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

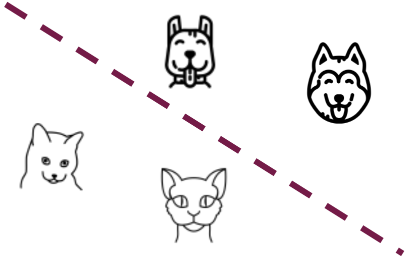
Generative Models

Outline

- What are generative models
- Types of generative models

Generative Models vs. Discriminative Models

Discriminative models



Features Class

$$X \rightarrow Y$$

$$P(Y|X)$$

Generative models

Noise Class Features

$$\xi, Y \rightarrow X$$

$$P(X|Y)$$

Generative Models vs. Discriminative Models

Generative models

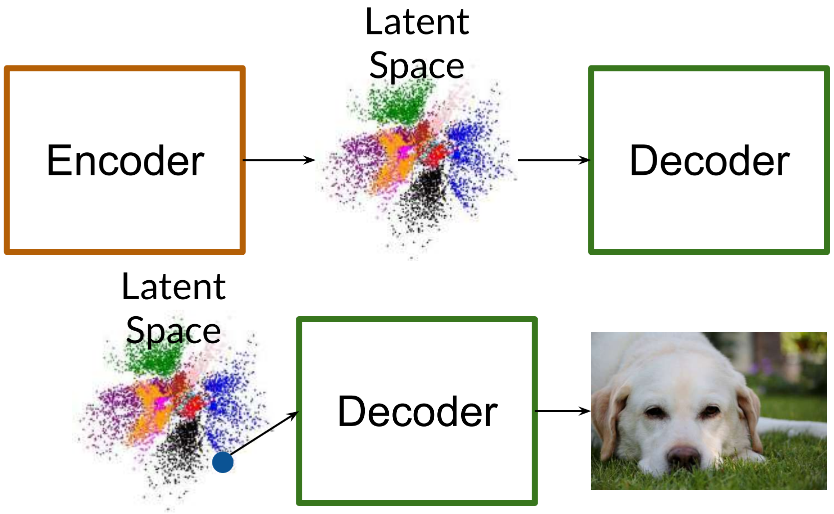
Noise Class Features

$$\xi, Y \rightarrow X$$

$$P(X|Y)$$

Generative Models

Variational Autoencoders



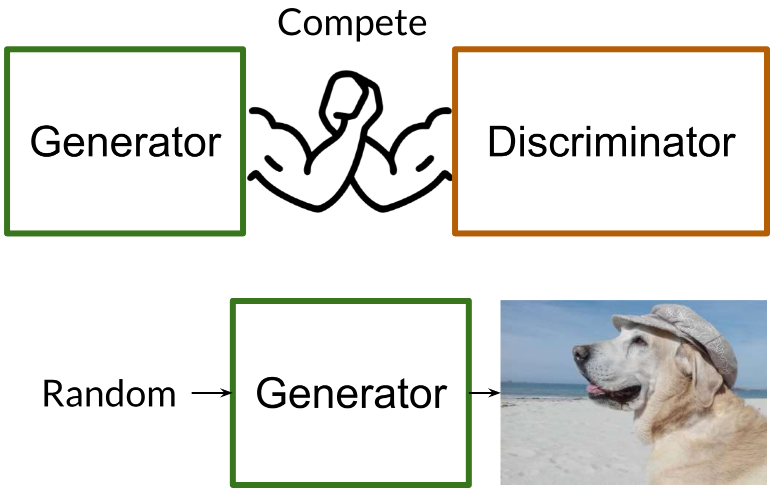
Generative Adversarial Networks

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

Generative Models

Variational Autoencoders

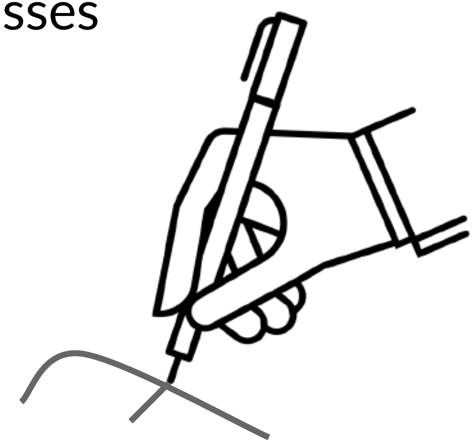
Generative Adversarial Networks



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

Summary

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

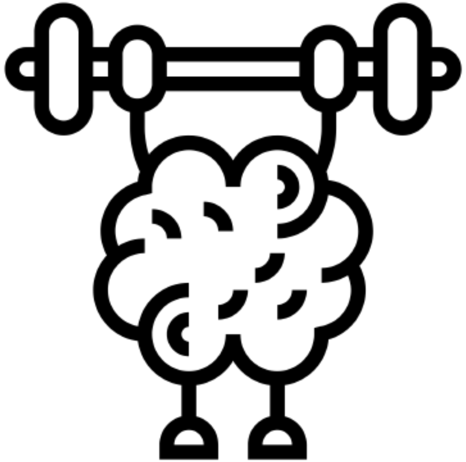


deeplearning.ai

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

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

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

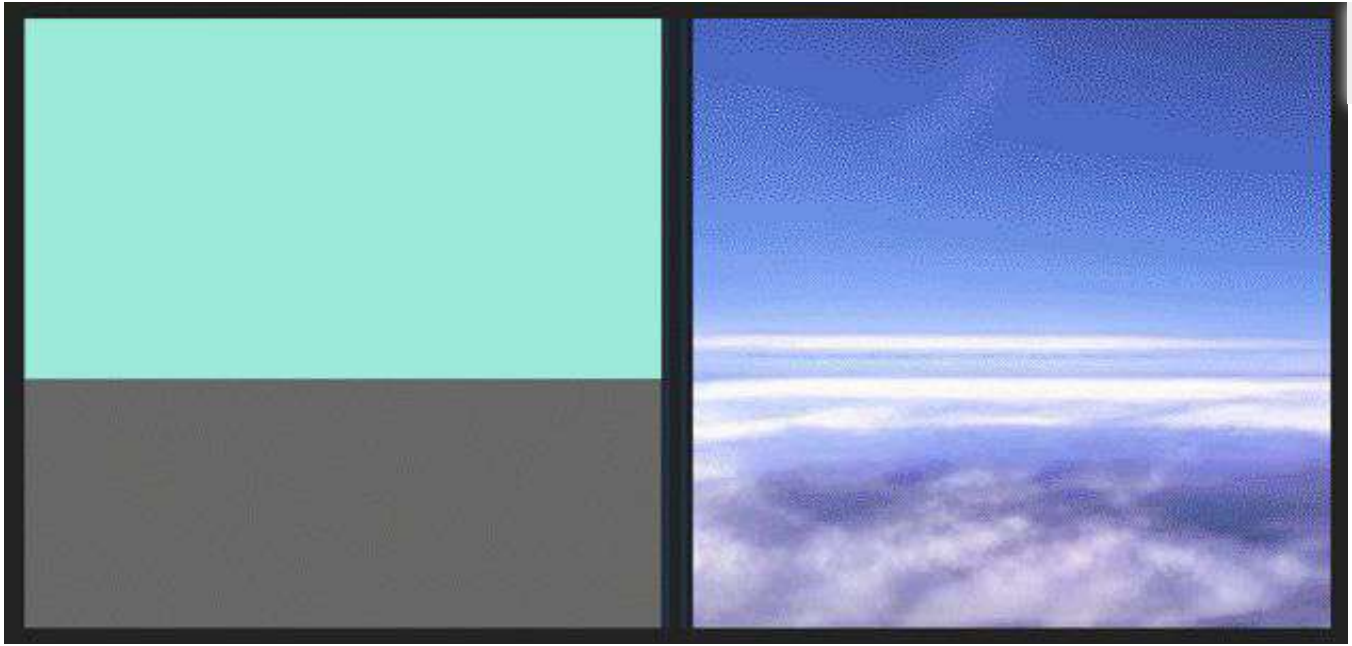
GANs for Image Translation

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

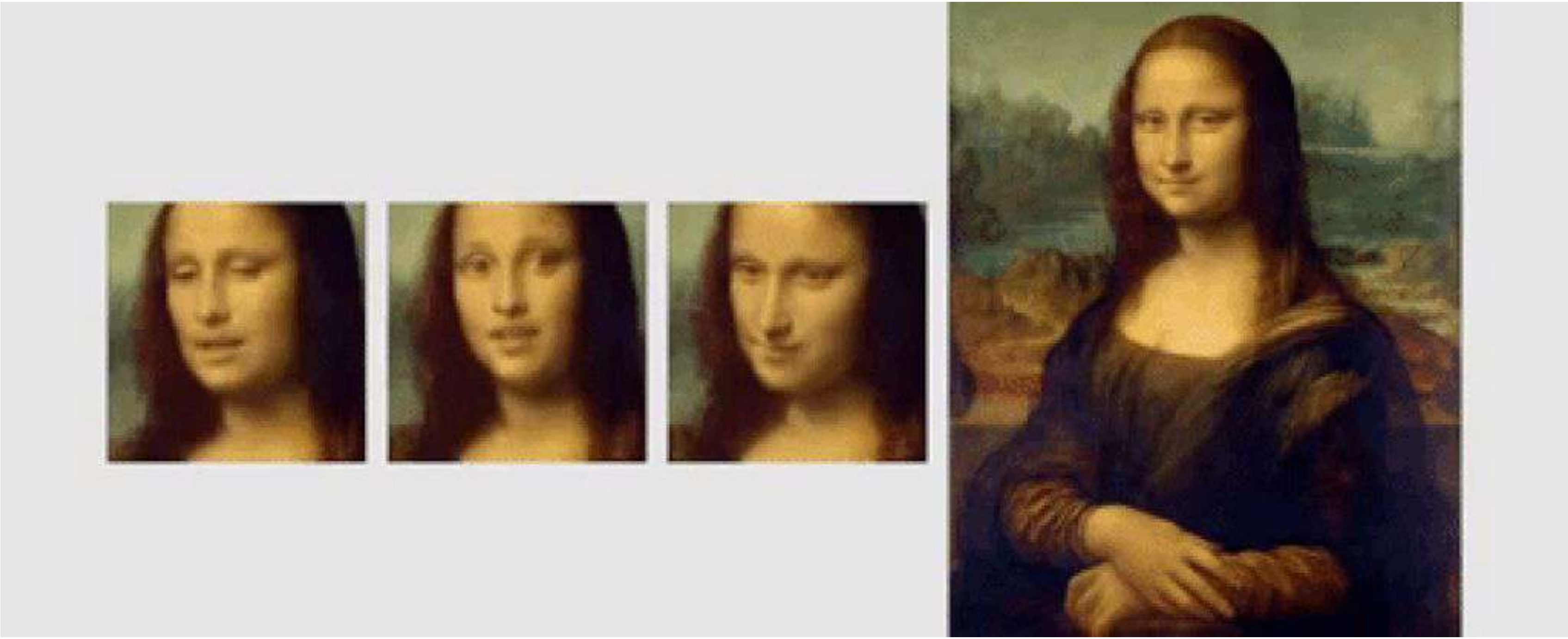
Doodles

↓

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

Adobe — Next-gen Photoshop

Google — Text Generation

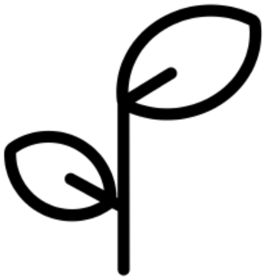
IBM — Data Augmentation

Snapchat, TikTok — Image Filters

Disney — Super-resolution

Summary

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



deeplearning.ai

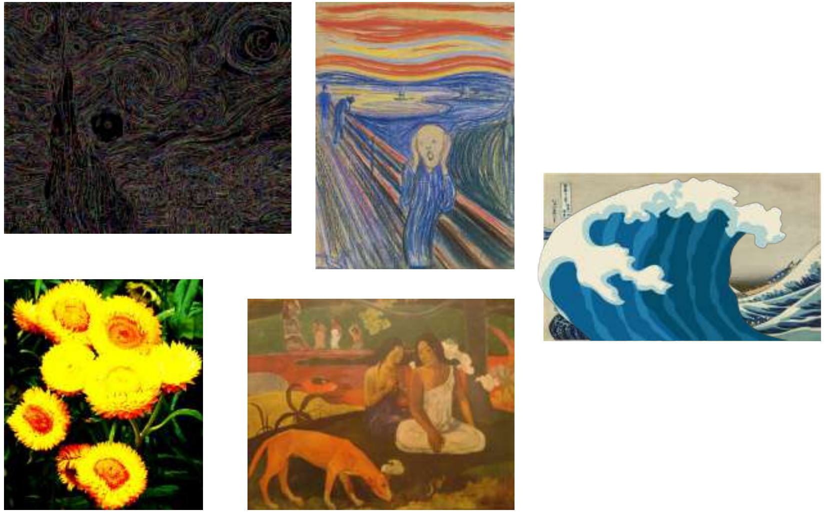
Intuition Behind GANs

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

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*

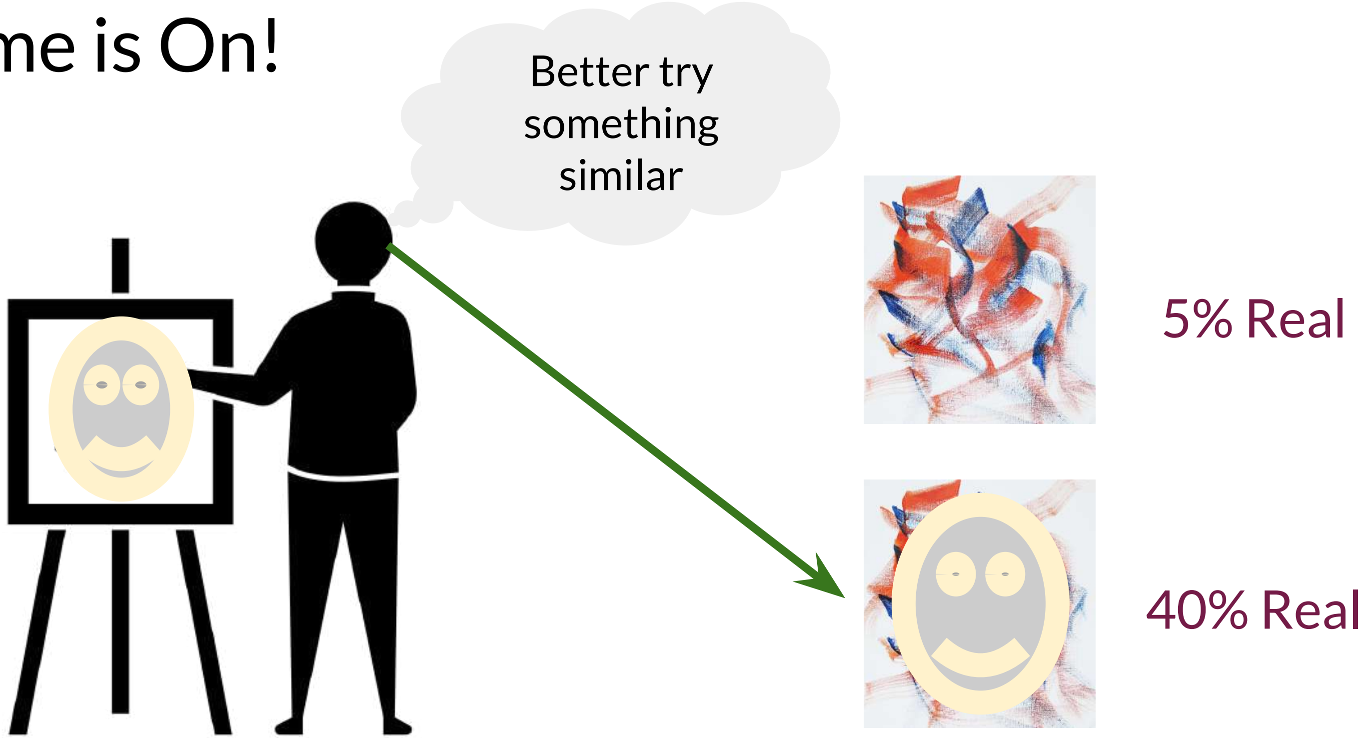
The Game Is On!

5% Real

40% Real

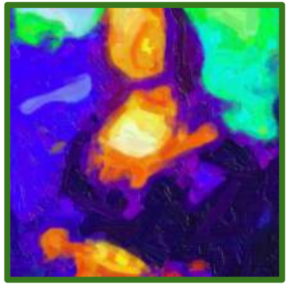
80% Real

The Game is On!



The Game Is On!

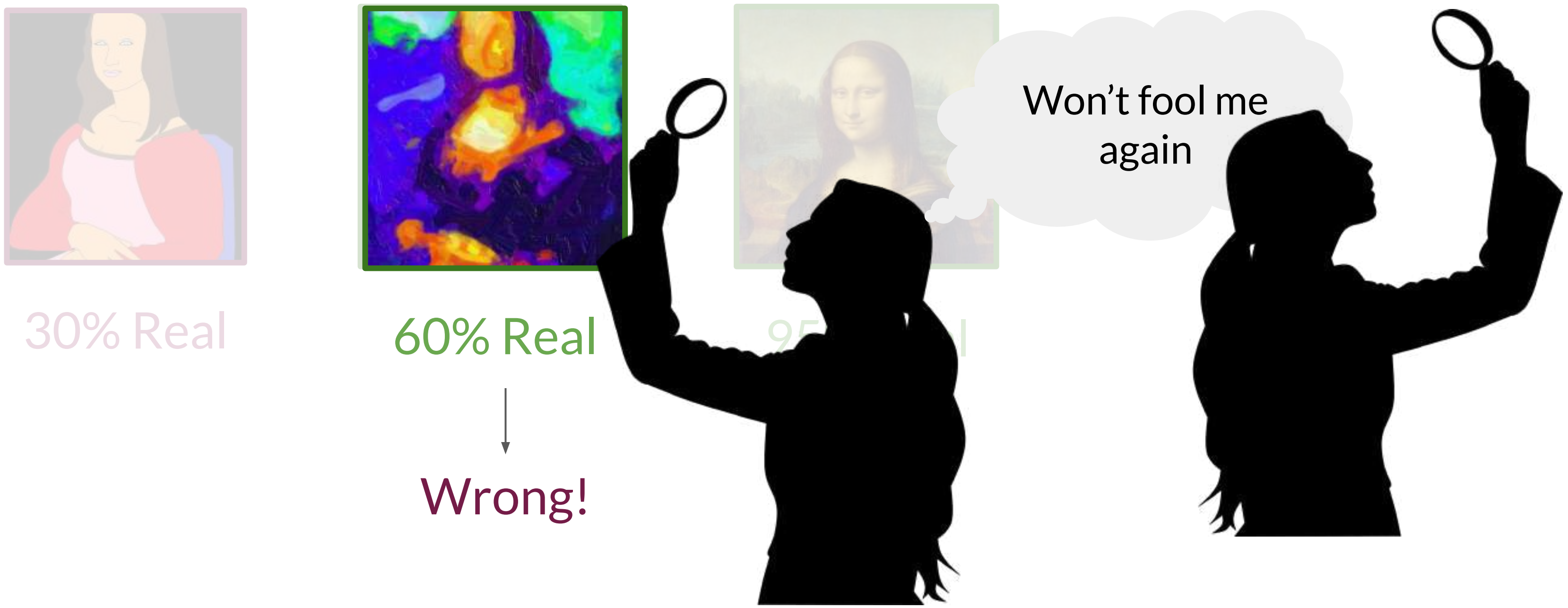
30% Real



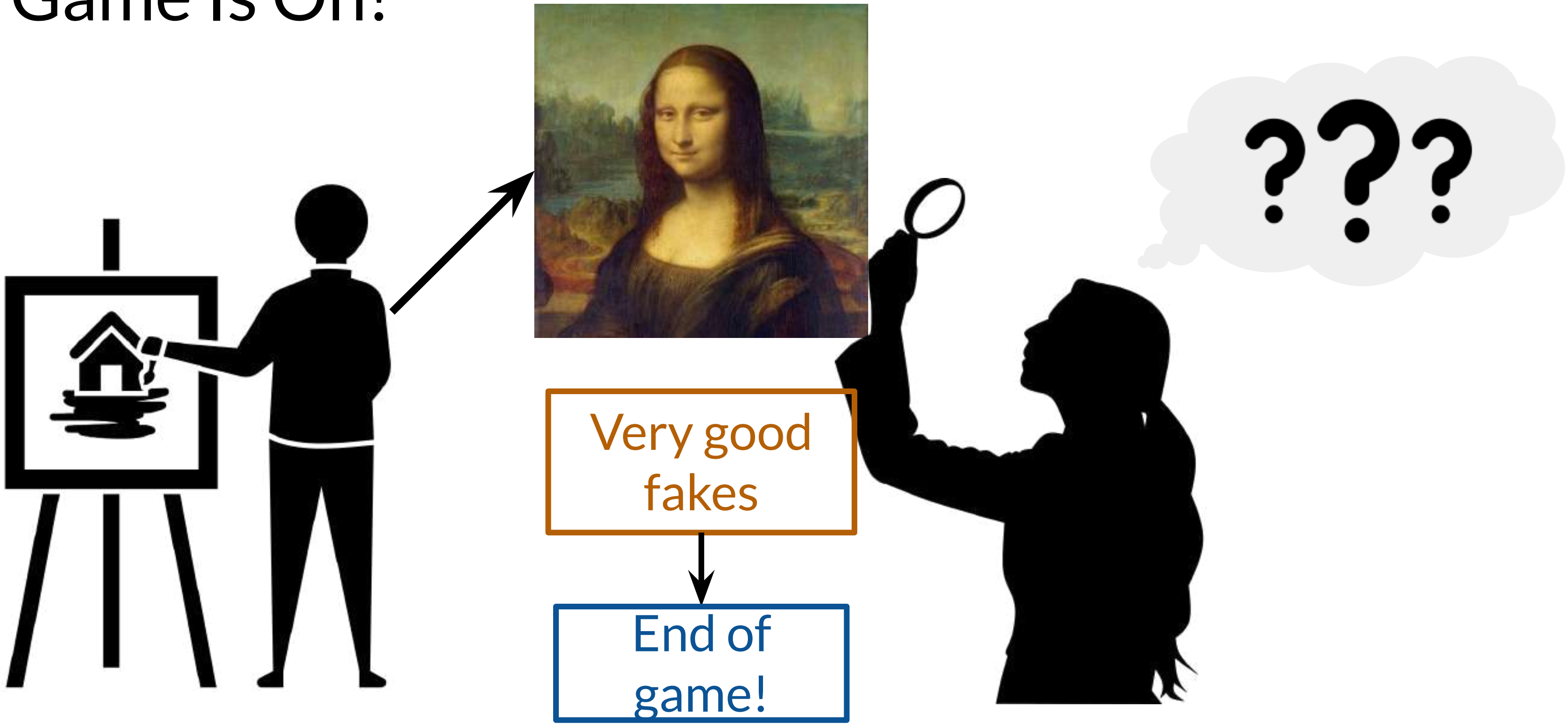
60% Real

95% Real

The Game Is On!



