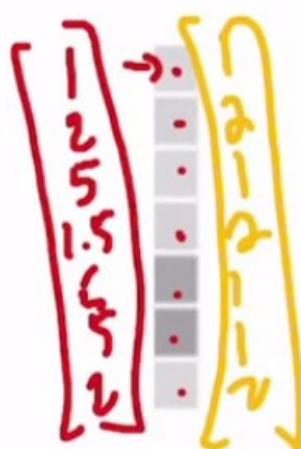
Neural Networks

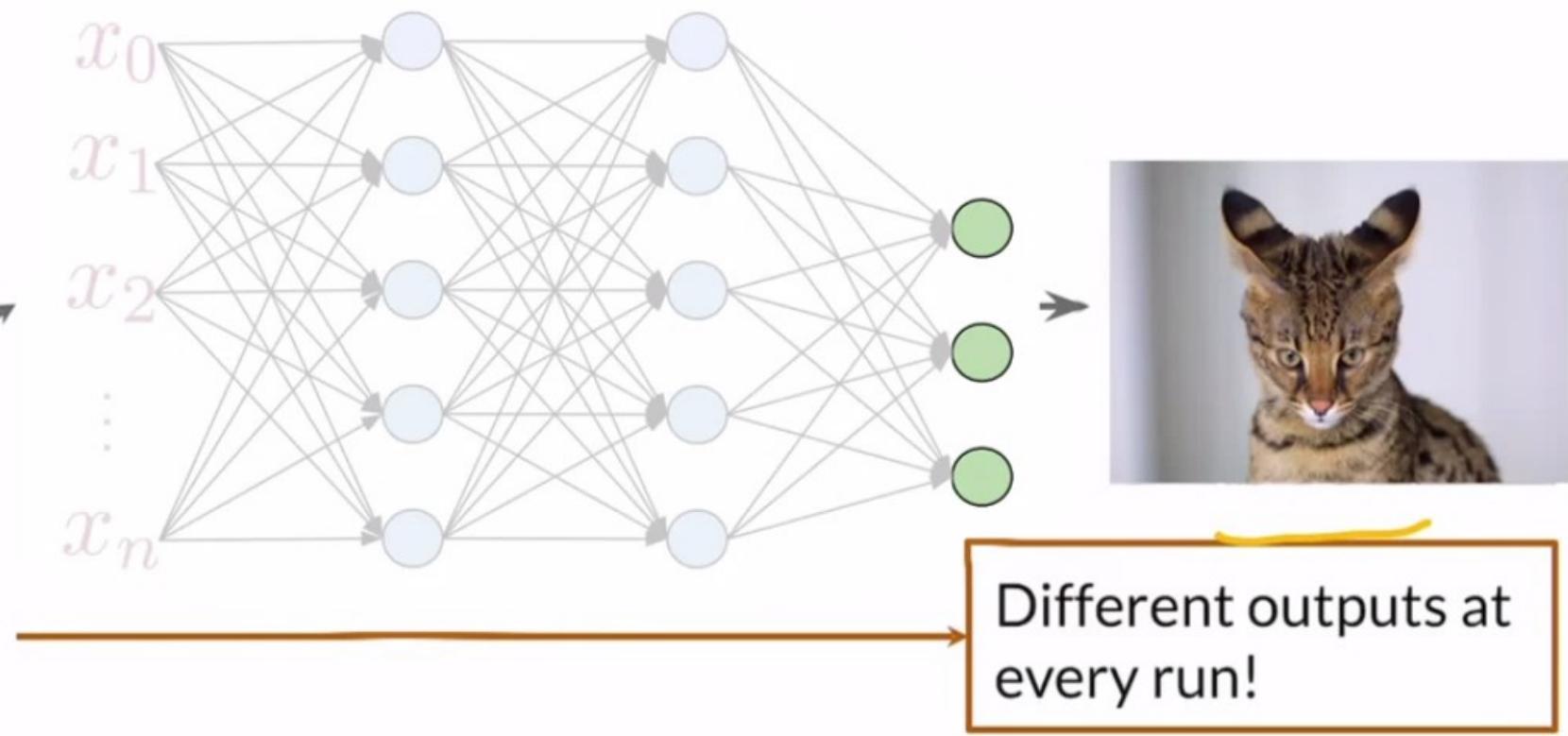


Noise
vector

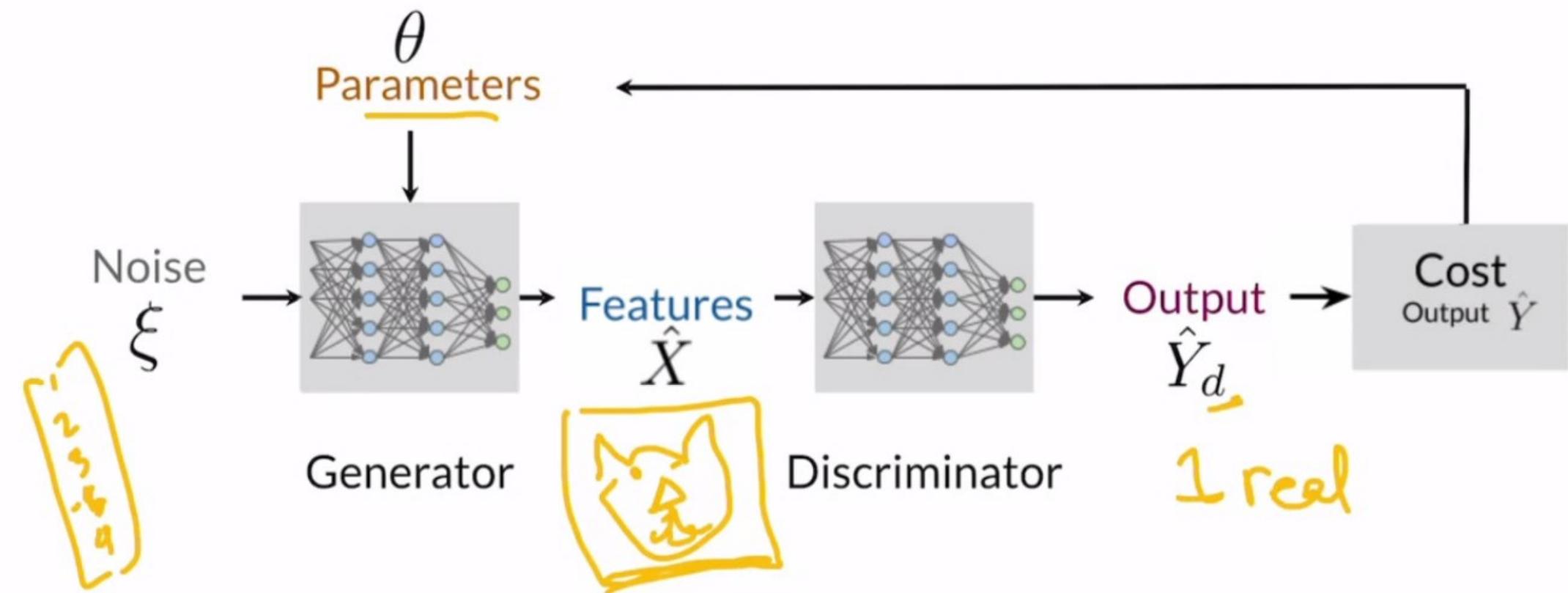