The Game Is On!



Summary

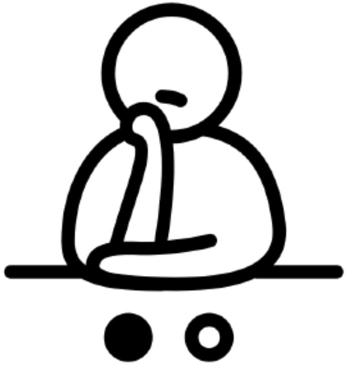
- The generator’s goal is to fool the discriminator
- The discriminator’s goal is to distinguish between real and fake
- They learn from the competition with each other
- At the end, *fakes* look real

deeplearning.ai

Discriminator

Outline

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



Classifiers

Distinguish between different classes

Classifier

Turtle

Bird

Cat

Dog

Fish

Classifiers

Distinguish between different classes

"It meows, and plays with yarn" → Classifier → Cat

Turtle

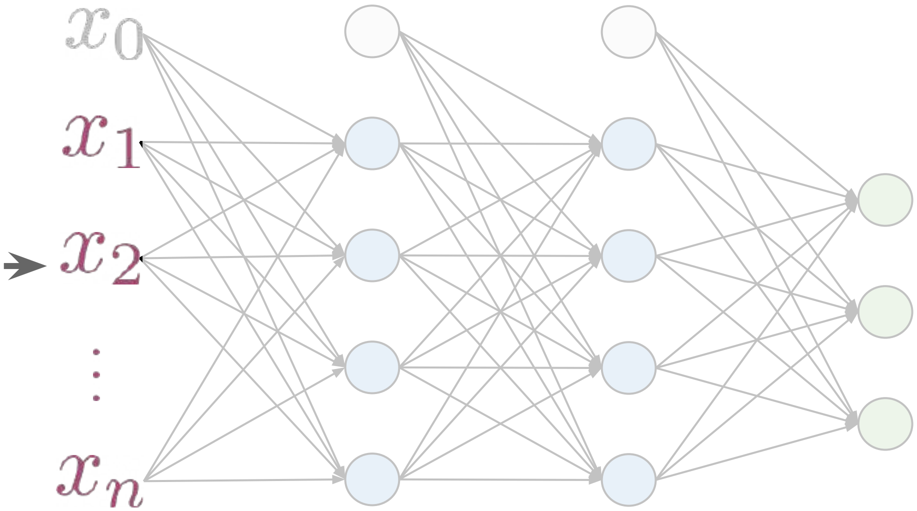
Bird

Cat

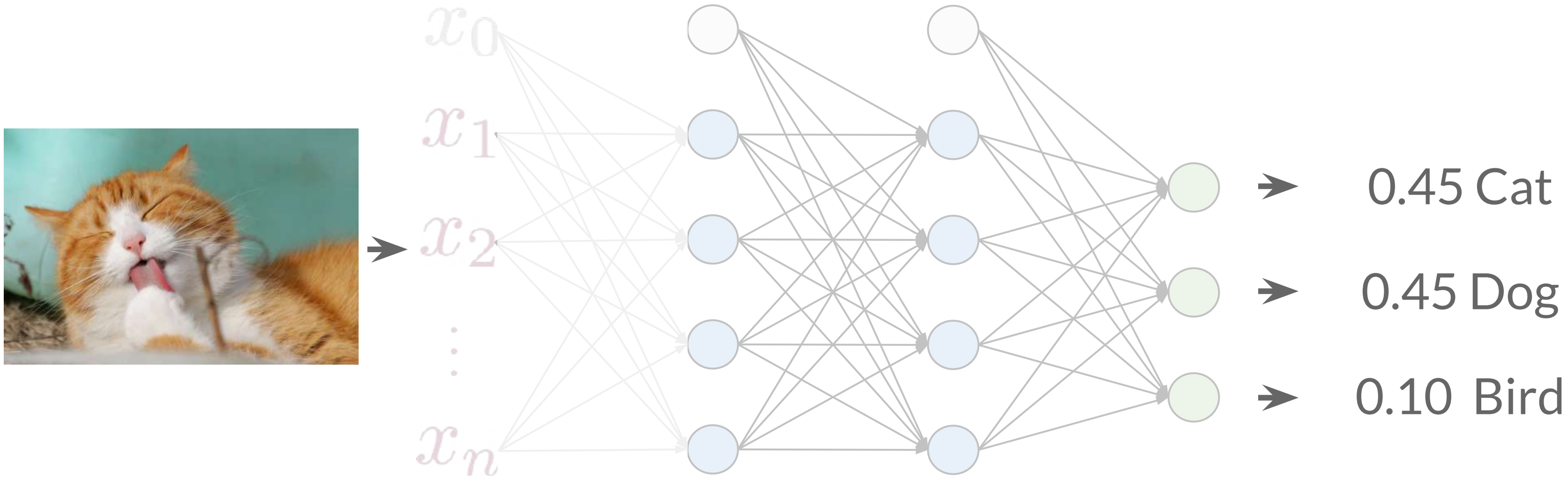
Dog

Fish

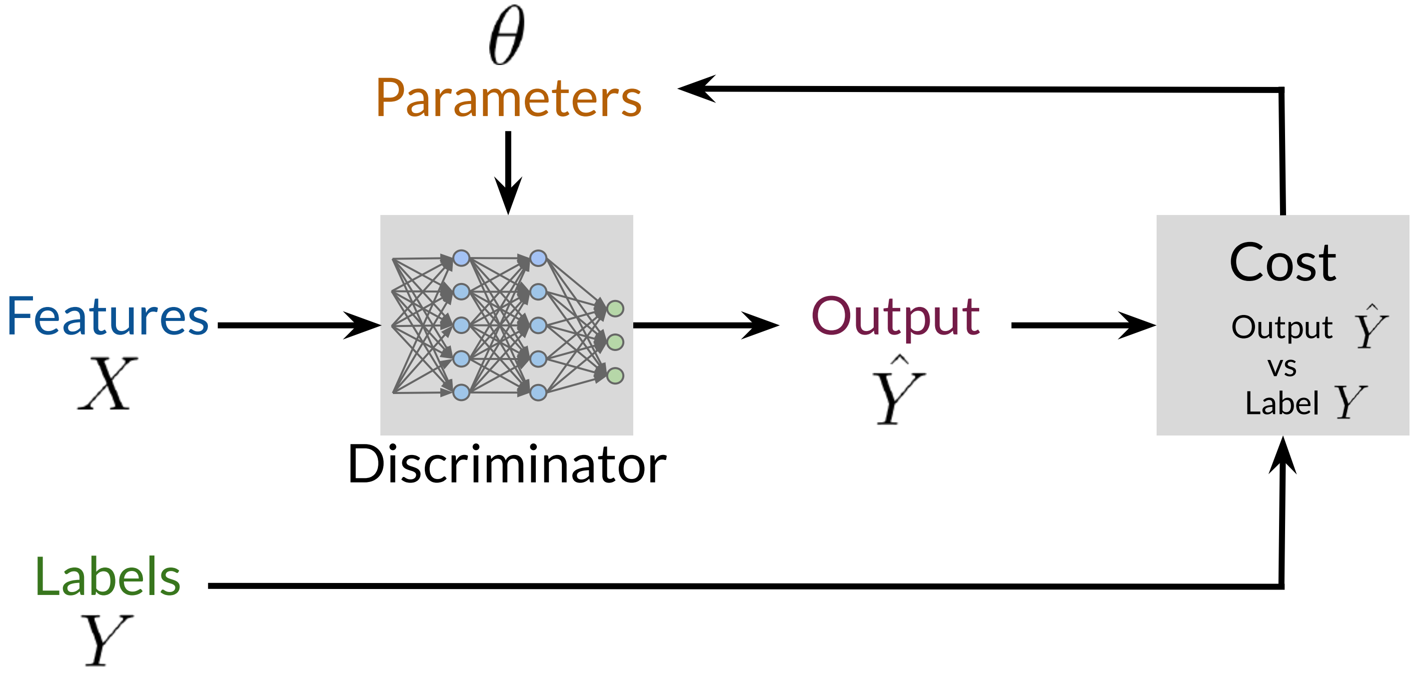
Neural Networks



Neural Networks



Classifiers (training)



Classifiers

Turtle

Bird

$P($ Cat $|$  $)$

Dog

Fish

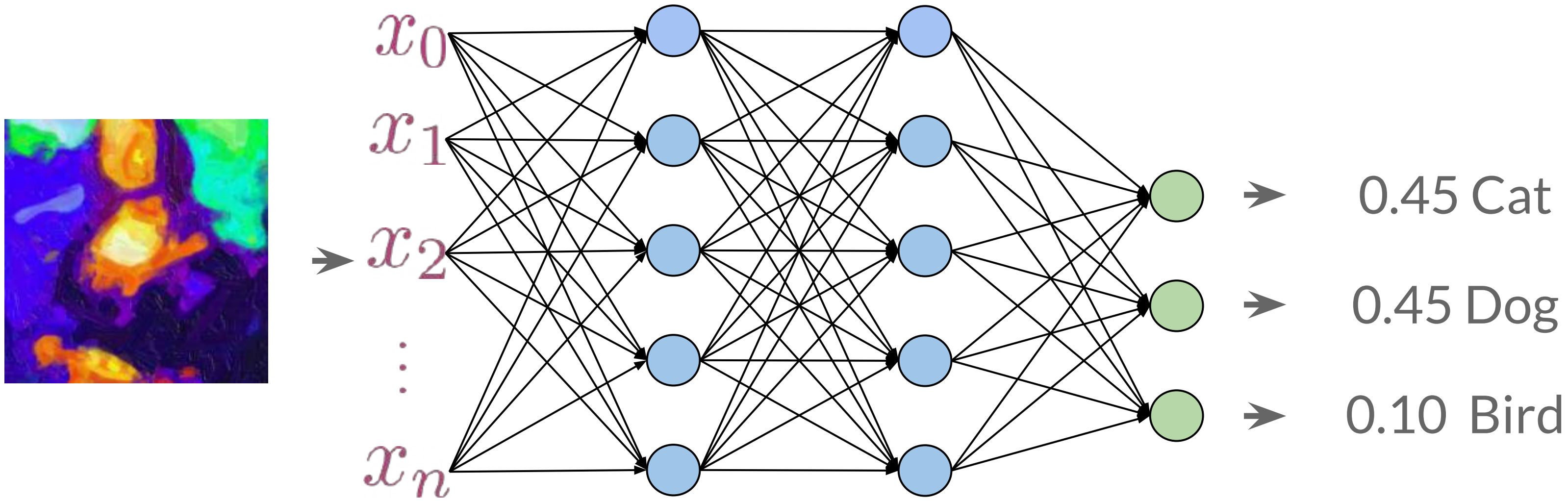
Classifiers

Conditional Probability

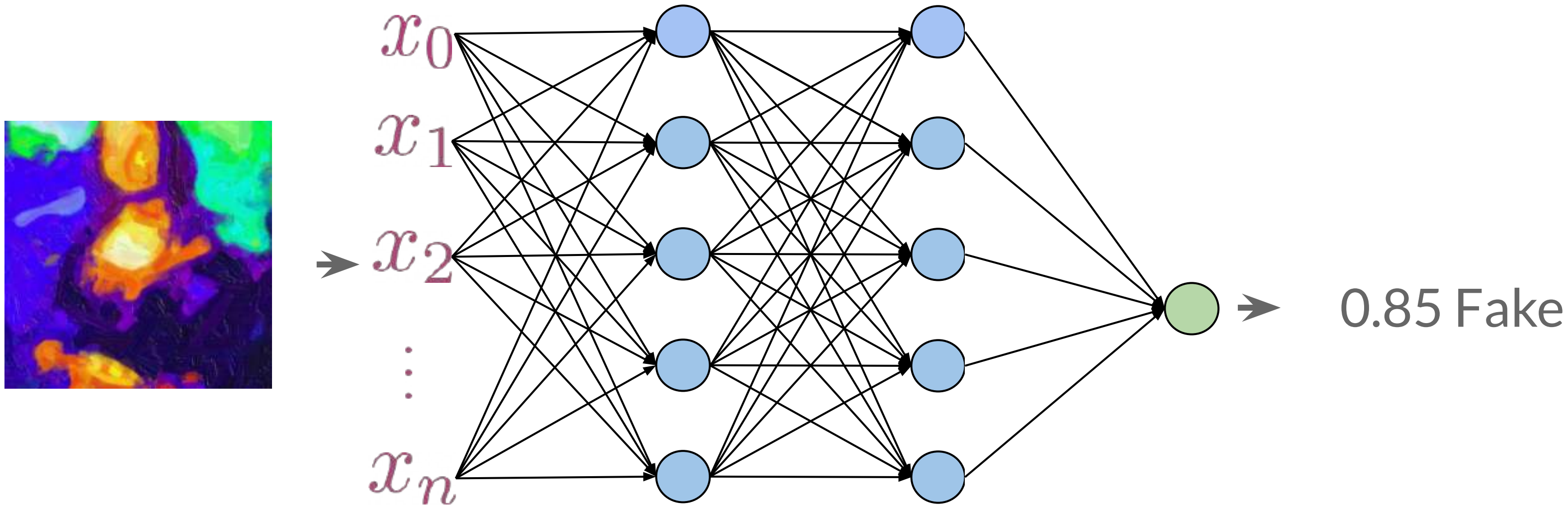
$$P(Y \mid X)$$

Class Features

Discriminator



Discriminator

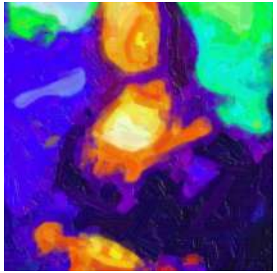


Discriminator

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

Class Features

Discriminator

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

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

Generator

Turtle

Generates examples of the class

Bird

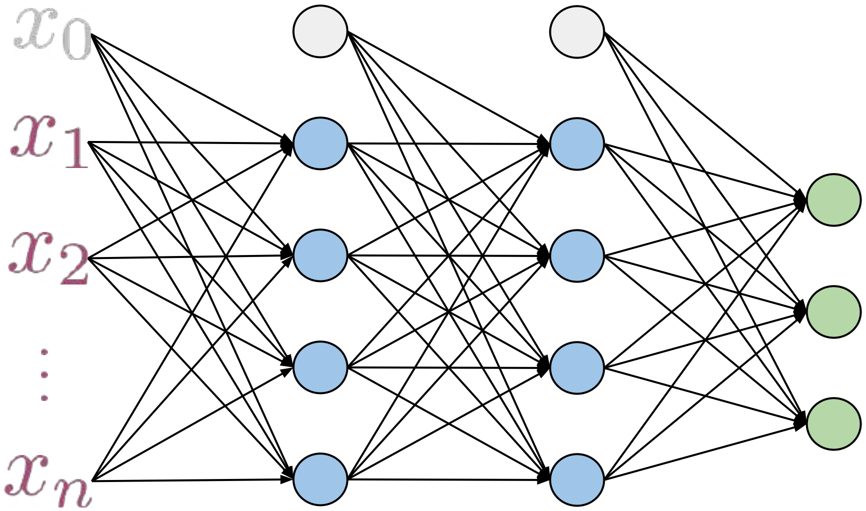
Cat → Generator → 

Dog

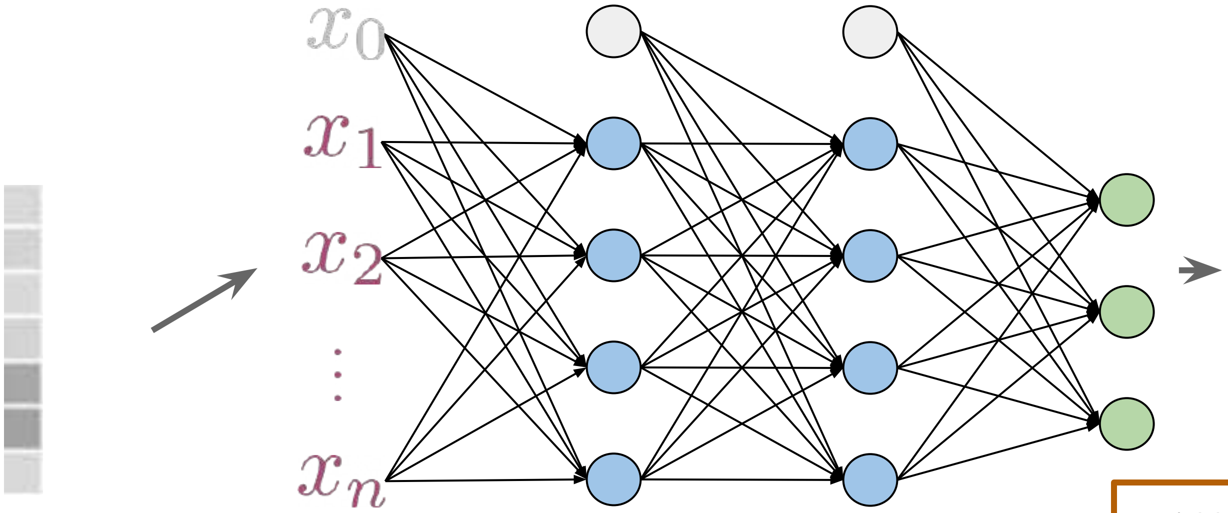
Fish

Neural Networks

Noise
(random features)



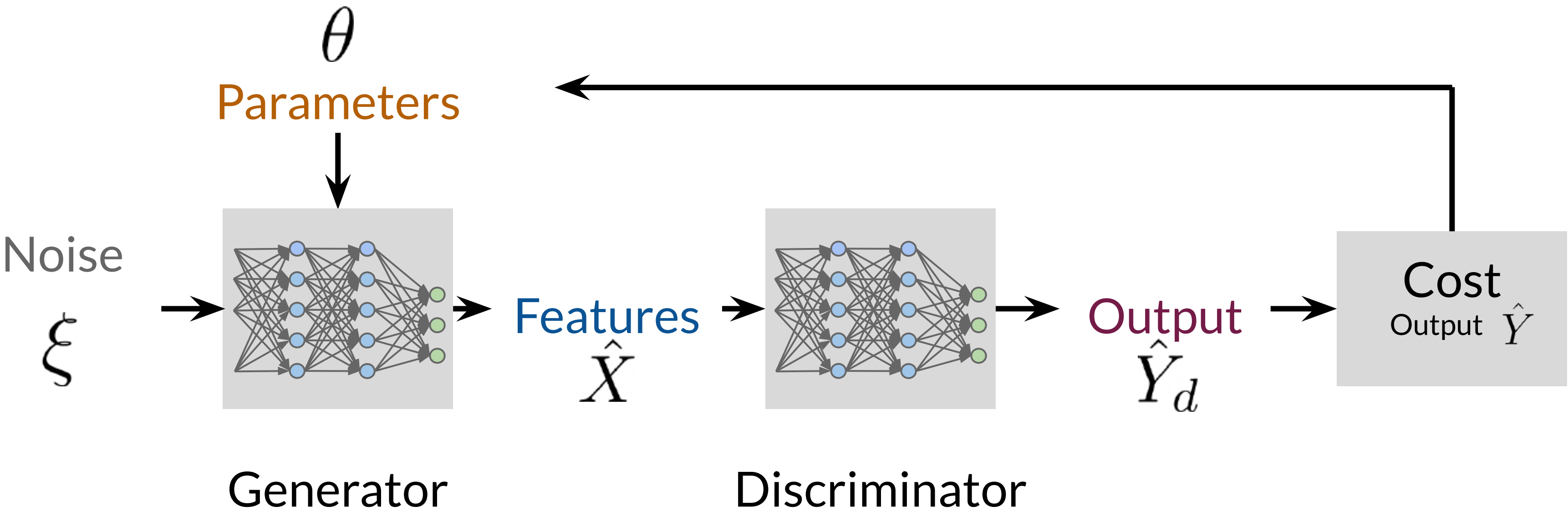
Neural Networks



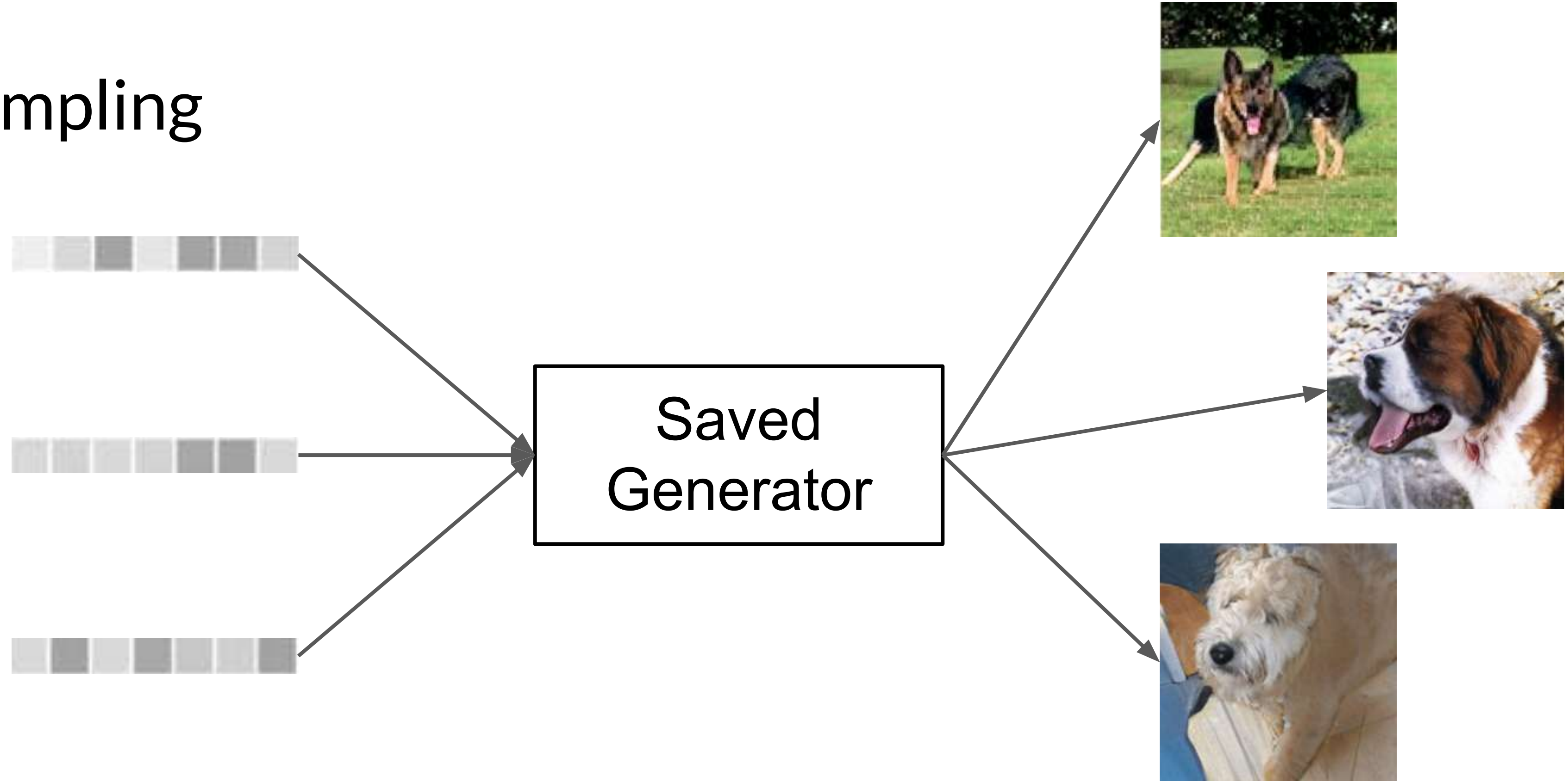
Noise
(random features)

Different outputs at every run!

Generator: Learning



Sampling



Generator

Turtle

Bird

$P($  $|$ Cat $)$

Dog

Fish

Generator

Conditional Probability

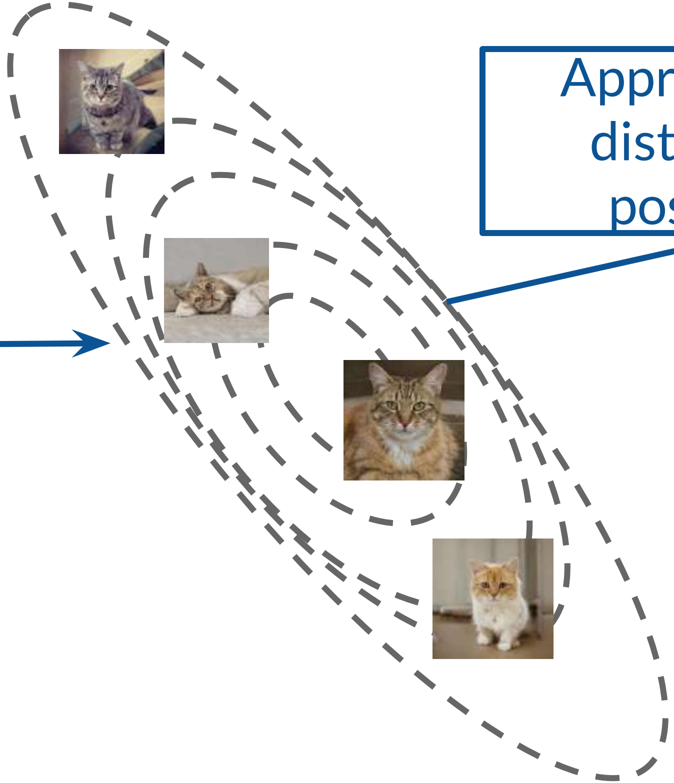
$$P(X \mid Y)$$

Features Class

Generator

$$P(X)$$

Features



Approximate the distribution of possible cats

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

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

Prediction

Label

Features

Parameters

Average loss of the whole batch

BCE Cost Function

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

BCE Cost Function

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

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

Relevant when the label is 1

BCE Cost Function

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

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

Relevant when the label is 0

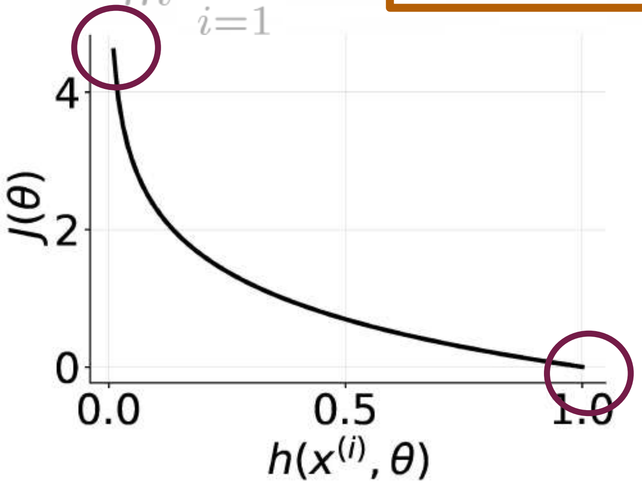
BCE Cost Function

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

Ensures that the cost is always greater or equal to 0

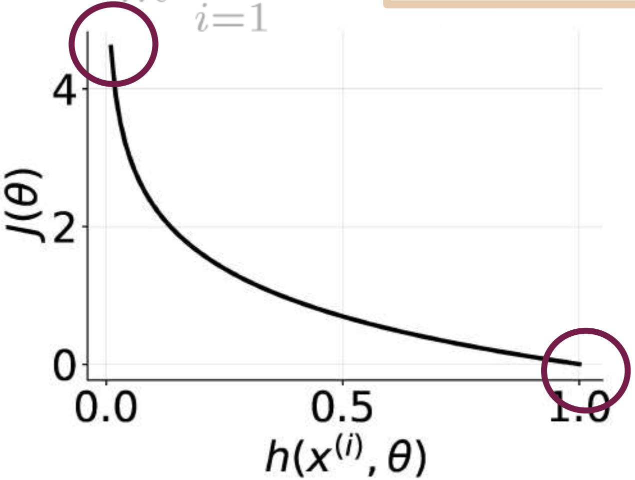
BCE Cost Function

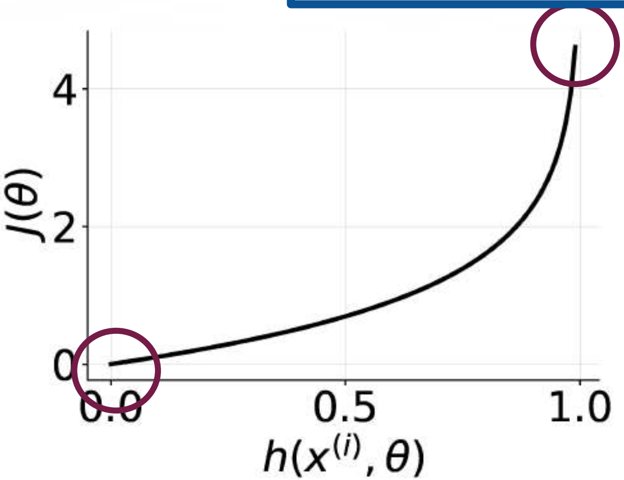
$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\boxed{\log h(x^{(i)},\theta)} + (1-y^{(i)})\boxed{\log(1-h(x^{(i)},\theta))}]$$



BCE Cost Function

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





Summary

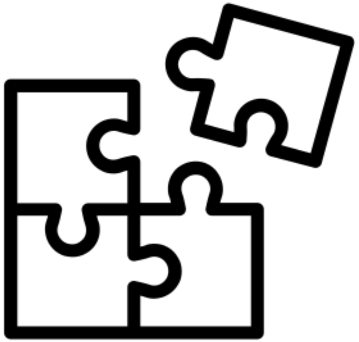
- The BCE cost function has two parts (one relevant for each class)
- Close to zero when the label and the prediction are similar
- Approaches infinity when the label and the prediction are different

deeplearning.ai

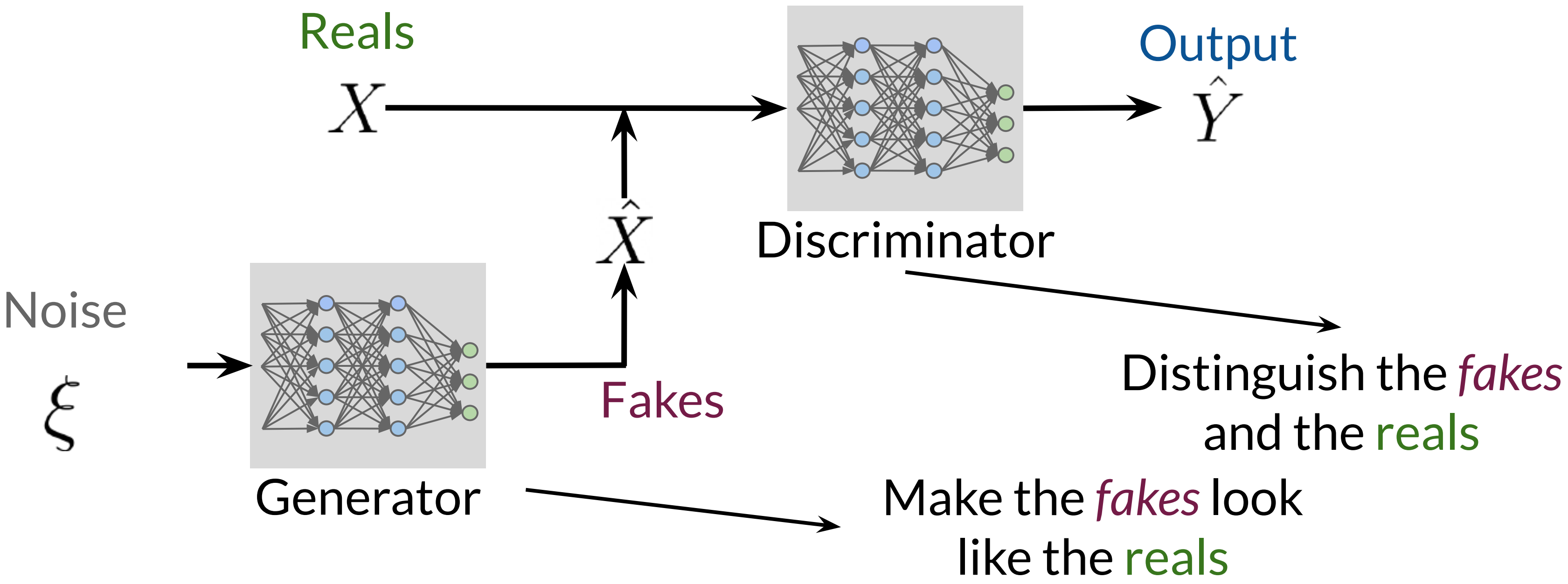
Putting It All Together

Outline

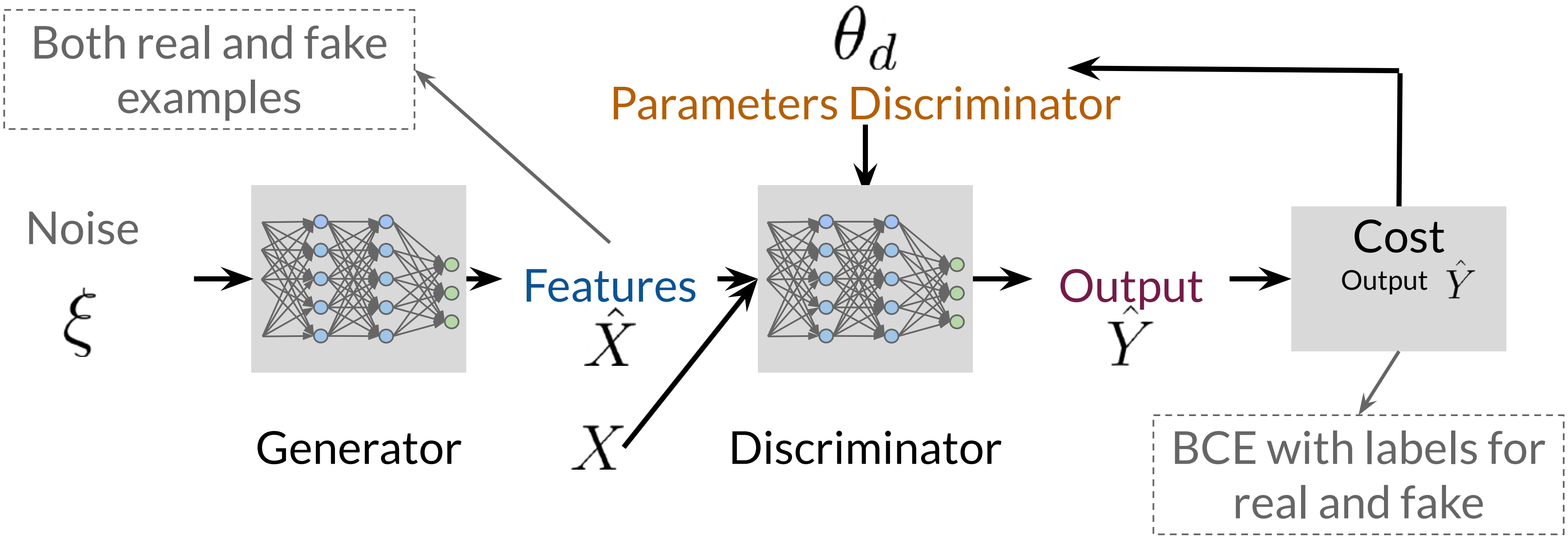
- How the whole architecture looks
- How to train GANs



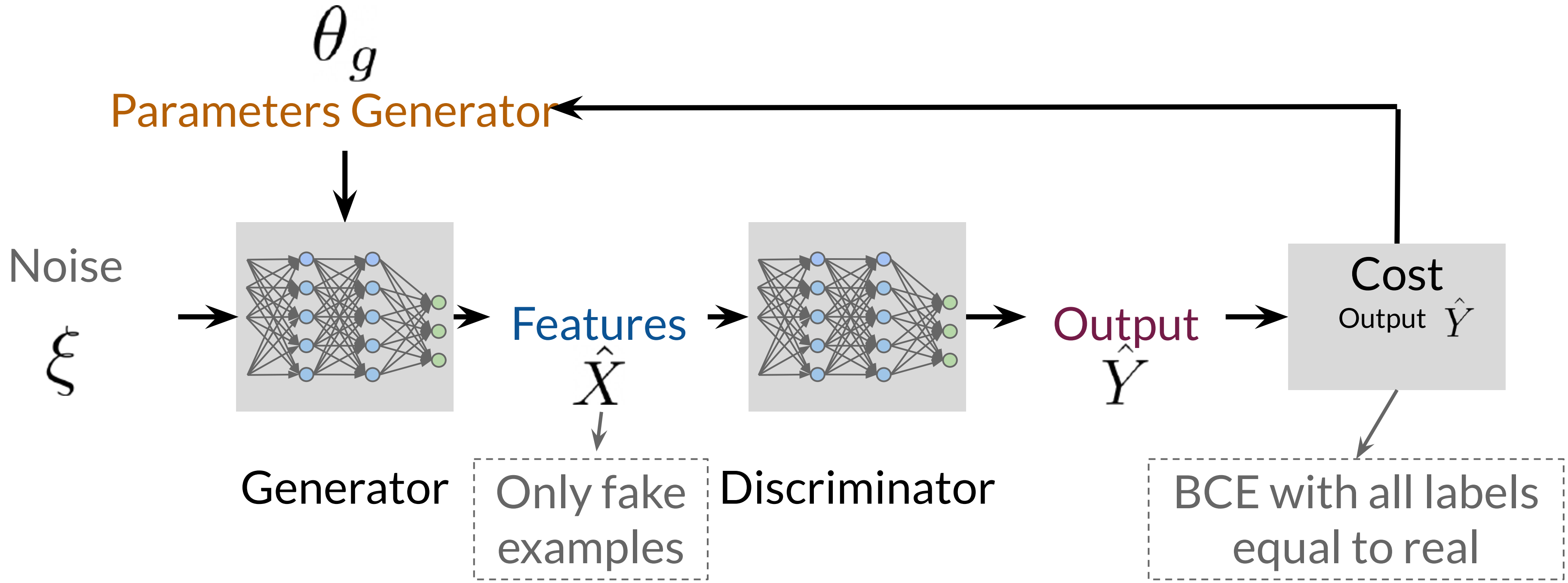
GANs Model


Reals
X
Output
$\hat{Y}$
$\hat{X}$
Discriminator
Noise
ξ
Fakes
Generator
Distinguish the *fakes* and the reals
Make the *fakes* look like the reals

Training GANs: Discriminator



Training GANs: Generator



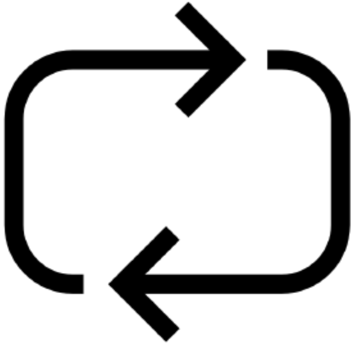
Training GANs

Discriminator Output

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

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

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

3

Dynamic Computational Graphs

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

3

Static Computational Graphs

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!

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.

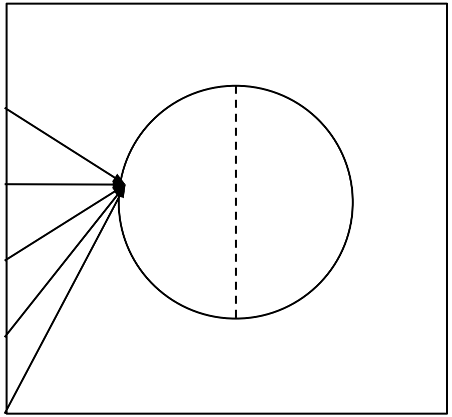
For the rest of the details of the license, see https://creativecommons.org/licenses/by-sa/2.0/legalcode

deeplearning.ai

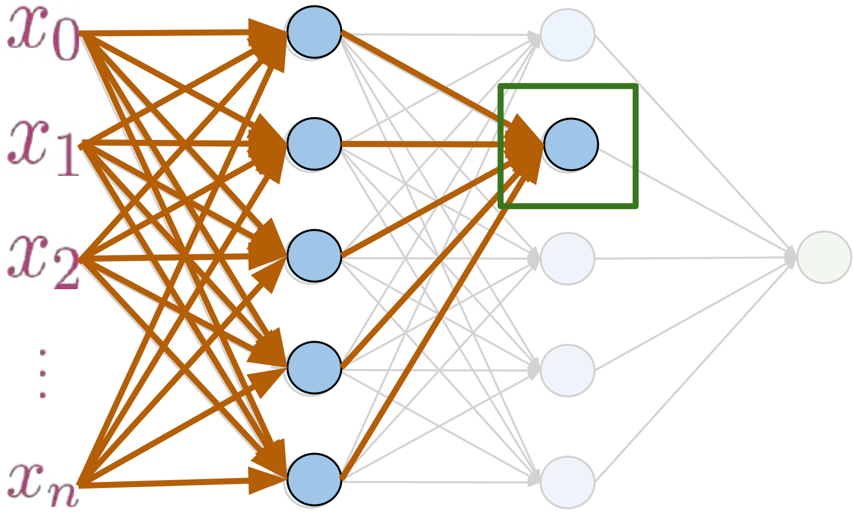
Activations (Basic Properties)

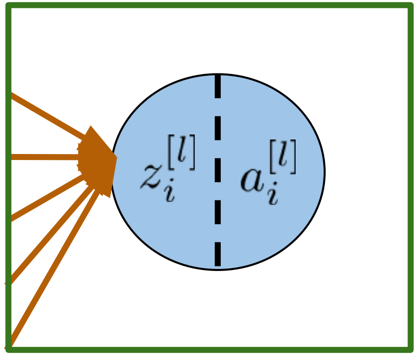
Outline

- What are activations
- Reasoning behind non-linear differential activations



Activations





$$z_i^{[l]} = \sum_{i=0} W_i^{[l]} a_i^{[l-1]}$$

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

Differentiable non-linear function

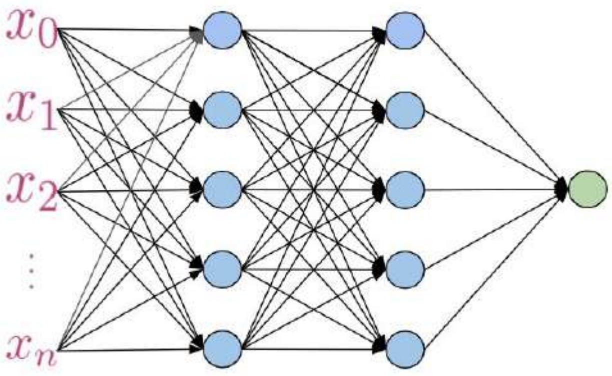
Activations

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

Differentiable non-linear function

1. Differentiable for backpropagation

2. Non-linear to compute complex features, **if not**:



$$\equiv \quad WX + b$$

Linear regression

Summary

- Activation functions are non-linear and differentiable
- Differentiable for backpropagation
- Non-linear to approximate complex functions

deeplearning.ai

Common Activation Functions

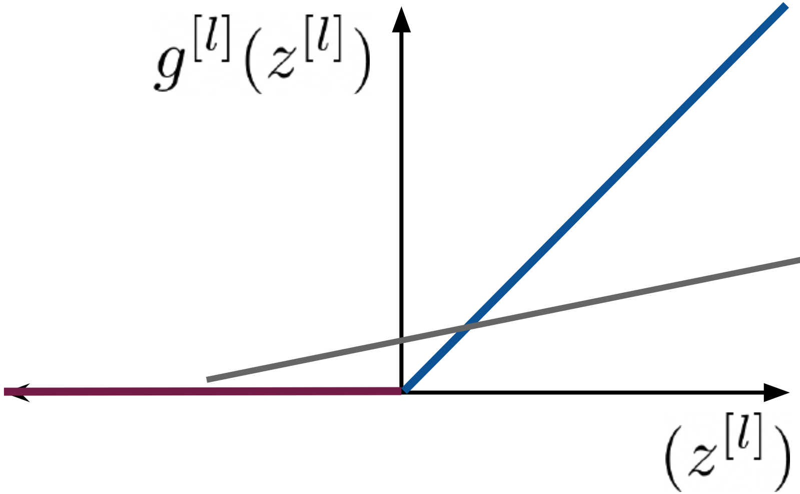
Outline

- Common activations and their structure
  - ReLU
  - Leaky ReLU
  - Sigmoid
  - Tanh