Neural Networks

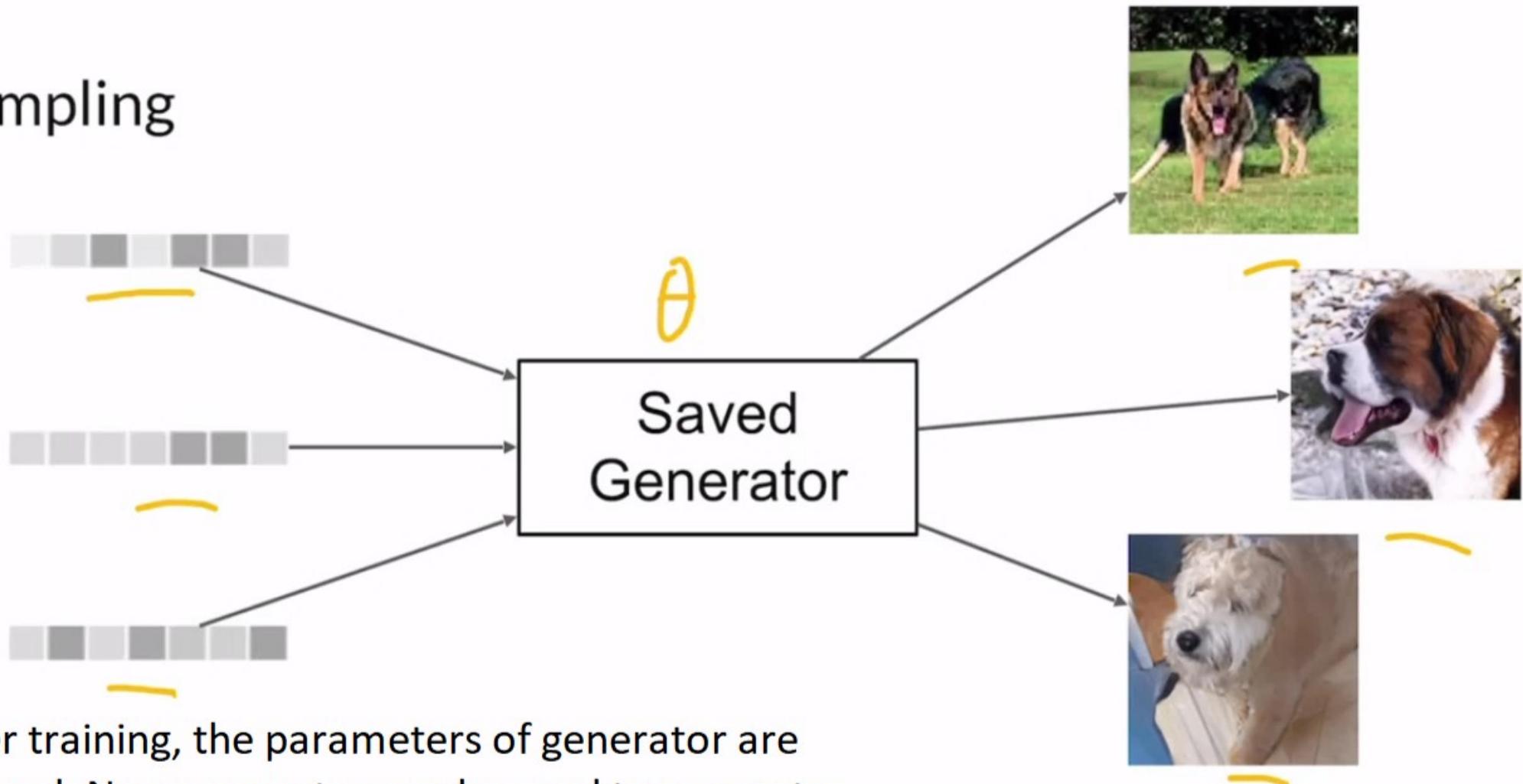

Noise vector



Generator: Learning

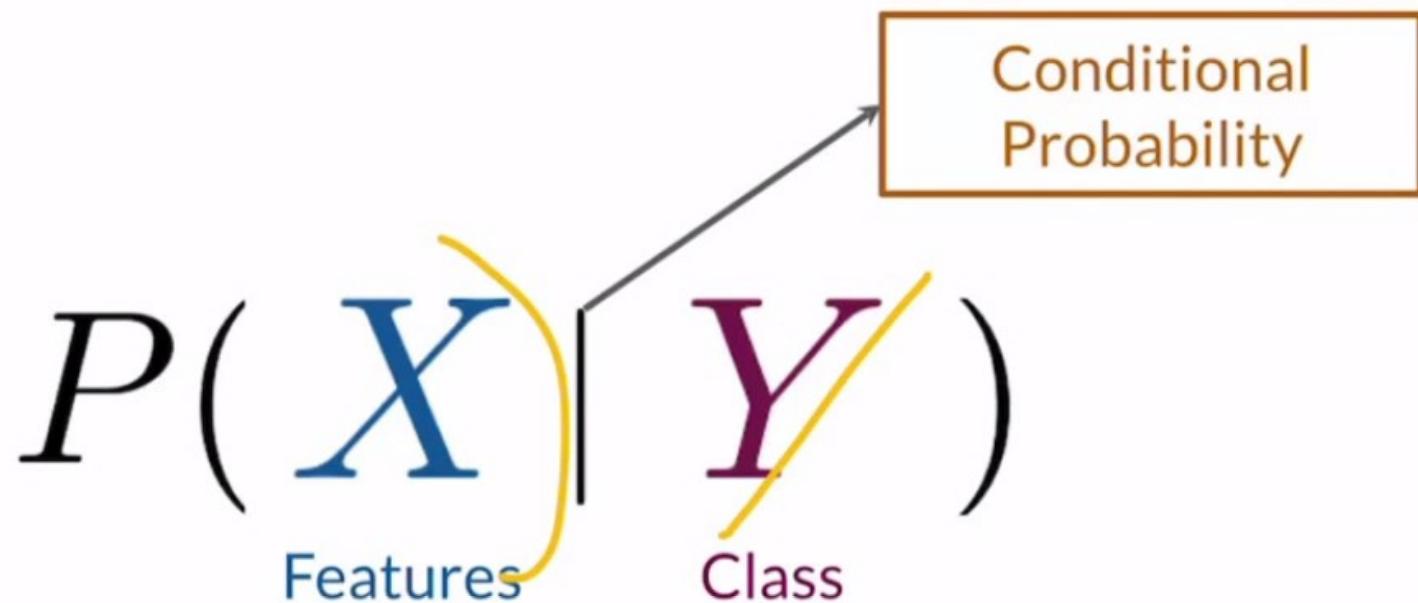


Sampling



After training, the parameters of generator are freezed. Now generator can be used to generate realistic looking fake images