Activations: ReLU

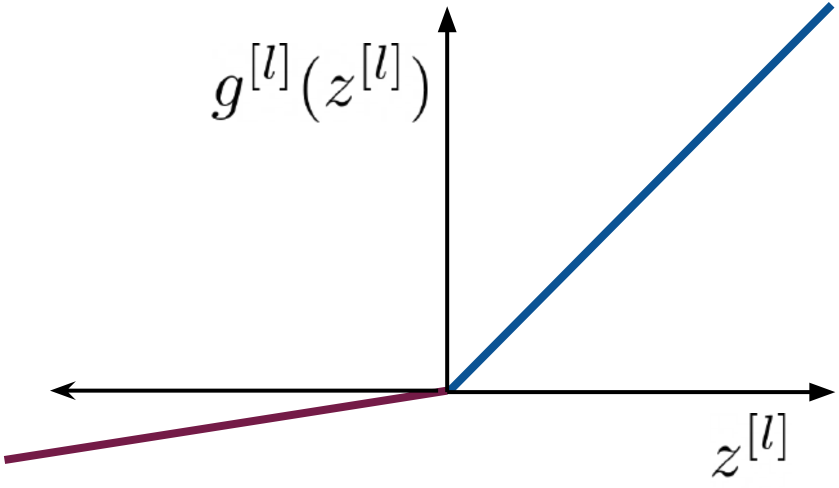
ReLU = Rectified Linear Unit

$$g^{[l]}(z^{[l]}) = \max(\underline{0}, \underline{z^{[l]}})$$



Dying ReLU problem

Activations: Leaky ReLU

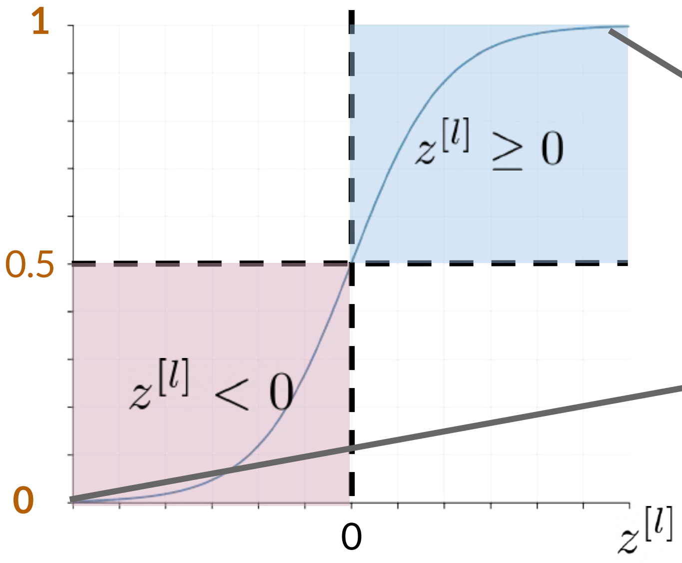


$$g^{[l]}(z^{[l]}) = max(\underline{az^{[l]}}, \underline{z^{[l]}})$$

Solves the dying ReLU problem

Activations: Sigmoid

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

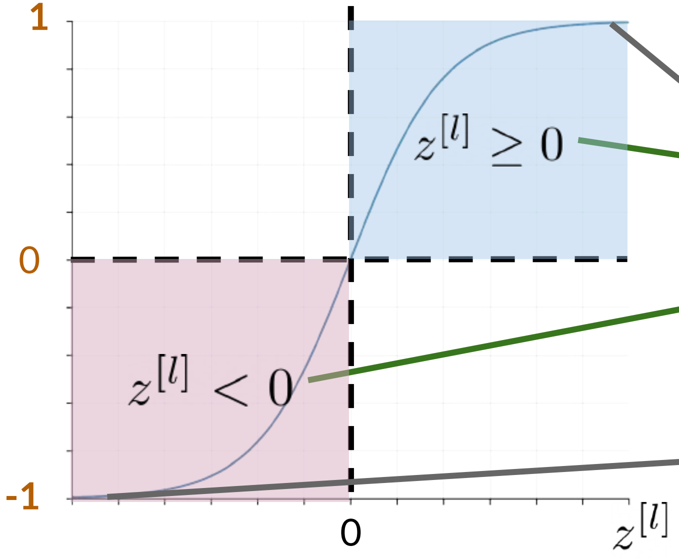


Values between 0 and 1

Vanishing gradient and saturation problems

Activations: Tanh

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



Values between -1 and 1

Keeps the sign of the input

Same issues as Sigmoid

Summary

- ReLU activations suffer from dying ReLU
- Leaky ReLU solve the dying ReLu problem
- Sigmoid and Tanh have vanishing gradient and saturation problems

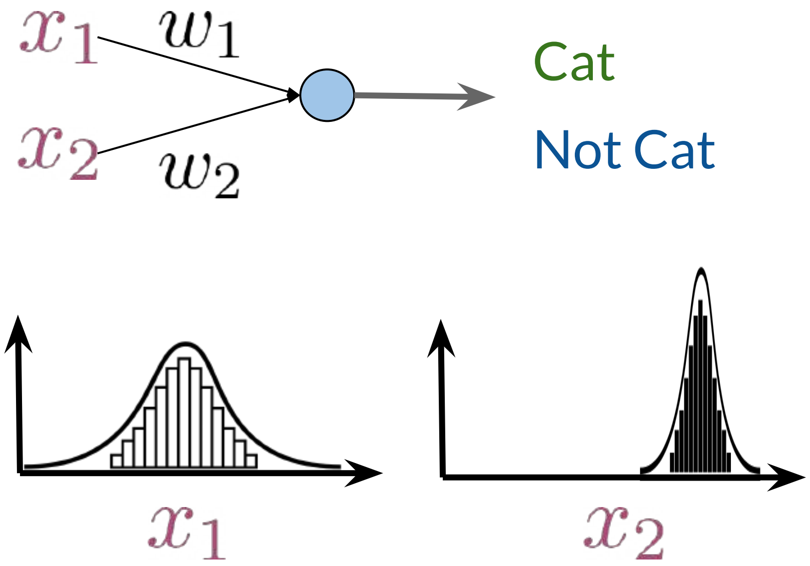
deeplearning.ai

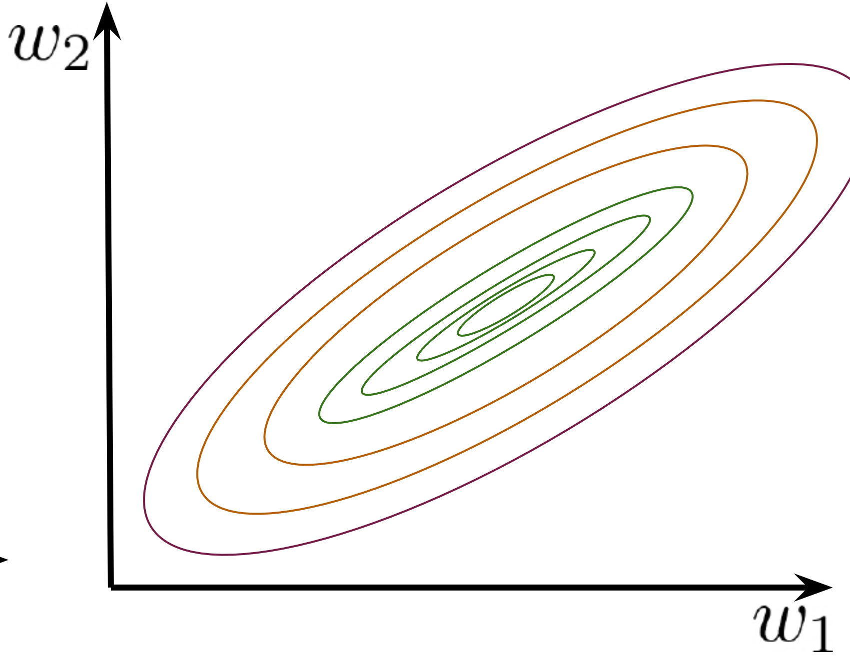
Batch Normalization (Explained)

Outline

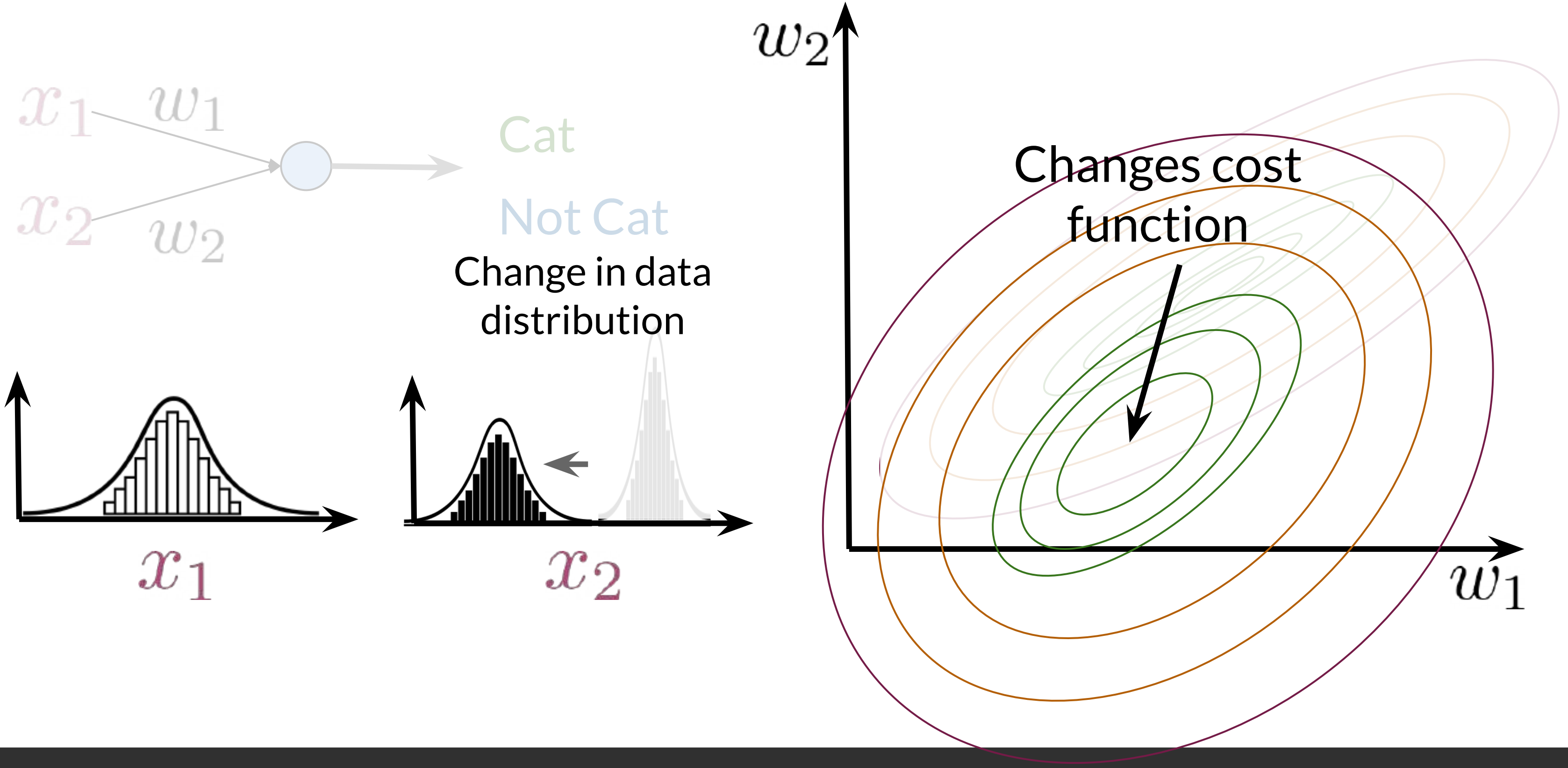
- How normalization helps models
- Internal covariate shift
- Batch normalization

Different Distributions

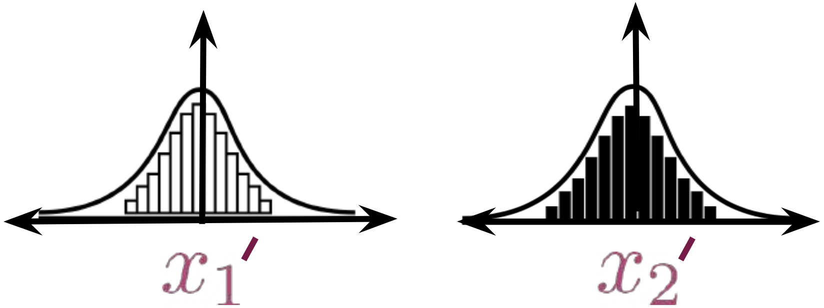




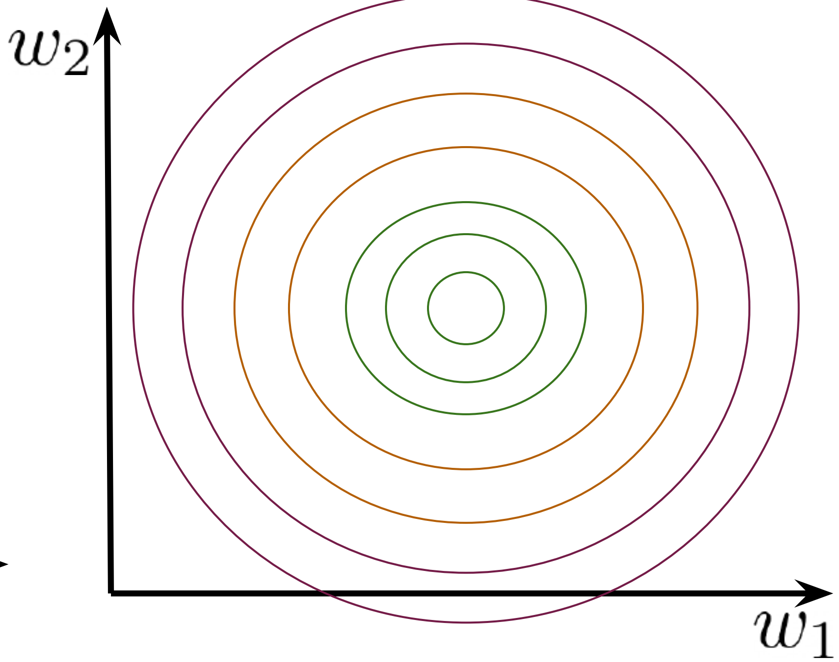
Covariate Shift



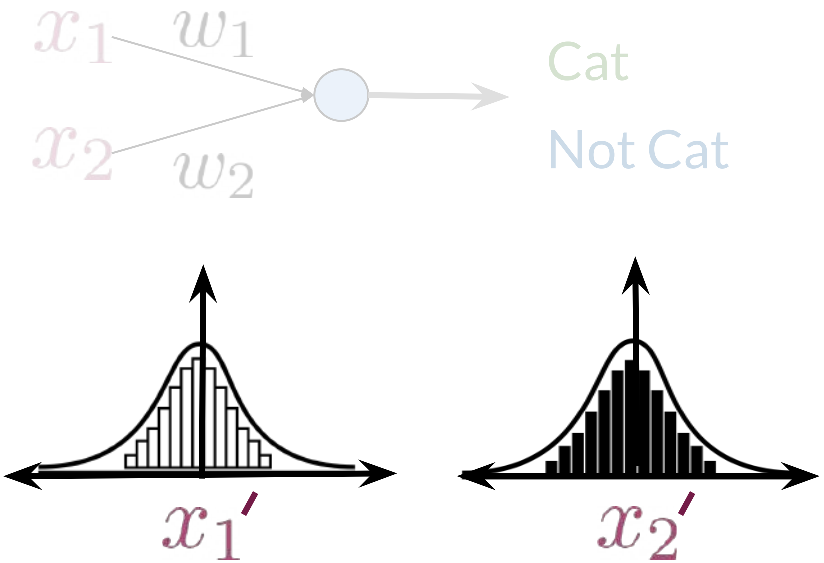
Normalization and Its Effects

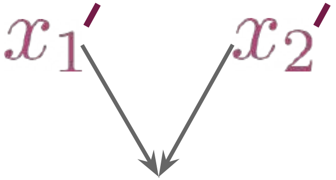


Cost function



Normalization and Its Effects



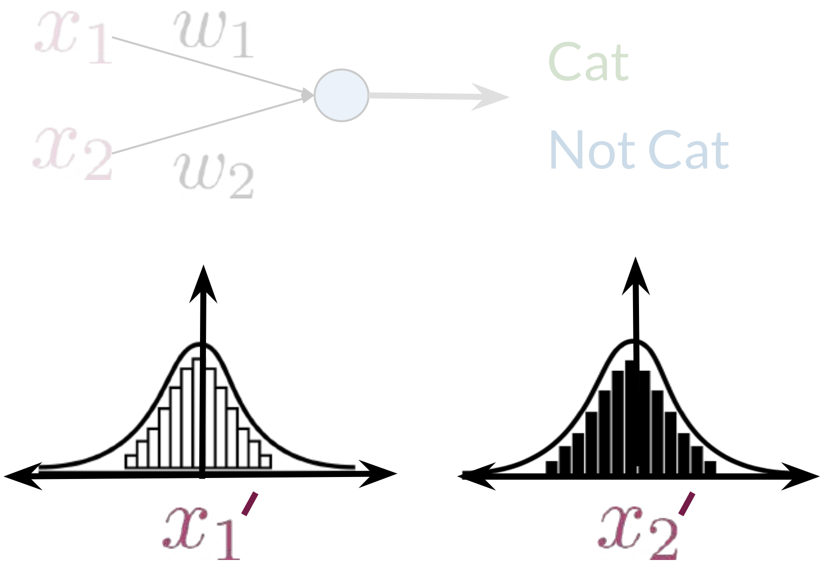


Around mean at 0
and std. at 1

Training data uses batch stats

Test data uses training stats

Normalization and Its Effects



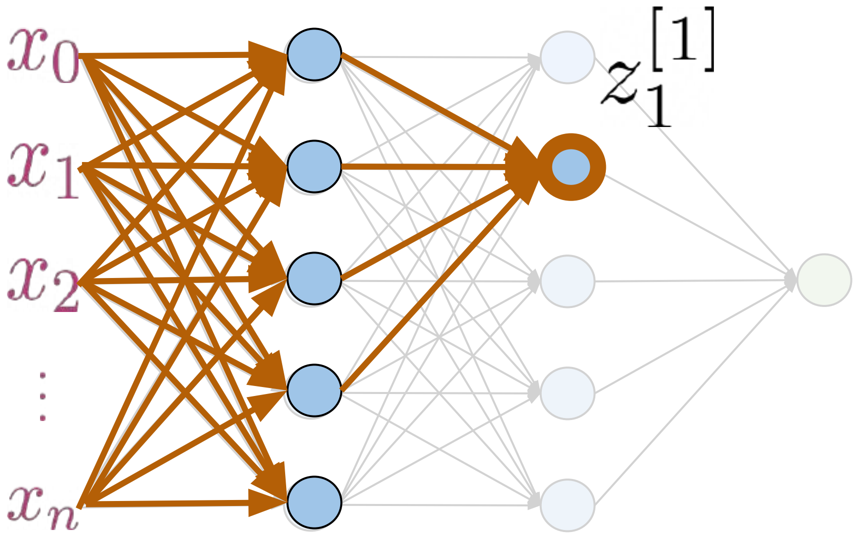
x_1' x_2'

Around mean at 0
and std. at 1

Reduction of
covariate shift

deeplearning.ai

Internal Covariate Shift

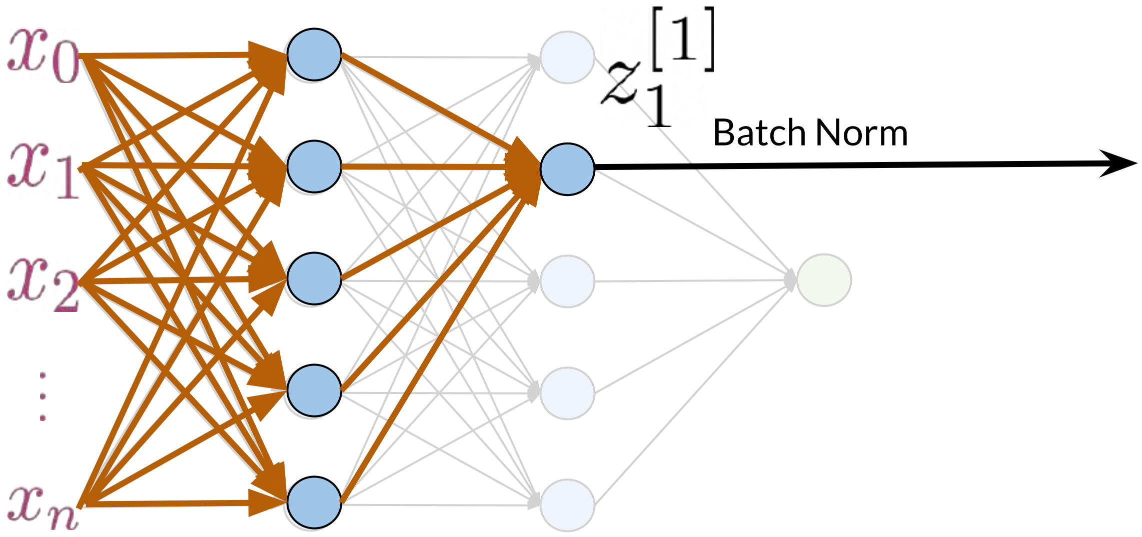


Changes in weights

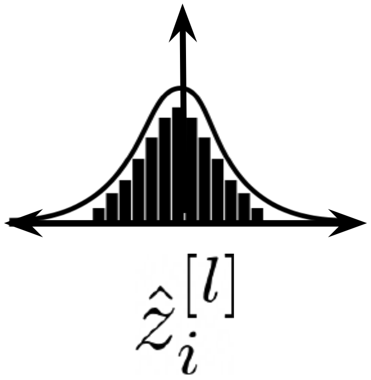
↓

Changes in activation distribution

Batch Normalization

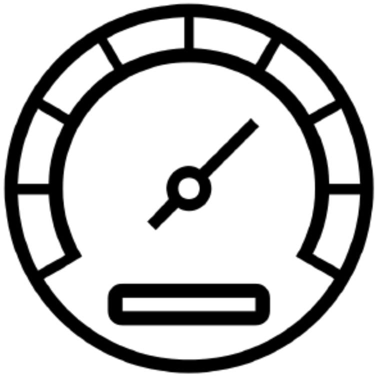


Normalizes the input for each neuron



Summary

- Batch normalization smooths the cost function
- Batch normalization reduces the internal covariate shift
- Batch normalization speeds up learning!

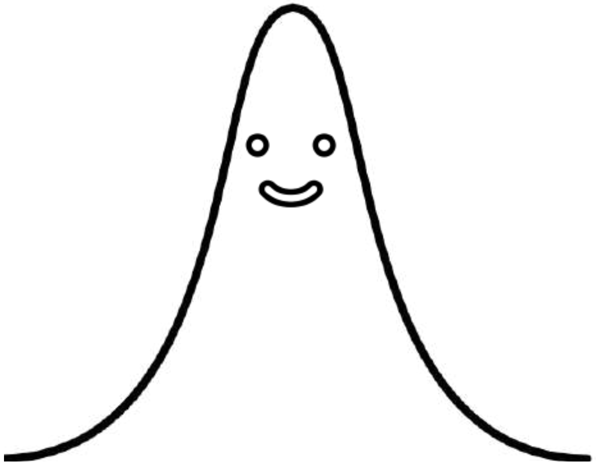


deeplearning.ai

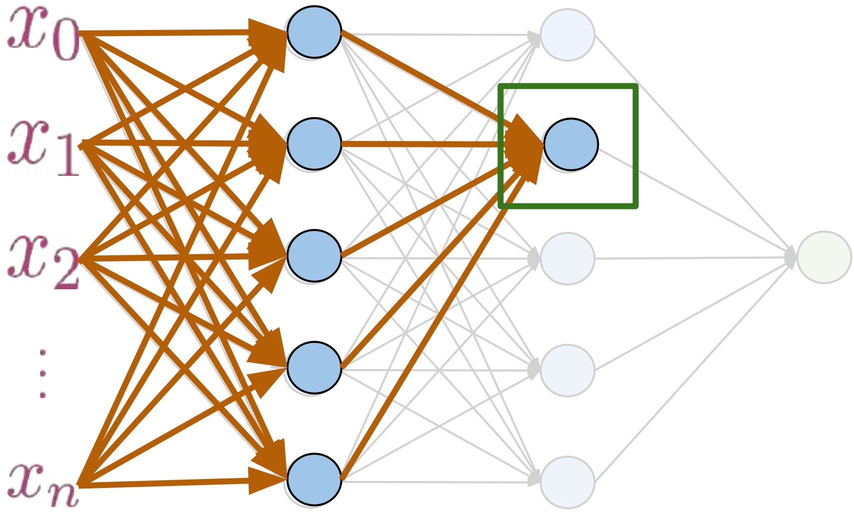
Batch Normalization (Procedure)

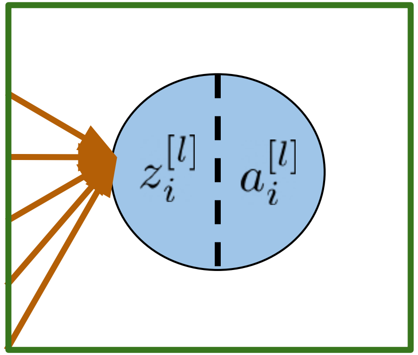
Outline

- Batch norm for training
- Batch norm for testing



Batch Normalization: Training





$$z_i^{[l]} = \sum_{i=0} W_i^{[l]} a_i^{[l-1]}$$

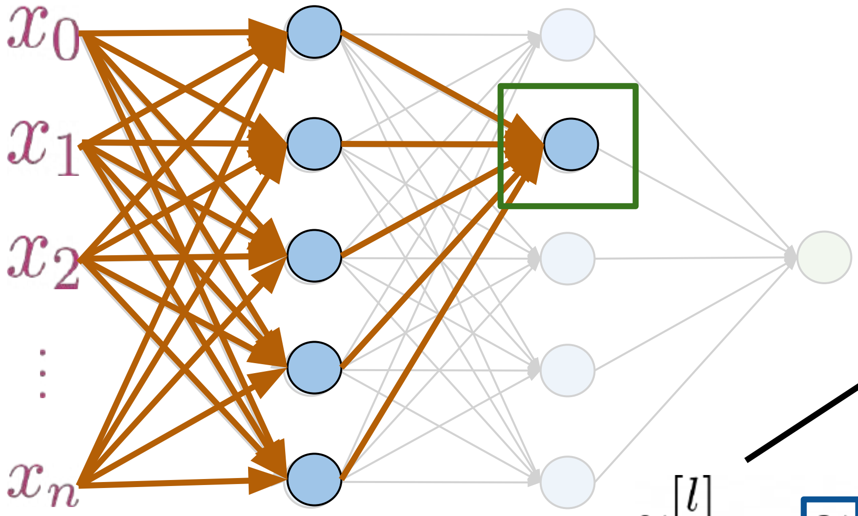
→ For every example in the batch

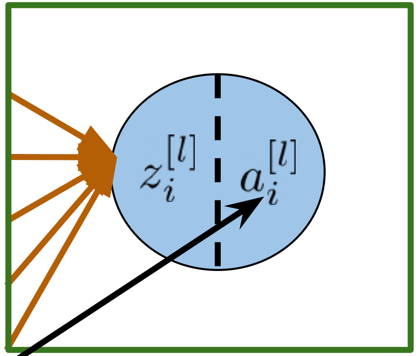
$$\hat{z}_i^{[l]} = \frac{z_i^{[l]} - \boxed{\mu_{z_i^{[l]}}}}{\boxed{\sqrt{\sigma^2_{z_i^{[l]}} + \epsilon}}}$$

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

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

Batch Normalization: Training





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

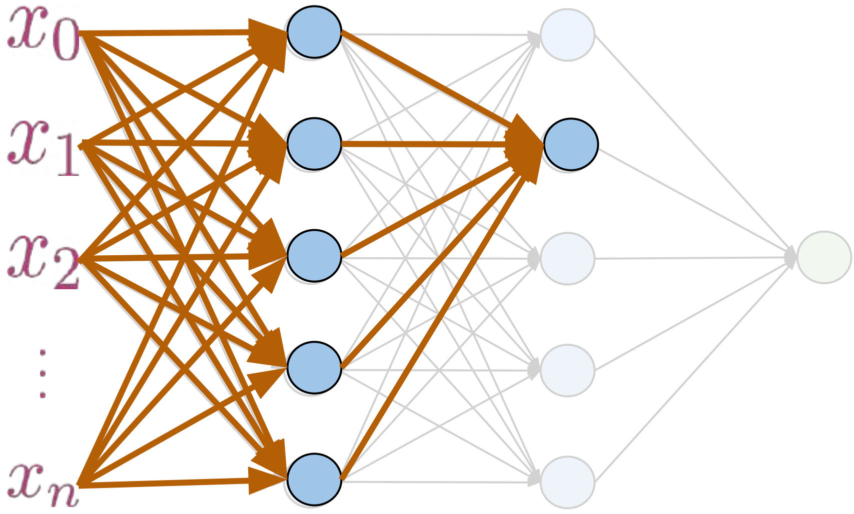
$$y_i^{[l]} = \boxed{\gamma}\hat{z}_i^{[l]} + \boxed{\beta}$$

Shift factor

Scale Factor

Learnable parameters to get the optimal dist.

Batch Normalization: Test



$$\hat{z}_i^{[l]} = \frac{z_i^{[l]} - \boxed{\mathbf{E}(z_i^{[l]})}}{\boxed{\sqrt{\mathrm{Var}(z_i^{[l]}) + \epsilon}}}$$

Running mean and running std from training

Frameworks like Tensorflow and Pytorch keep track of them

Summary

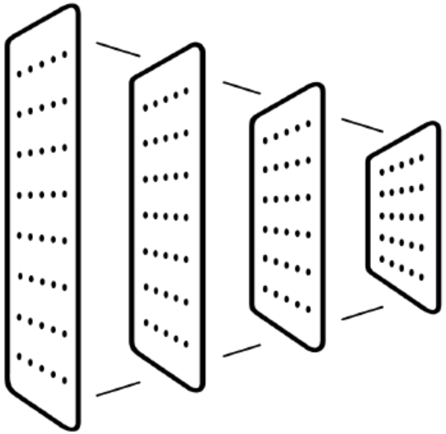
- Batch norm introduces learnable shift and scale factors
- During test, the running statistics from training are used
- Frameworks take care of the whole process

deeplearning.ai

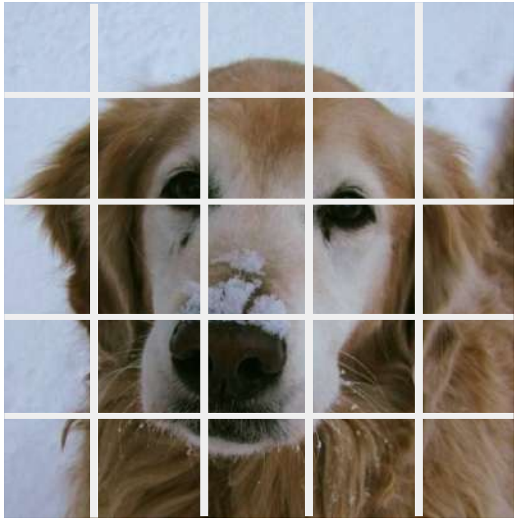
Review of Convolutions

Outline

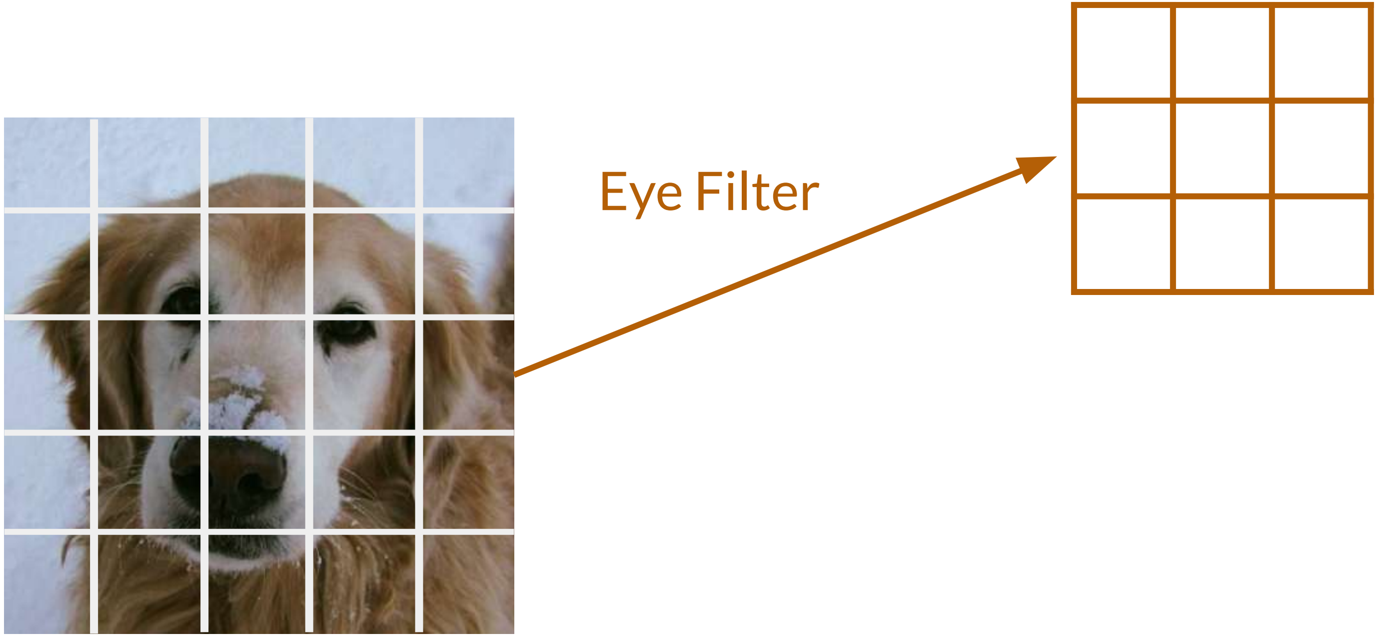
- What convolutions are
- How they work



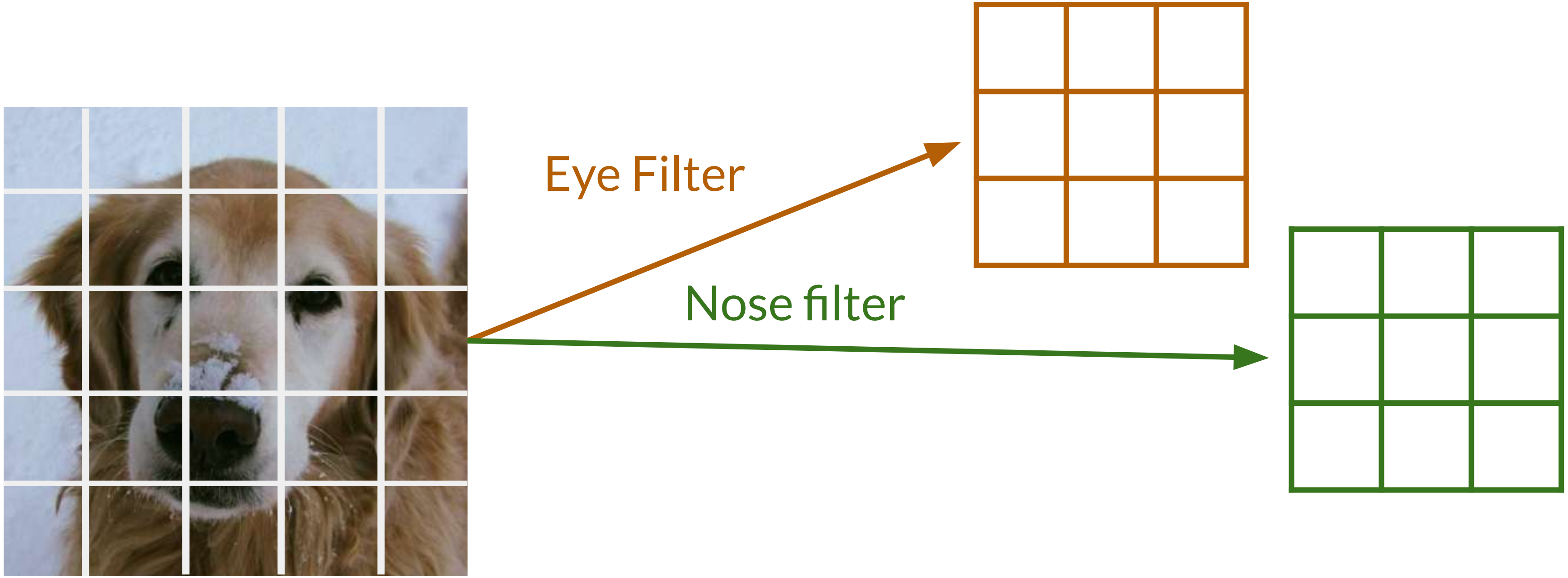
What is a convolution?



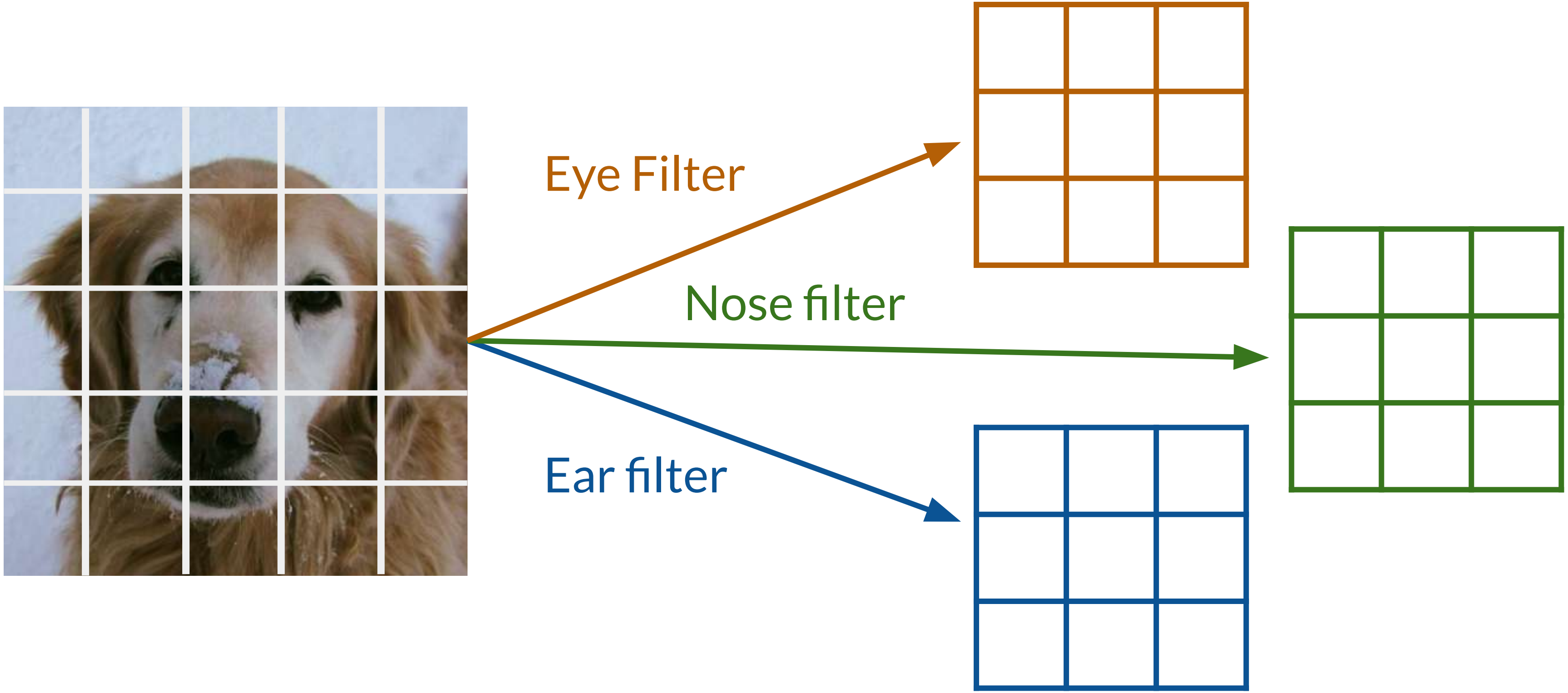
What is a convolution?



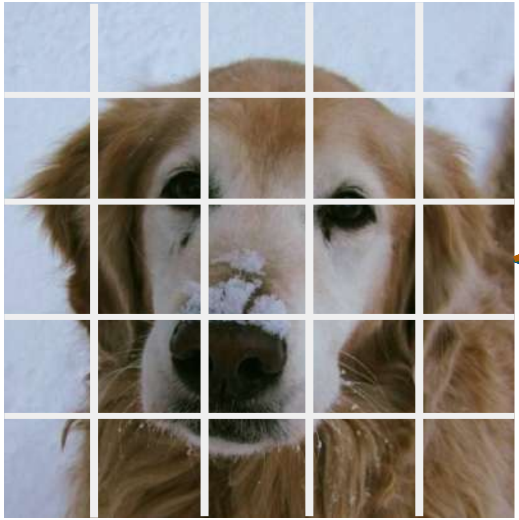
What is a convolution?



What is a convolution?



What is a convolution?



Eye Filter

5	3	1
10	23	1
1	42	4

Nose filter

2	21	5
23	45	12
3	32	4

Ear filter

32	2	24
25	12	66
3	45	2

What is a convolution?

50	50	0	0	0
50	50	0	0	0
50	50	0	0	0
50	50	0	0	0
50	50	0	0	0

Grayscale image

0 (black) to 255 (white)

What is a convolution?

50	50	0	0	0
50	50	0	0	0
50	50	0	0	0
50	50	0	0	0
50	50	0	0	0

Grayscale image

$*$

Filter

1	0	-1
1	0	-1
1	0	-1

0 (black) to 255 (white)

What is a convolution?

50	50	0	0	0
50	50	0	0	0
50	50	0	0	0
50	50	0	0	0
50	50	0	0	0

Grayscale image

$*$

Filter

1	0	-1
1	0	-1
1	0	-1

Convolution

0 (black) to 255 (white)

What is a convolution?

50*1	50*0	0*-1	0	0
50*1	50*0	0*-1	0	0
50*1	50*0	0*-1	0	0
50	50	0	0	0
50	50	0	0	0

Grayscale image

*

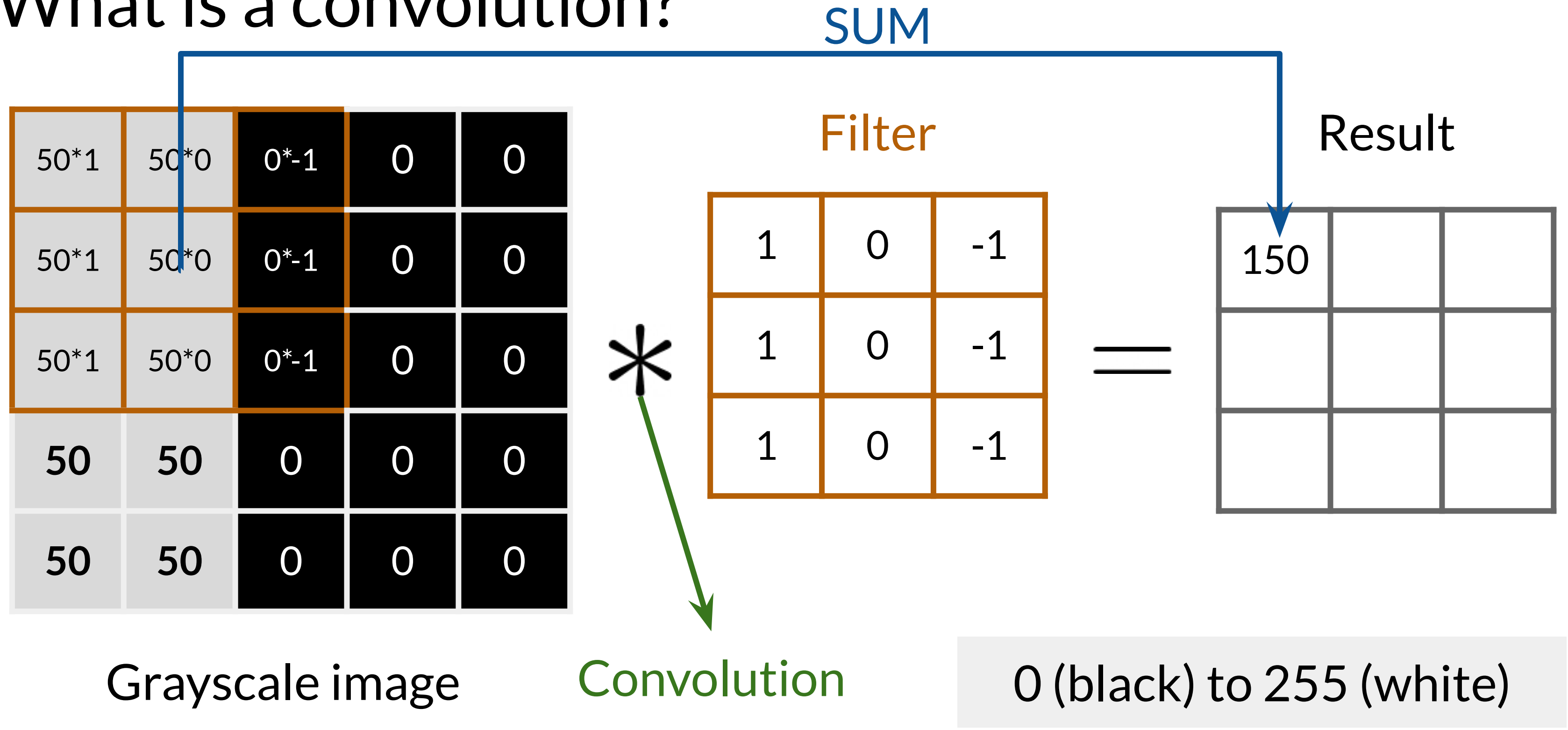
Filter

1	0	-1
1	0	-1
1	0	-1

Convolution

0 (black) to 255 (white)

What is a convolution?