Generator



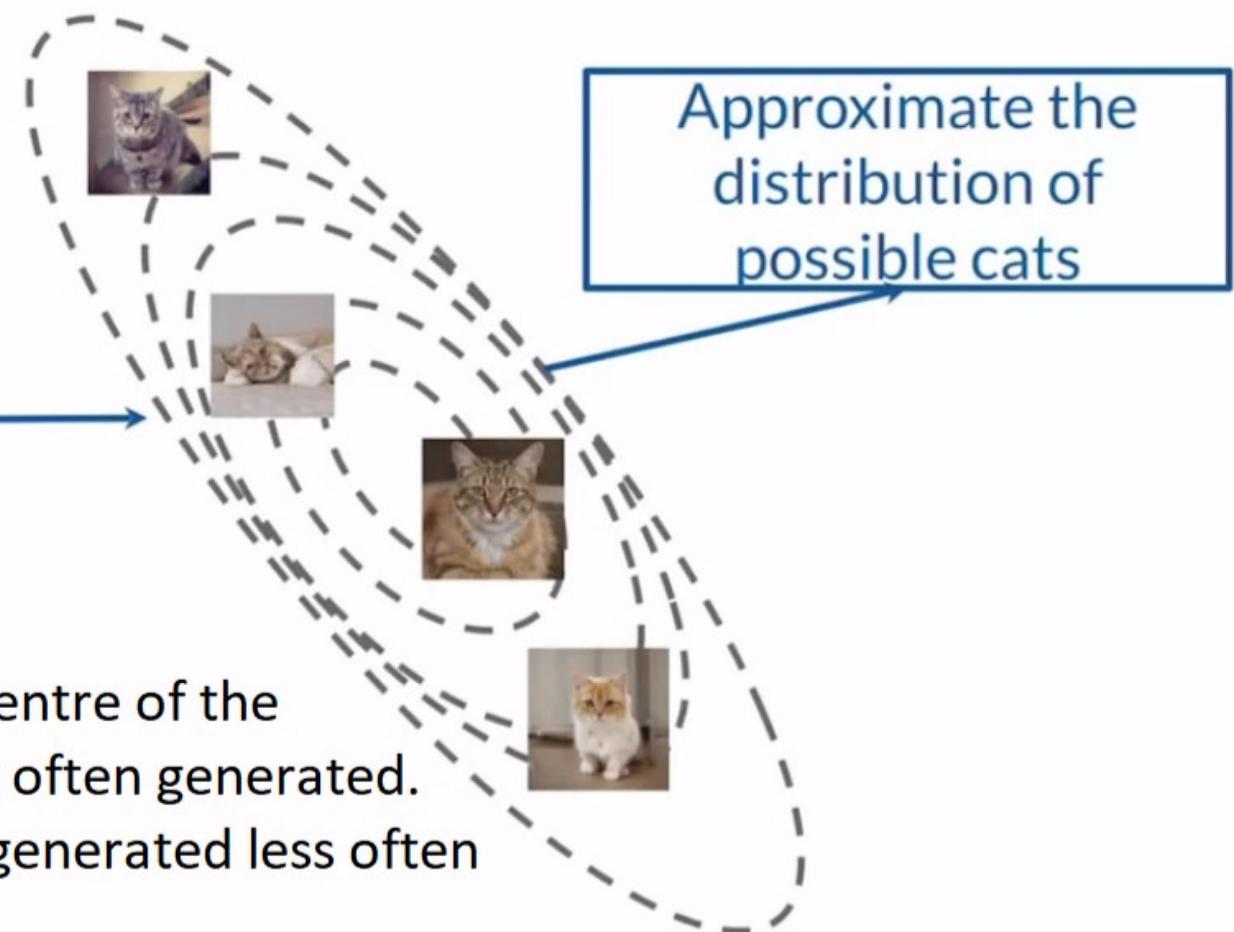
We do not need Y (class) if there is only 1 class to generate. But if there are more than 2 classes (H and NH), then we need to use conditional probability

Generator

$$P(X)$$

Features

Common breeds of cats will be at centre of the distribution and hence will be more often generated.
Whereas rare breeds of cat will be generated less often



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

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



BCE Cost Function

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

Average loss of the whole batch

The diagram illustrates the components of the BCE Cost Function. A purple circle highlights the first term of the summation: $\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log h(x^{(i)}, \theta)]$. A grey arrow points from this term to a box labeled "Average loss of the whole batch". Another grey arrow points from the term $y^{(i)} \log h(x^{(i)}, \theta)$ to the label "Prediction". A yellow arrow points from the term $(1 - y^{(i)}) \log(1 - h(x^{(i)}, \theta))$ to the label "Label". A yellow arrow points from the term $h(x^{(i)}, \theta)$ to the label "Features". A vertical grey arrow points from the term $h(x^{(i)}, \theta)$ to the label "Parameters".

BCE Cost Function

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

y⁽ⁱ⁾ log *h(x⁽ⁱ⁾; θ)*

<i>y⁽ⁱ⁾</i>	<i>h(x⁽ⁱ⁾, θ)</i>	<i>y⁽ⁱ⁾ log h(x⁽ⁱ⁾, θ)</i>
0	any	0
1	0.99	~0
1	~0	-inf

real → *fake*

Relevant when the label is 1

BCE Cost Function

$$(1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

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

$$-(-\infty) = +\infty$$

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

Relevant when
the label is 0

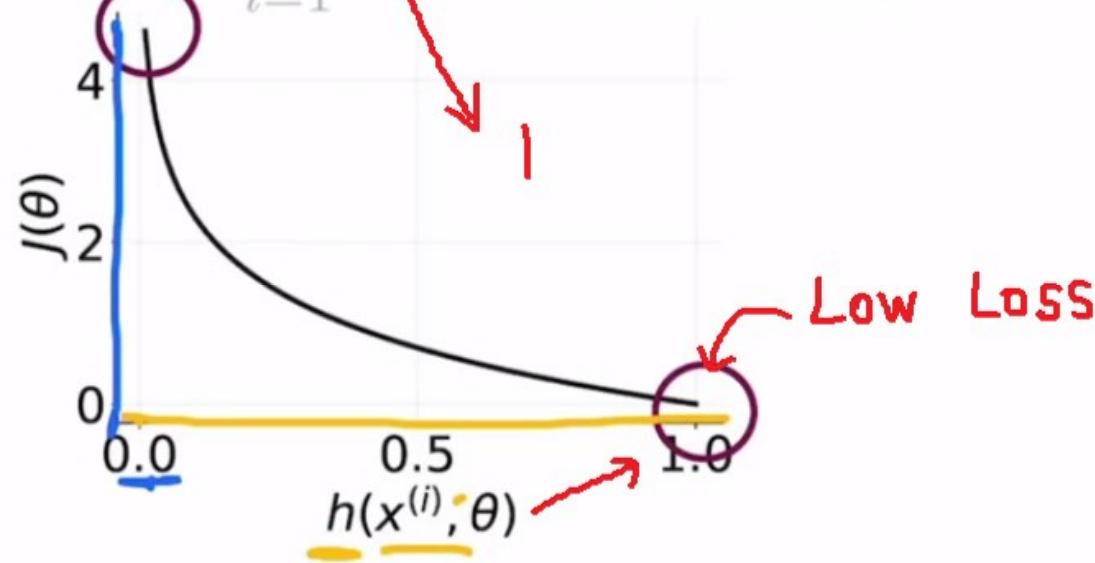
BCE Cost Function

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

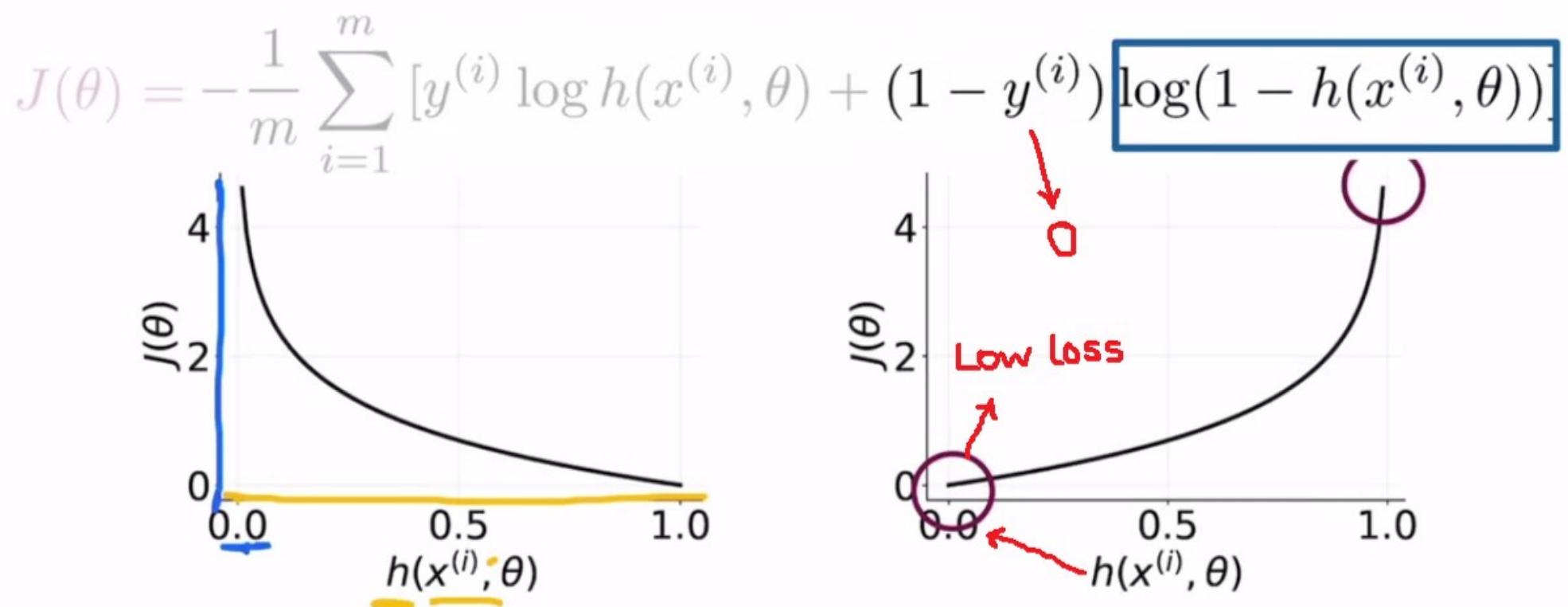
Ensures that the cost is always greater or equal to 0