What is a convolution?

50	50*1	0*0	0*-1	0
50	50*1	0*0	0*-1	0
50	50*1	0*0	0*-1	0
50	50	0	0	0
50	50	0	0	0

Grayscale image

*

Convolution

Filter

1	0	-1
1	0	-1
1	0	-1

=

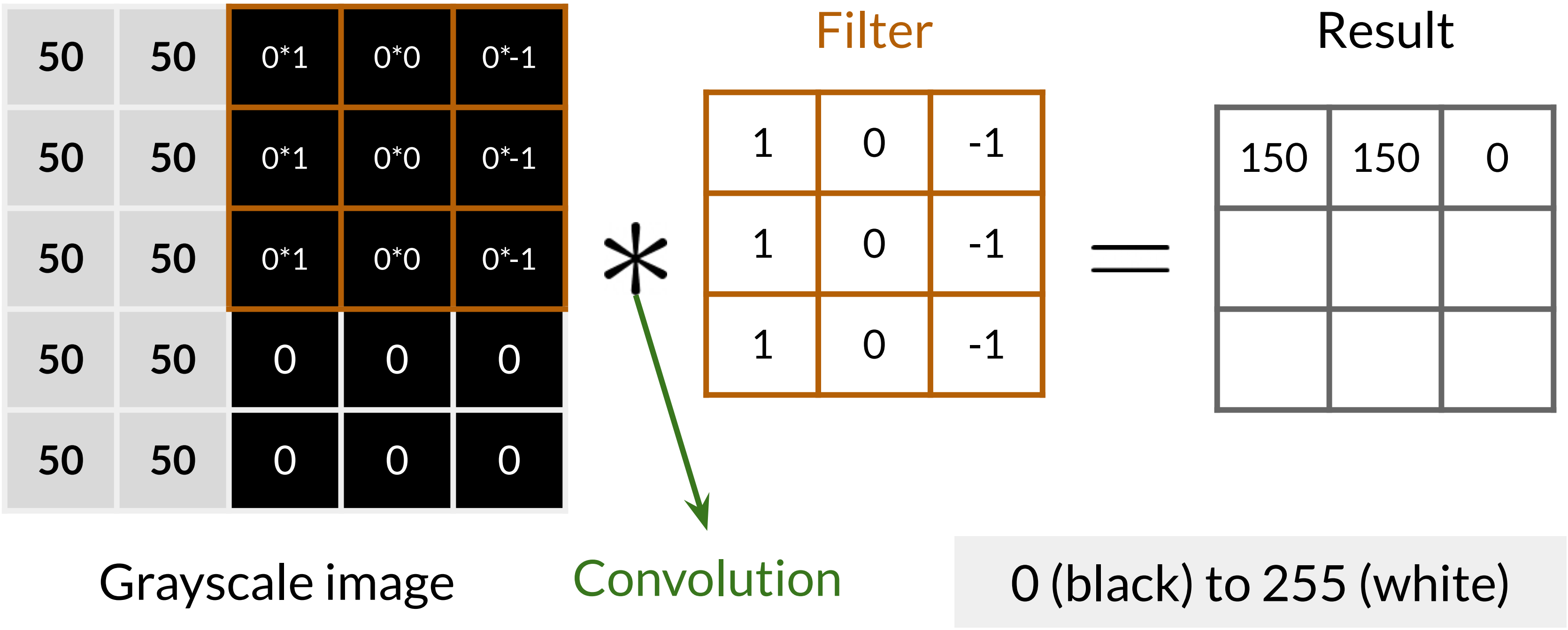
Result

150	150	

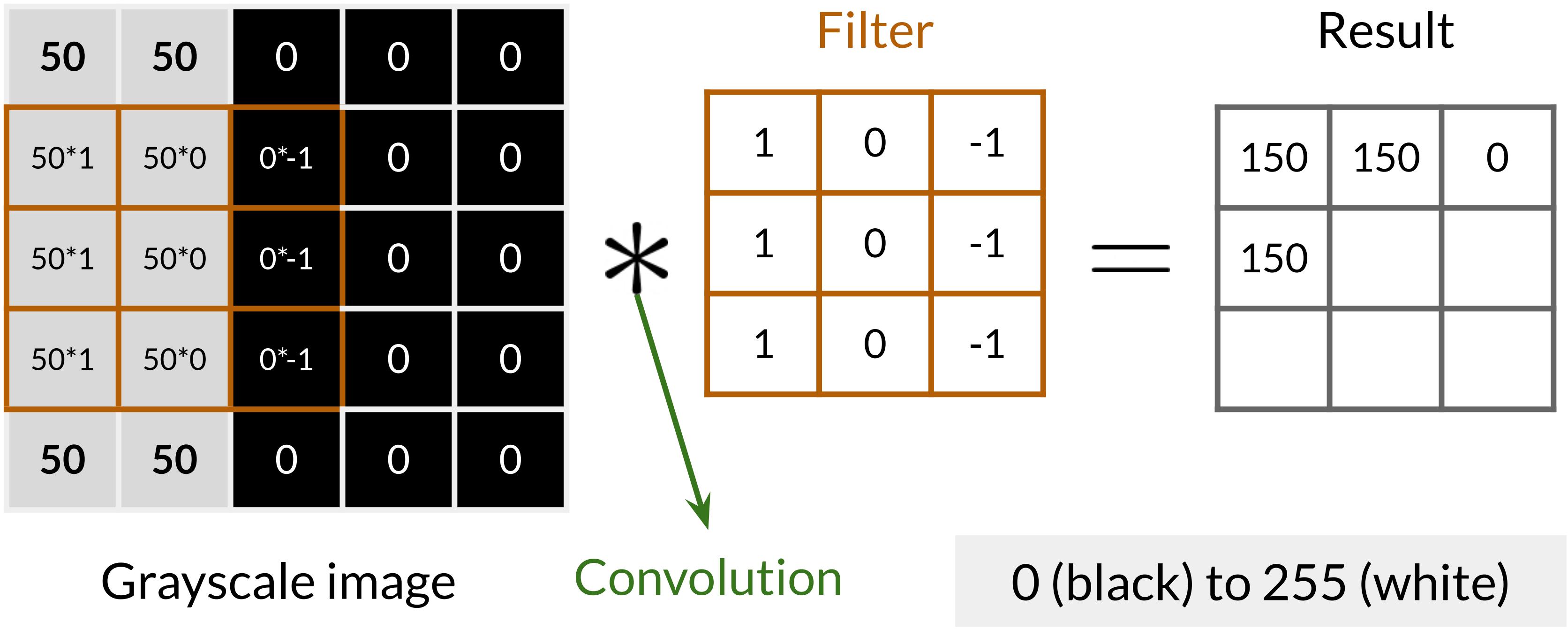
0 (black) to 255 (white)

deeplearning.ai

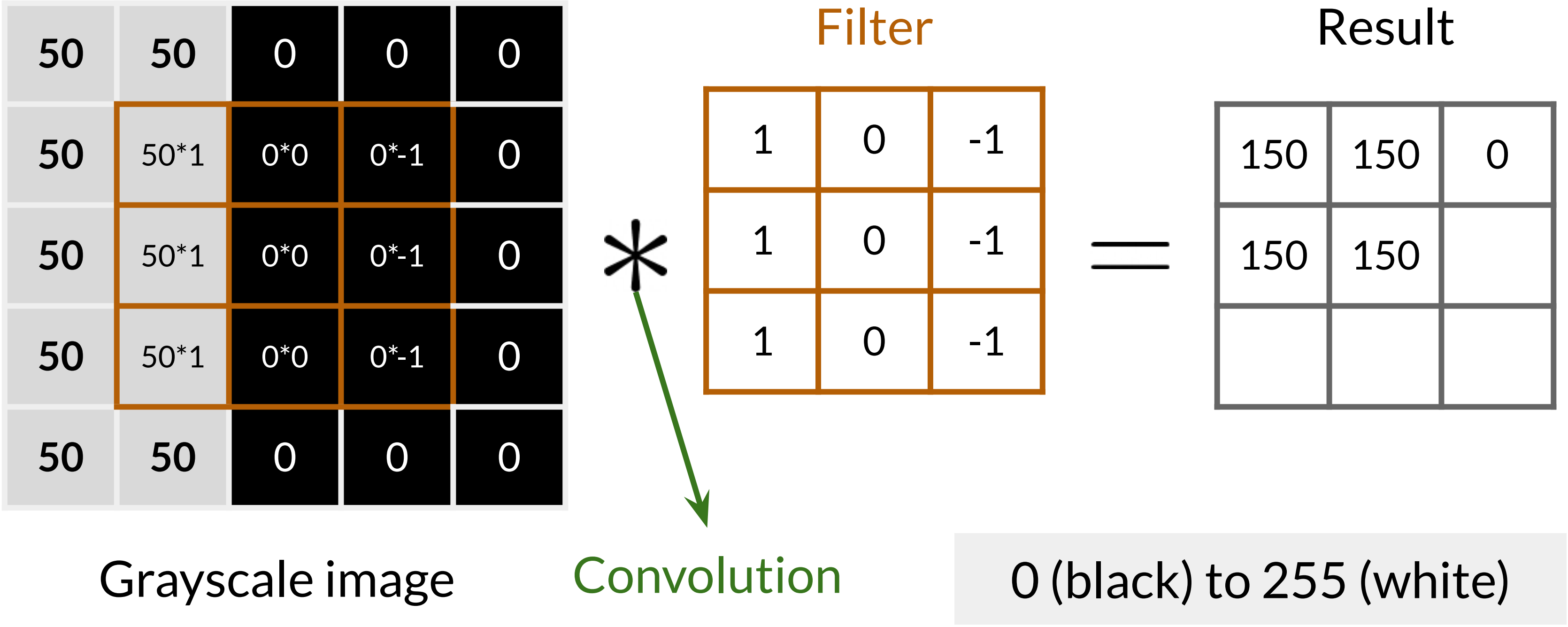
What is a convolution?



What is a convolution?



What is a convolution?



What is a convolution?

50	50	0	0	0
50	50	0	0	0
50	50	0	0	0
50	50	0	0	0
50	50	0	0	0

Grayscale image

*

Filter

1	0	-1
1	0	-1
1	0	-1

=

Result

150	150	0
150	150	0
150	150	0

Convolution

0 (black) to 255 (white)

Summary

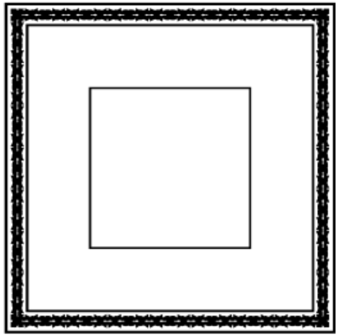
- Convolutions are useful layers for processing images
- They scan the image to detect useful features
- Just element-wise products and sums!

deeplearning.ai

Padding and Stride

Outline

- Padding and stride
- The intuition behind padding



deeplearning.ai

Stride

50	50	0	0	0
50	50	0	0	0
50	50	0	0	0
50	50	0	0	0
50	50	0	0	0

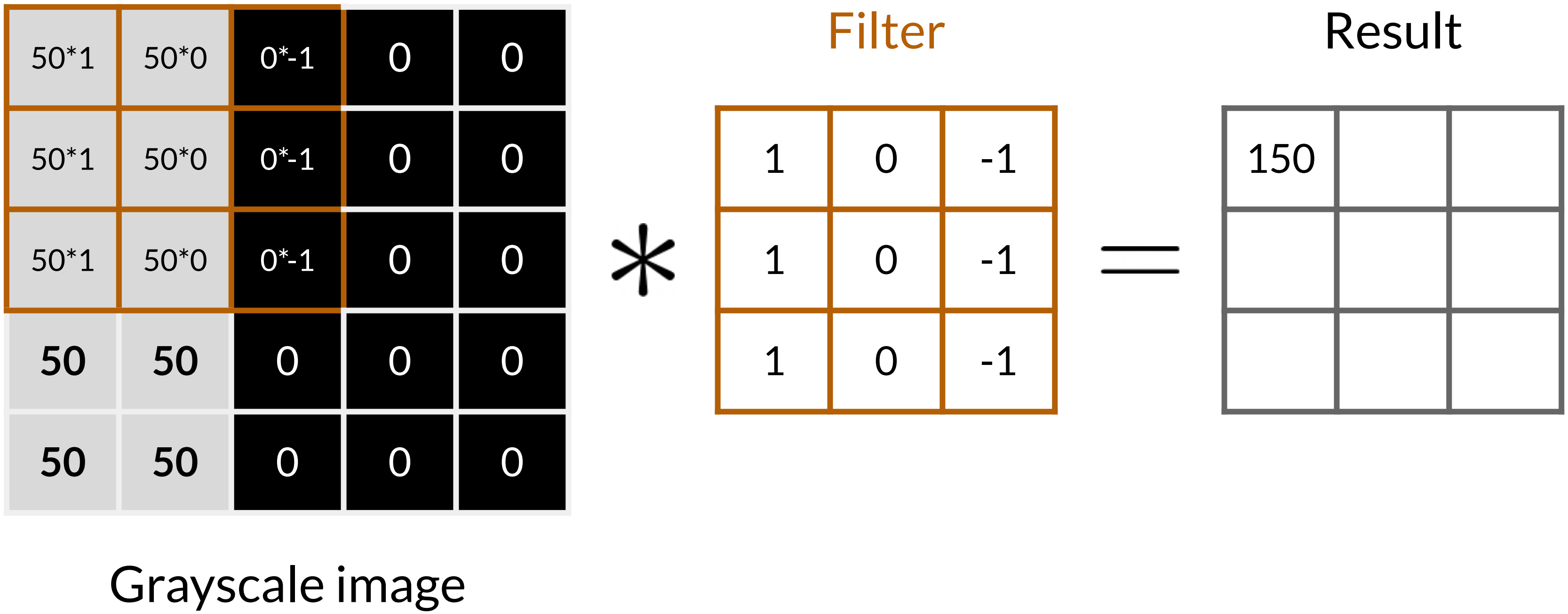
Grayscale image

$*$

Filter

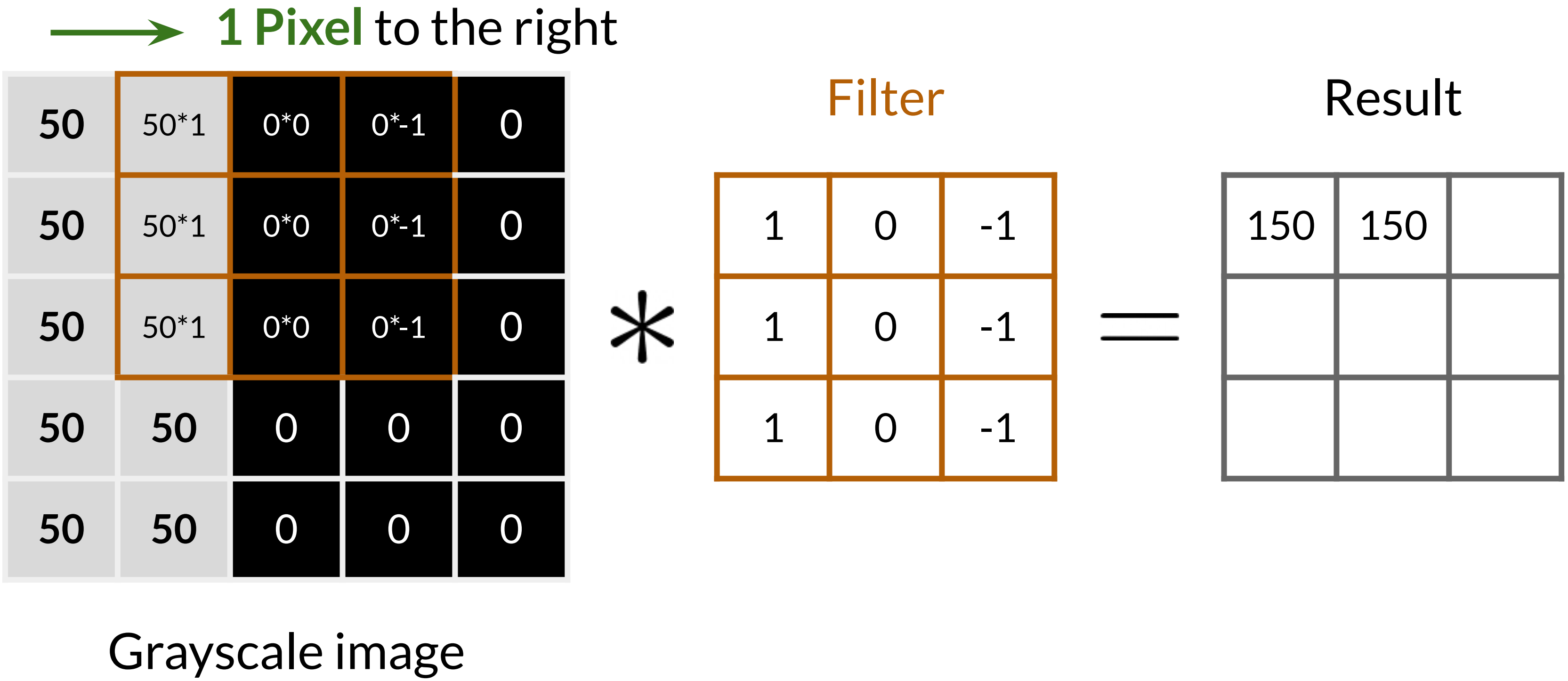
1	0	-1
1	0	-1
1	0	-1

Stride

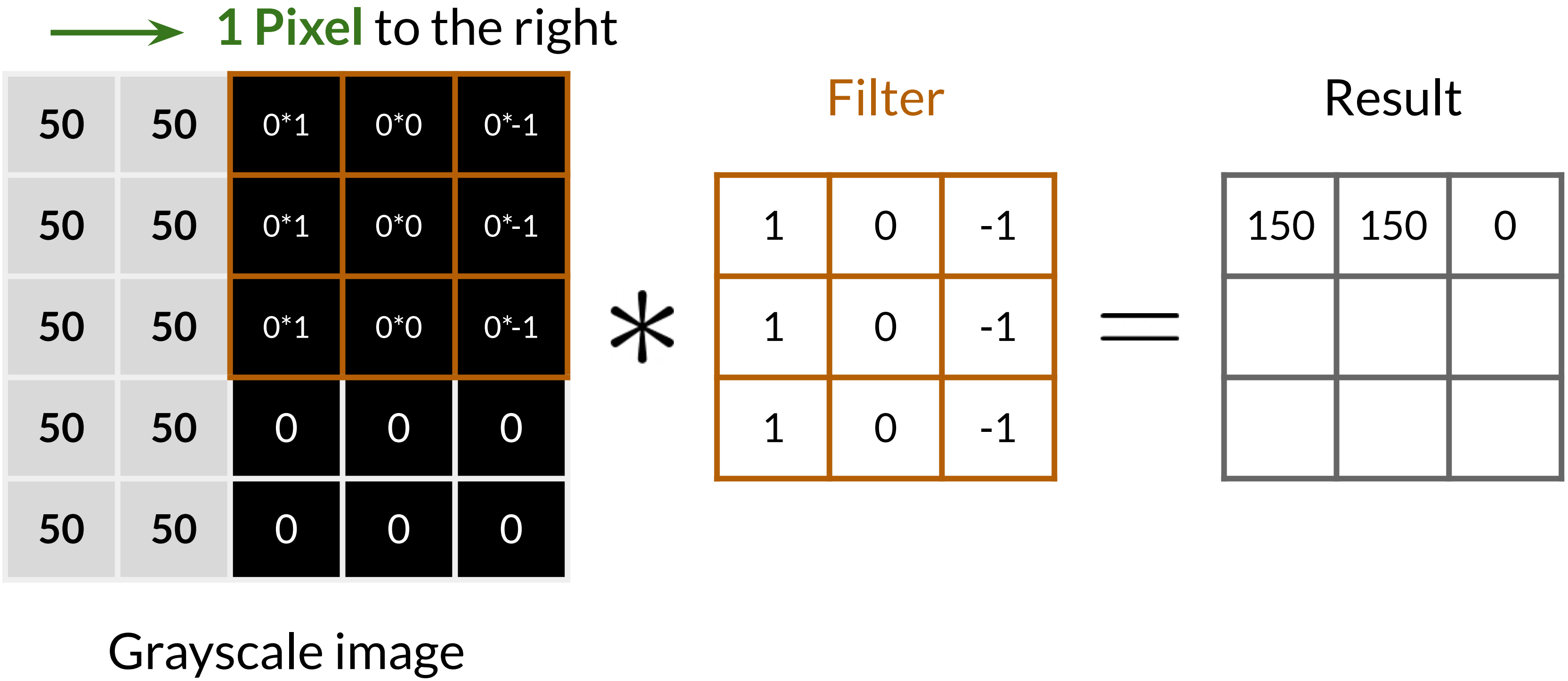


deeplearning.ai

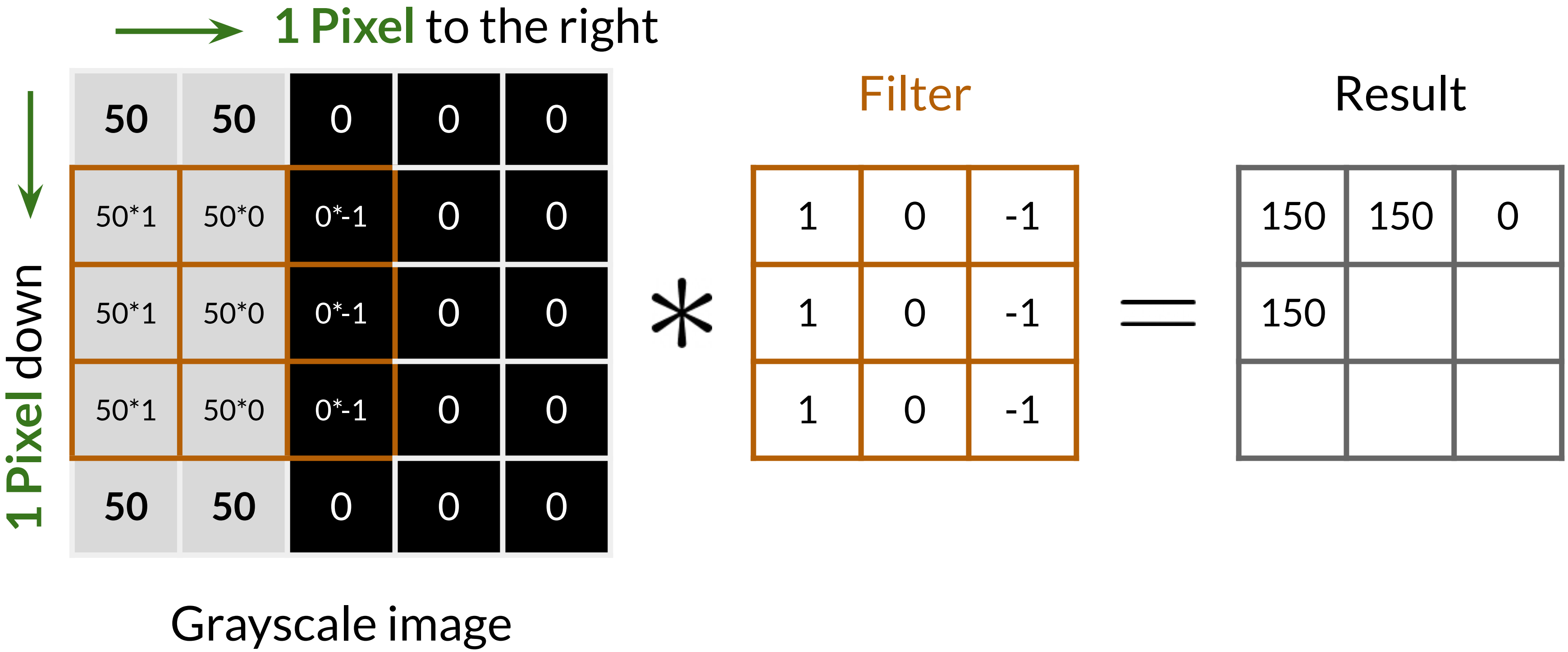
Stride


1 Pixel to the right
50 50*1 0*0 0*-1 0
50 50*1 0*0 0*-1 0
50 50*1 0*0 0*-1 0
50 50 0 0 0
50 50 0 0 0
Grayscale image
Filter
1 0 -1
1 0 -1
1 0 -1
Result
150 150