BCE Cost Function

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



BCE Cost Function



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

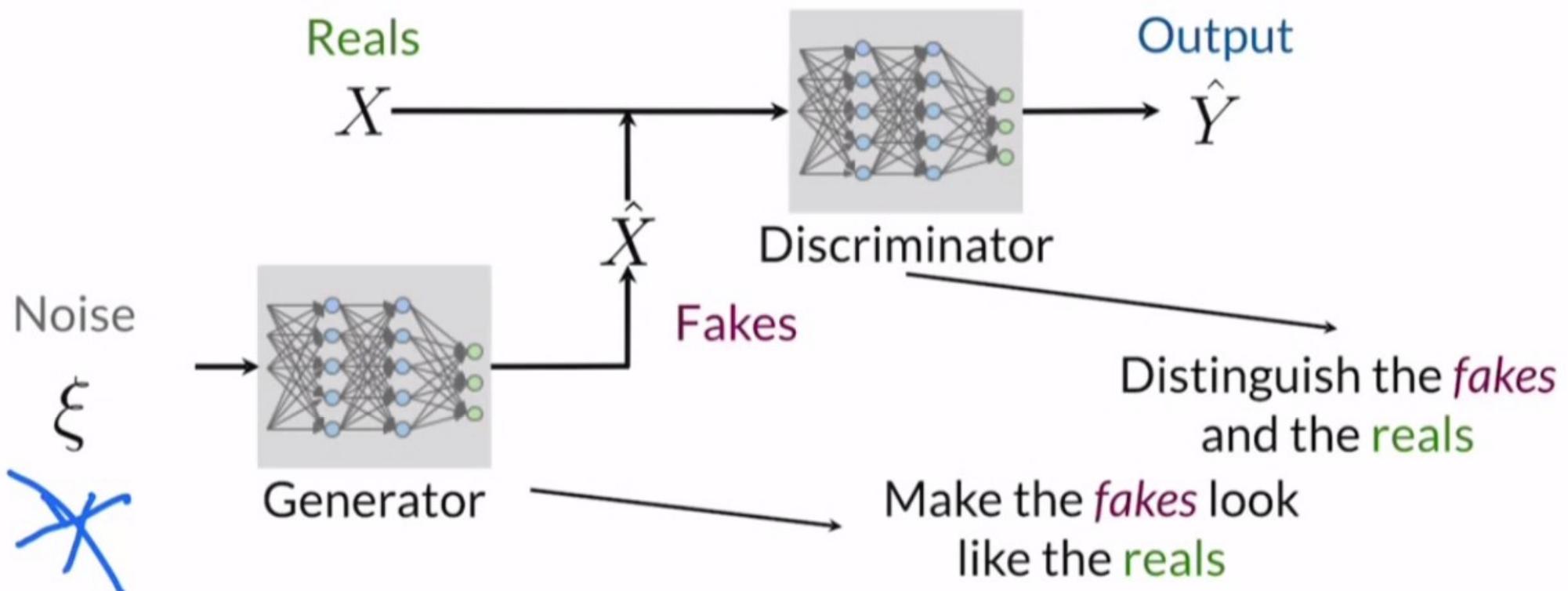
Putting It All Together

Outline

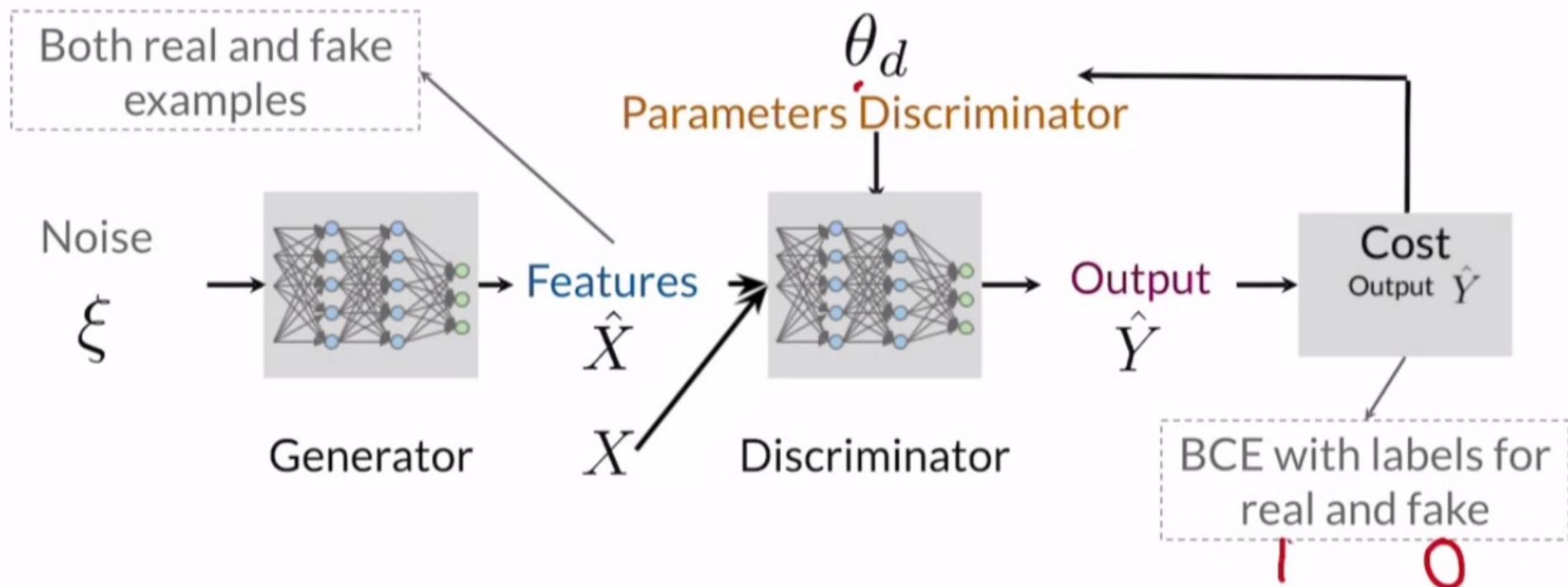
- How the whole architecture looks
- How to train GANs