Stride

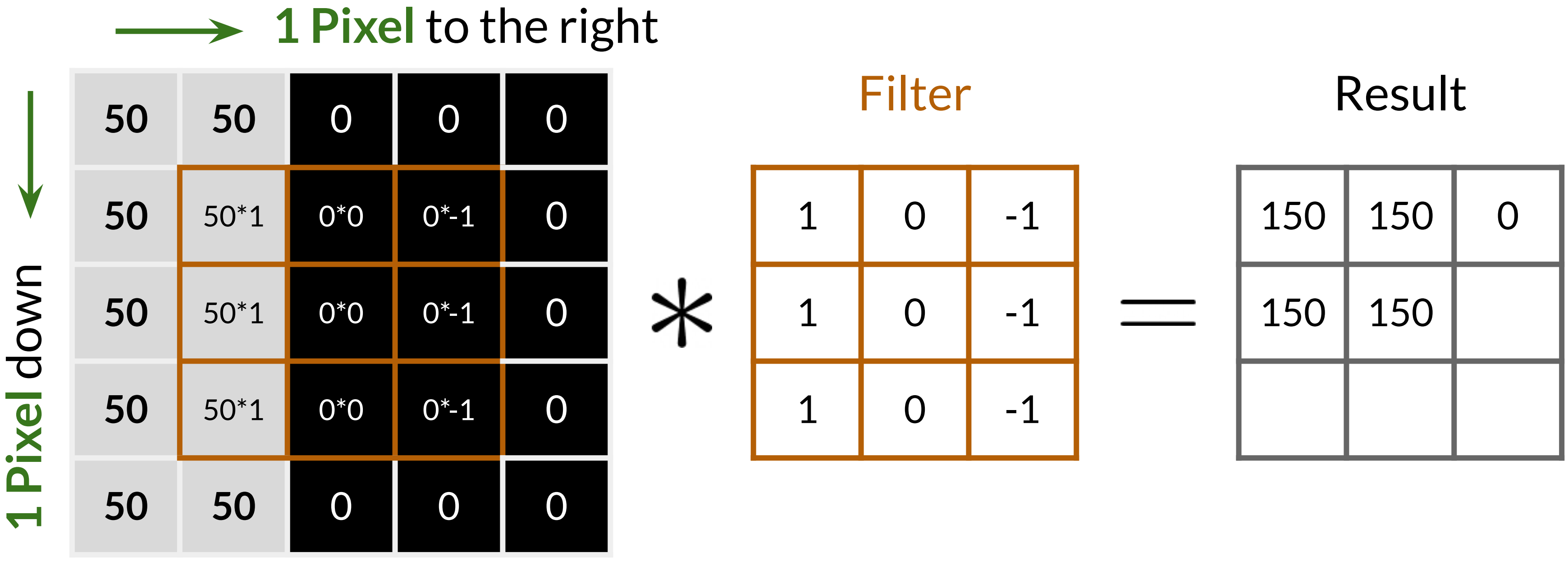

1 Pixel to the right
50
50
0*1
0*0
0*-1
50
50
0*1
0*0
0*-1
50
50
0*1
0*0
0*-1
50
50
0
0
0
50
50
0
0
0
Filter
1
0
-1
1
0
-1
1
0
-1
Result
150
150
0
Grayscale image

Stride



1 Pixel to the right
1 Pixel down
50 50 0 0 0
50*1 50*0 0*-1 0 0
50*1 50*0 0*-1 0 0
50*1 50*0 0*-1 0 0
50 50 0 0 0
Grayscale image
Filter
1 0 -1
1 0 -1
1 0 -1
Result
150 150 0
150

Stride



Grayscale image

Stride

1 Pixel to the right

1 Pixel down

50	50	0	0	0
50	50	0	0	0
50	50	0	0	0
50	50	0	0	0
50	50	0	0	0

Grayscale image

$*$

Filter

1	0	-1
1	0	-1
1	0	-1

$=$

Result

150	150	0
150	150	0
150	150	0

Stride = 1

Stride

2 Pixels to the right

2 Pixels down

50	50	0	0	0
50	50	0	0	0
50	50	0	0	0
50	50	0	0	0
50	50	0	0	0

Grayscale image

$*$

Filter

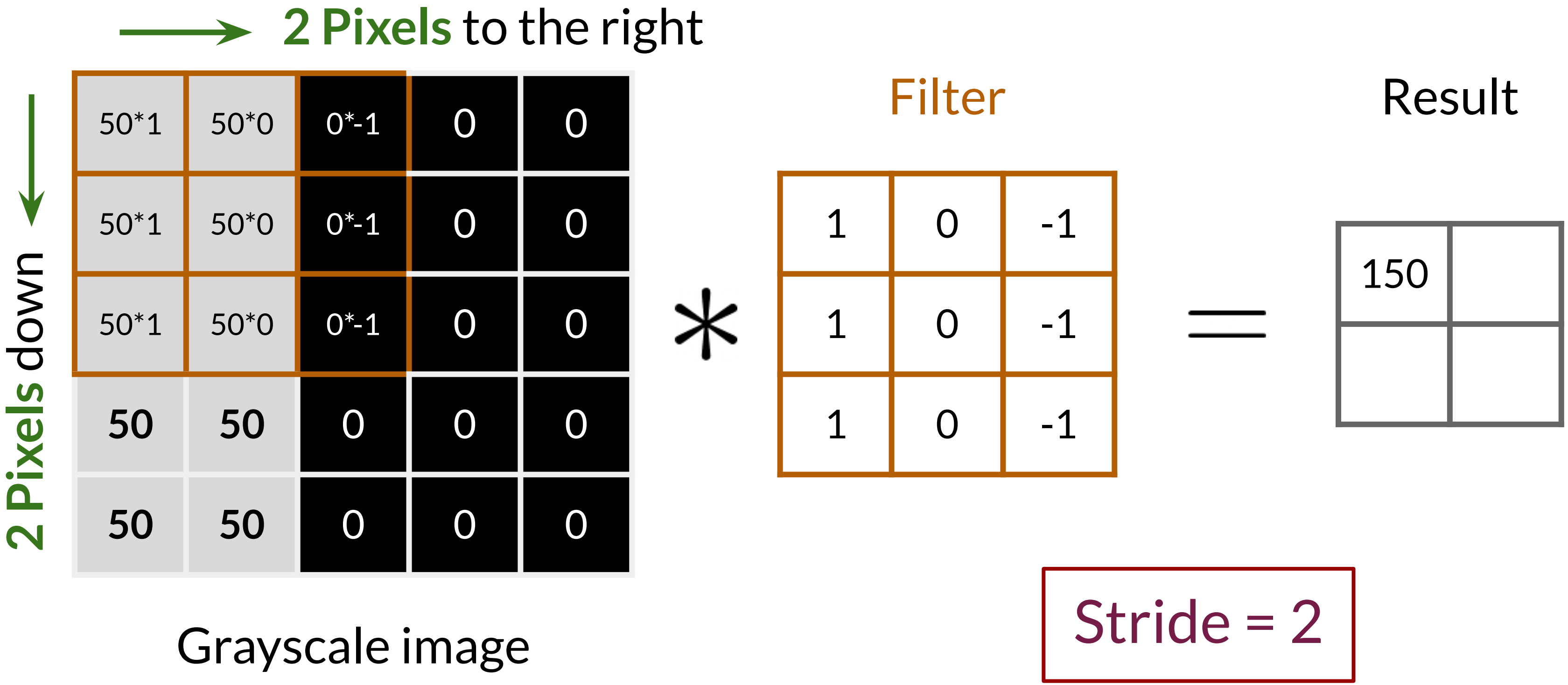
1	0	-1
1	0	-1
1	0	-1

$=$

Result

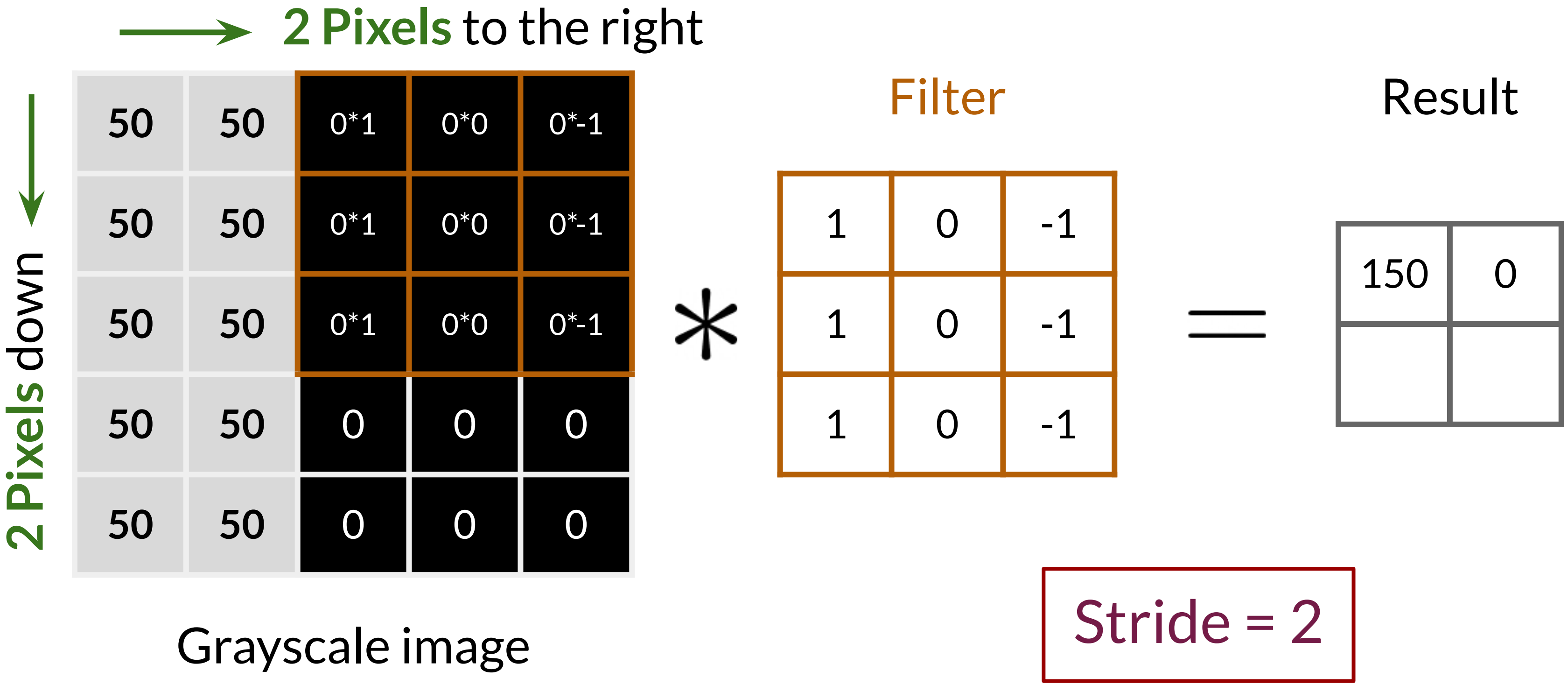
Stride = 2

Stride

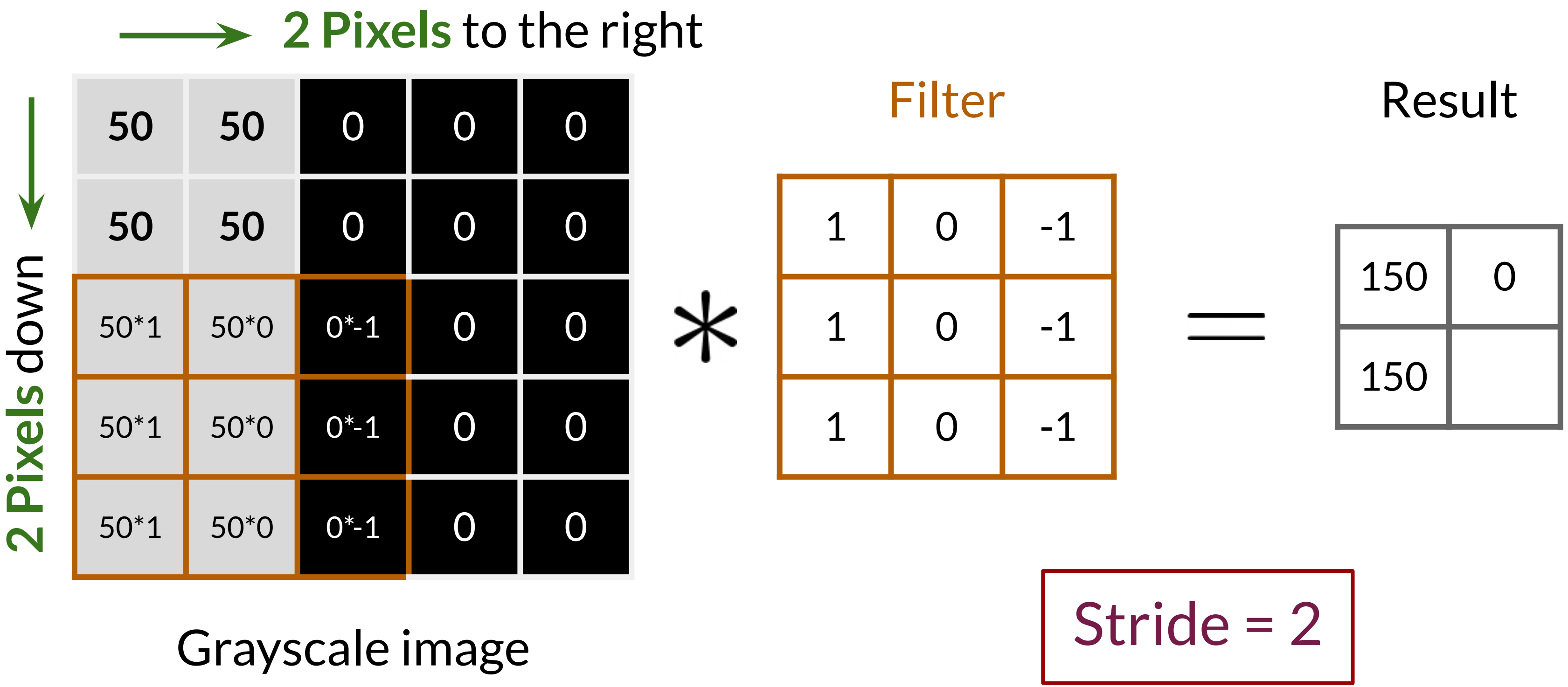


2 Pixels to the right
2 Pixels down
50*1 50*0 0*-1 0 0
50*1 50*0 0*-1 0 0
50*1 50*0 0*-1 0 0
50 50 0 0 0
50 50 0 0 0
Grayscale image
Filter
1 0 -1
1 0 -1
1 0 -1
Result
150
Stride = 2

Stride

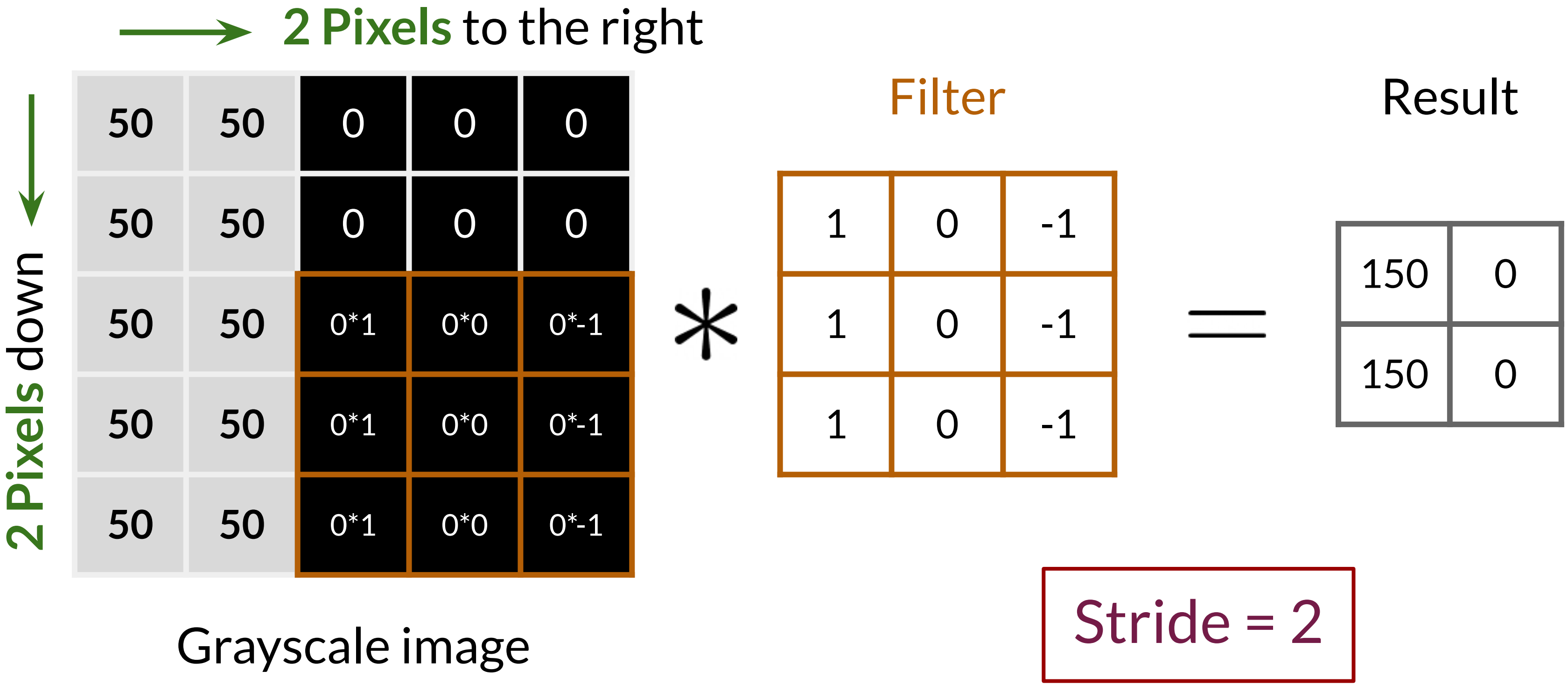

2 Pixels to the right
2 Pixels down
50 50 0*1 0*0 0*-1
50 50 0*1 0*0 0*-1
50 50 0*1 0*0 0*-1
50 50 0 0 0
50 50 0 0 0
Grayscale image
*
Filter
1 0 -1
1 0 -1
1 0 -1
=
Result
150 0
Stride = 2

Stride



2 Pixels to the right
2 Pixels down
50 50 0 0 0
50 50 0 0 0
50*1 50*0 0*-1 0 0
50*1 50*0 0*-1 0 0
50*1 50*0 0*-1 0 0
Grayscale image
Filter
1 0 -1
1 0 -1
1 0 -1
Result
150 0
150
Stride = 2

Stride


2 Pixels to the right
2 Pixels down
50 50 0 0 0
50 50 0 0 0
50 50 0*1 0*0 0*-1
50 50 0*1 0*0 0*-1
50 50 0*1 0*0 0*-1
Grayscale image
Filter
1 0 -1
1 0 -1
1 0 -1
Result
150 0
150 0
Stride = 2