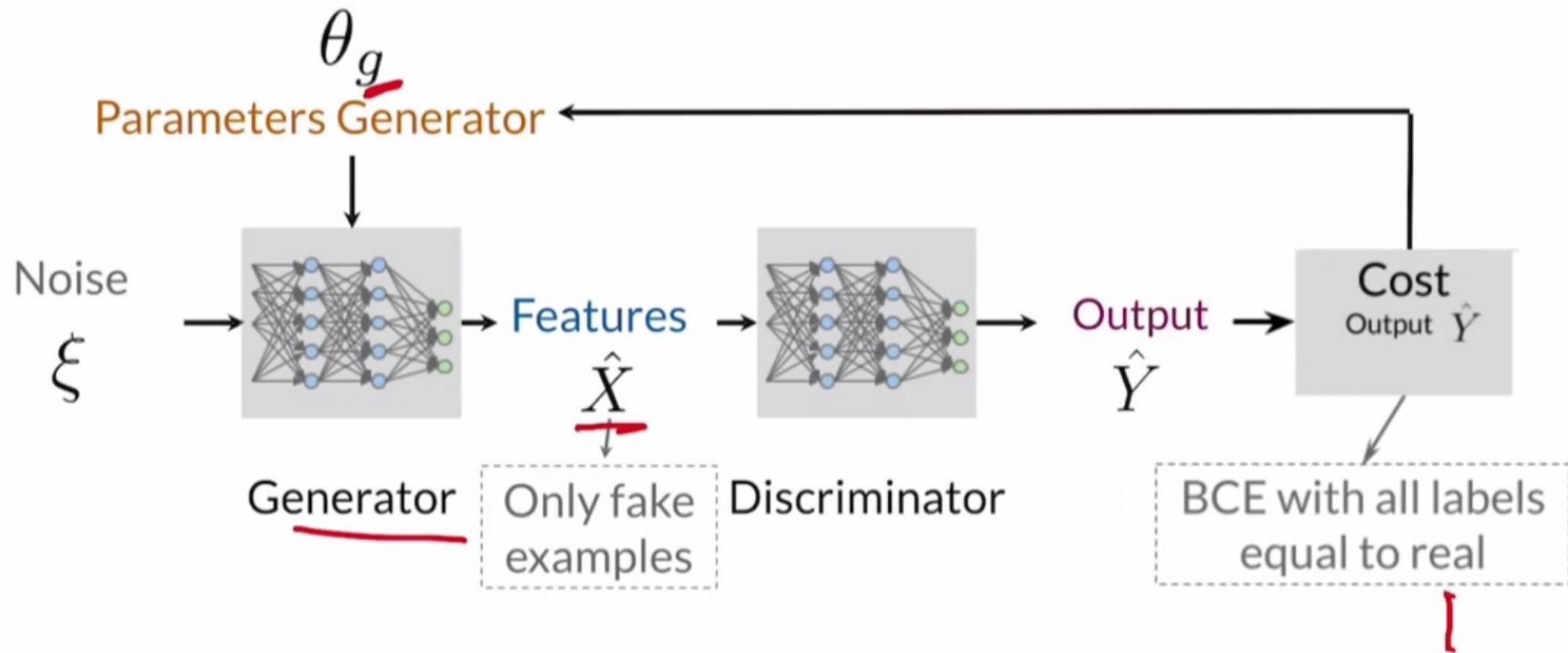
GANs Model



Training GANs: Discriminator



Training GANs: Generator



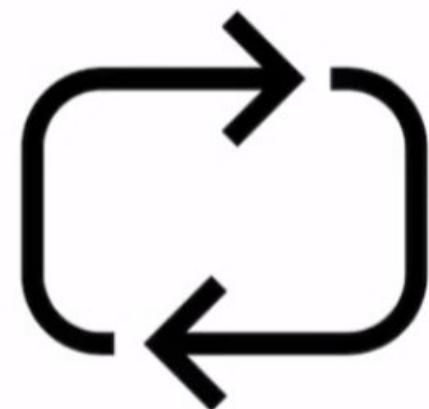
Training GANs



- (1) If we have superior discriminator, generator will have no clue in which direction to improve and update weights.
- (2) Discriminator task is easier than generator (classification vs generate distribution of features).
- (3) Thus both discriminator and generator must be at same scale and learn together.

Summary

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



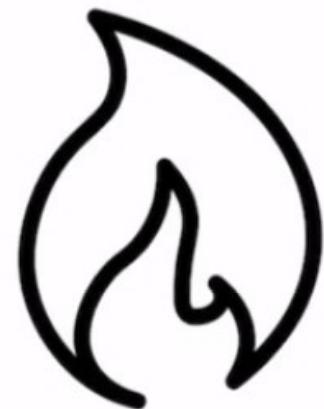


deeplearning.ai

Intro to PyTorch (Optional)

Outline

- Comparison with TensorFlow
- Defining Models
- Training



PyTorch vs TensorFlow

PyTorch	TensorFlow
<p>Imperative, computations on the go</p> <pre>A, B = 1, 2 C = A + B print(C)</pre> <p>3</p> <p>Dynamic Computational Graphs</p>	<p>Symbolic, first define and then compile</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):  
    y_pred = model(x)                                         Training loop for number of epochs  
    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!