Stride

2 **Pixels** to the right

2 **Pixels** down

50	50	0	0	0
50	50	0	0	0
50	50	0	0	0
50	50	0	0	0
50	50	0	0	0

Grayscale image

$*$

Filter

1	0	-1
1	0	-1
1	0	-1

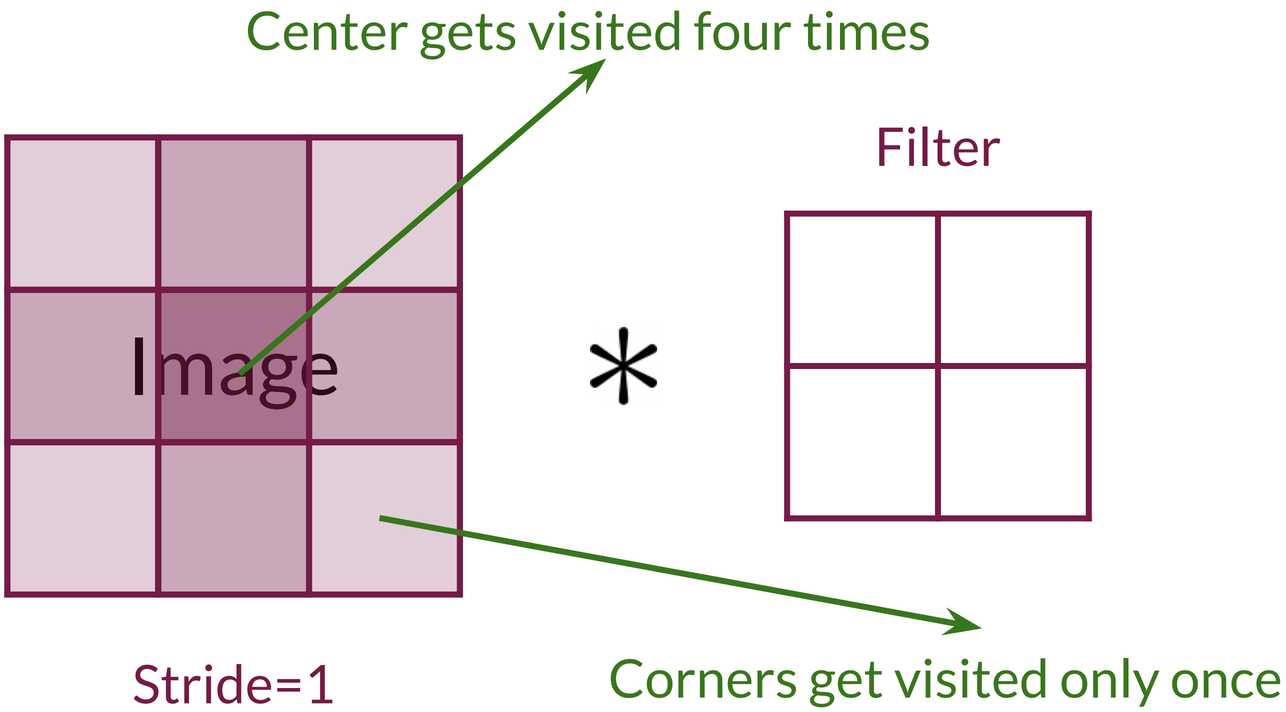
$=$

Result

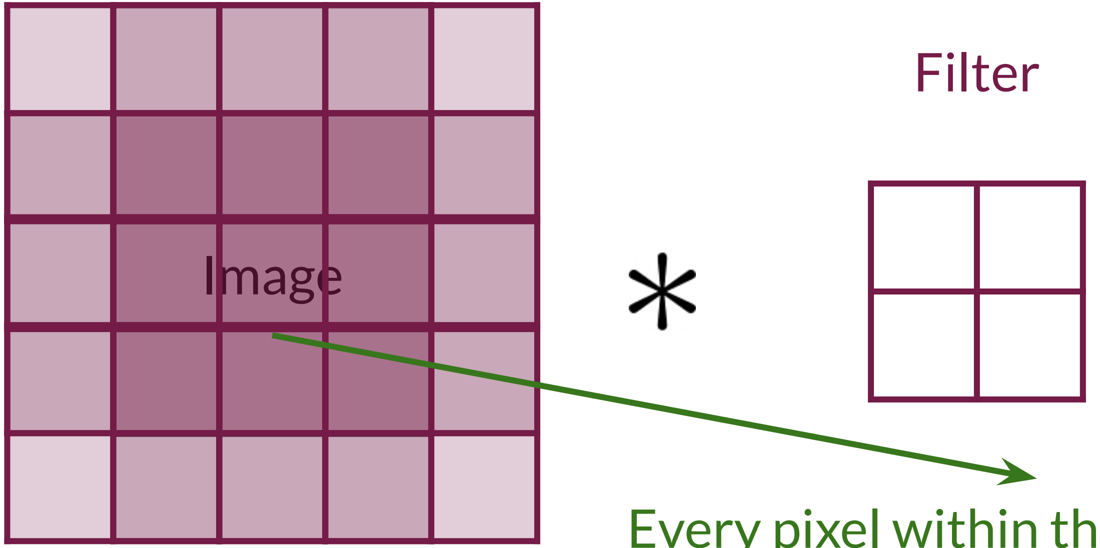
150	0
150	0

Stride = 2

Padding


Center gets visited four times
Filter
Image
*
Stride=1
Corners get visited only once

Padding



Every pixel within the image gets visited the same number of times

Summary

- Stride determines how the filter scans the image
- Padding is like a frame on the image
- Padding gives similar importance to the edges and the center

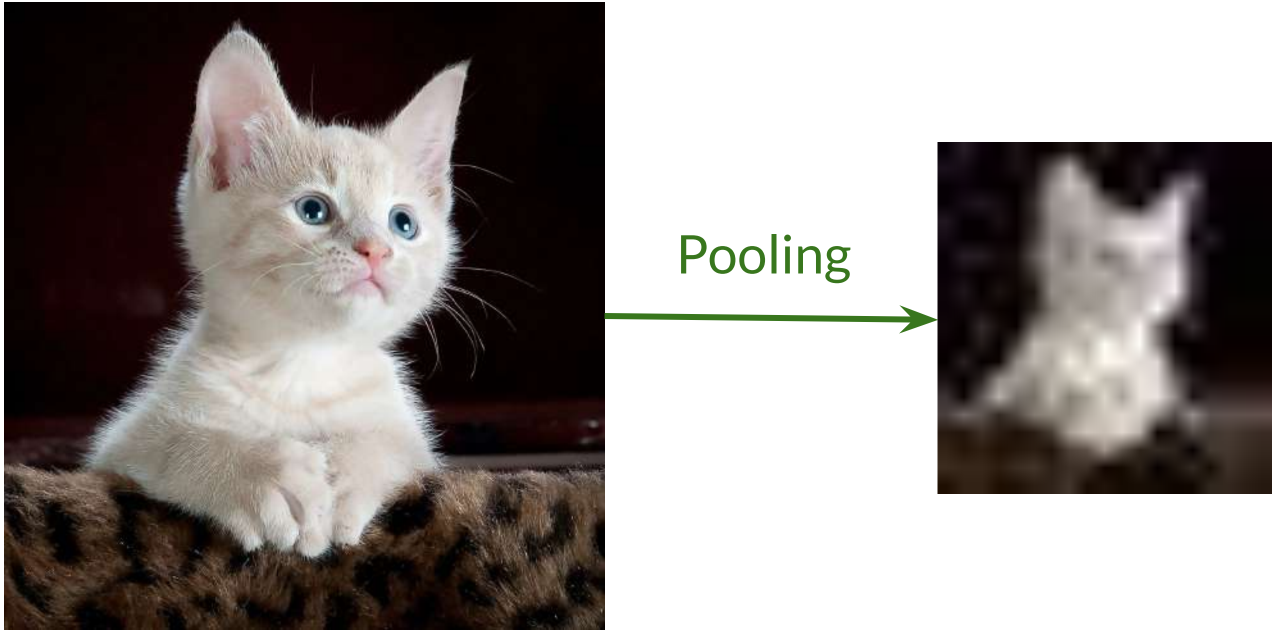
deeplearning.ai

Pooling and Upsampling

Outline

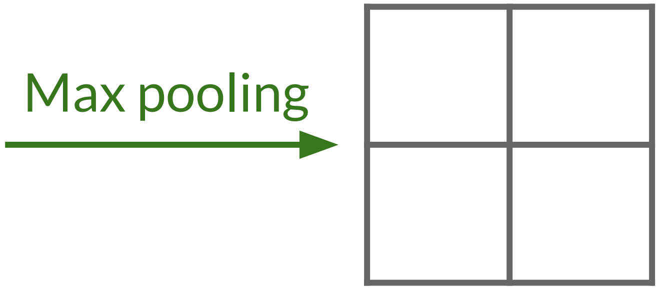
- Pooling
- Upsampling and its relation to pooling

Pooling



Max Pooling

0	20	30	45
8	1	23	43
0	4	40	20
10	6	43	42

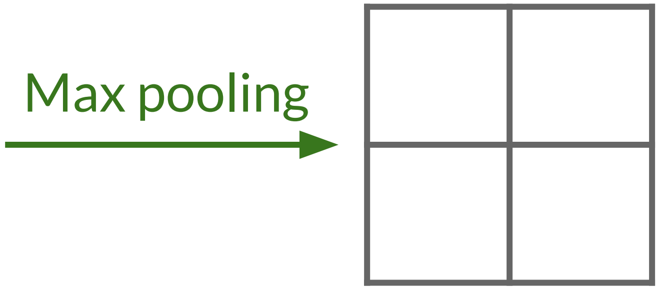


4x4 input to 2x2 output

Max Pooling

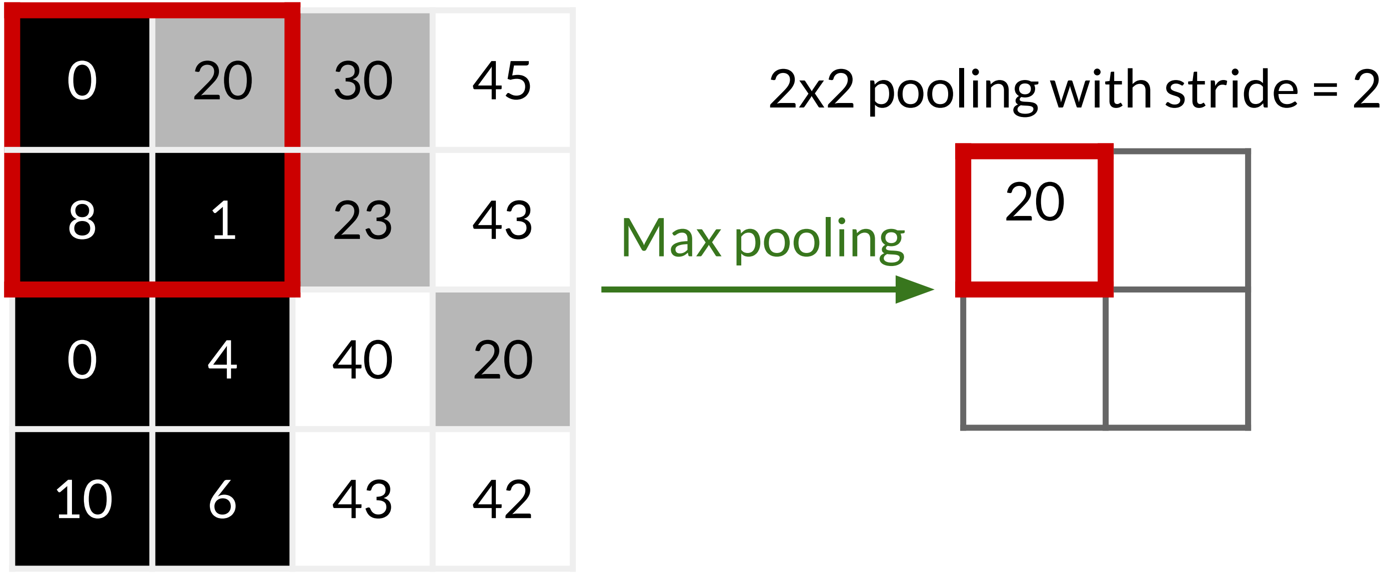
0	20	30	45
8	1	23	43
0	4	40	20
10	6	43	42

2x2 pooling with stride = 2



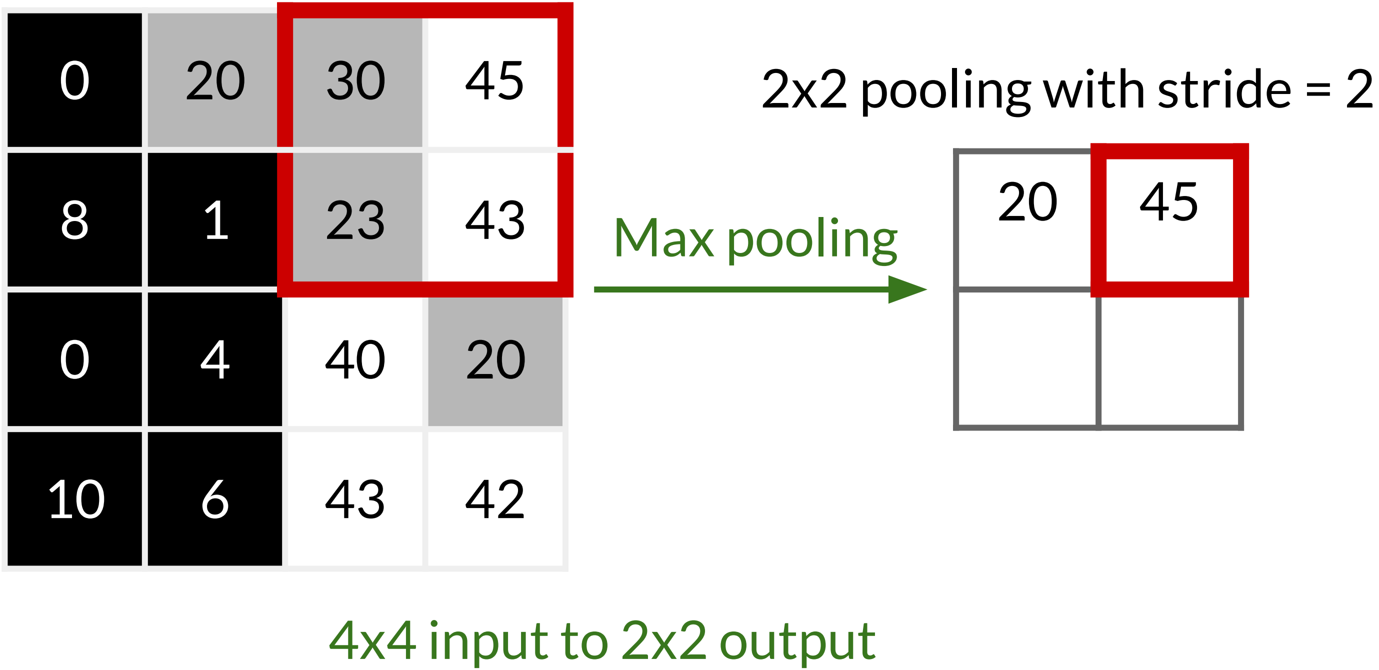
4x4 input to 2x2 output

Max Pooling



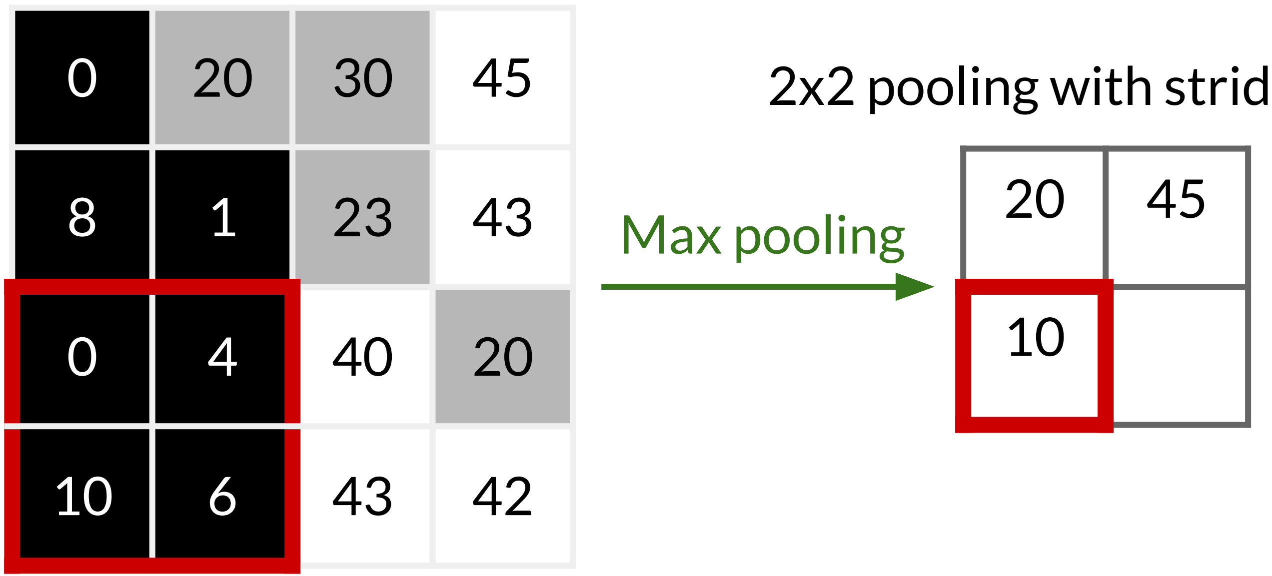
4x4 input to 2x2 output

Max Pooling


0
20
30
45
8
1
23
43
0
4
40
20
10
6
43
42
2x2 pooling with stride = 2
Max pooling
20
45
4x4 input to 2x2 output

Max Pooling

2x2 pooling with stride = 2



4x4 input to 2x2 output

Max Pooling

0	20	30	45
8	1	23	43
0	4	40	20
10	6	43	42

2x2 pooling with stride = 2

Max pooling →

20	45
10	43

4x4 input to 2x2 output

Max Pooling

0	20	30	45
8	1	23	43
0	4	40	20
10	6	43	42

Max pooling

2x2 pooling with stride = 2

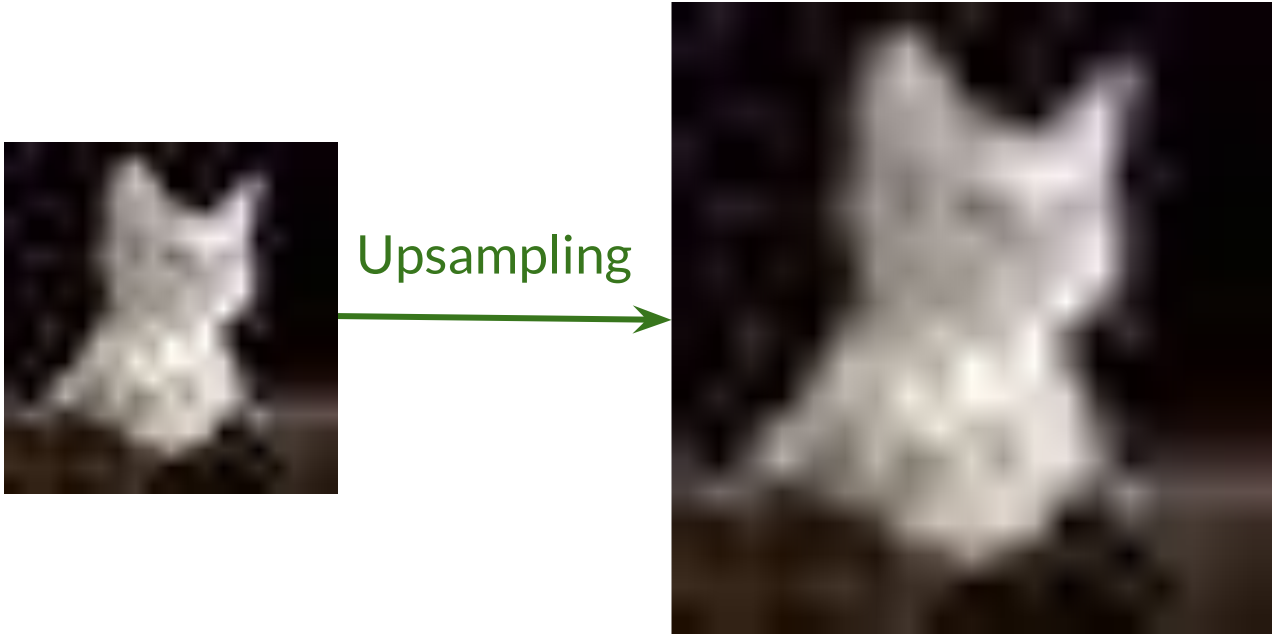
20	45
10	43

4x4 input to 2x2 output

Other types include:

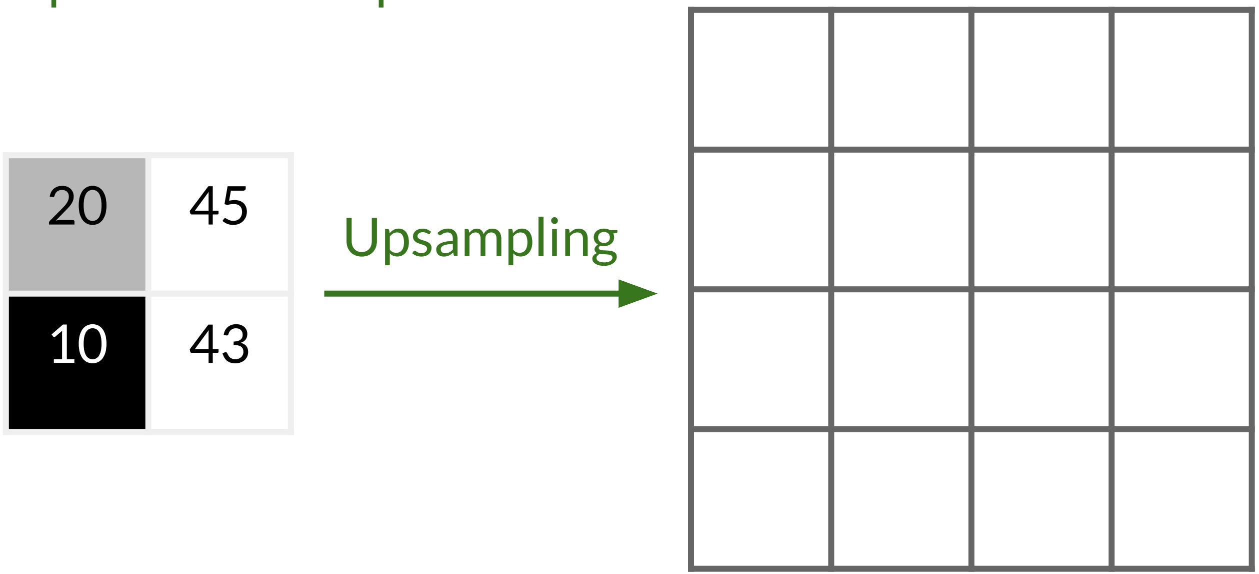
1. Average pooling
2. Min pooling

Upsampling



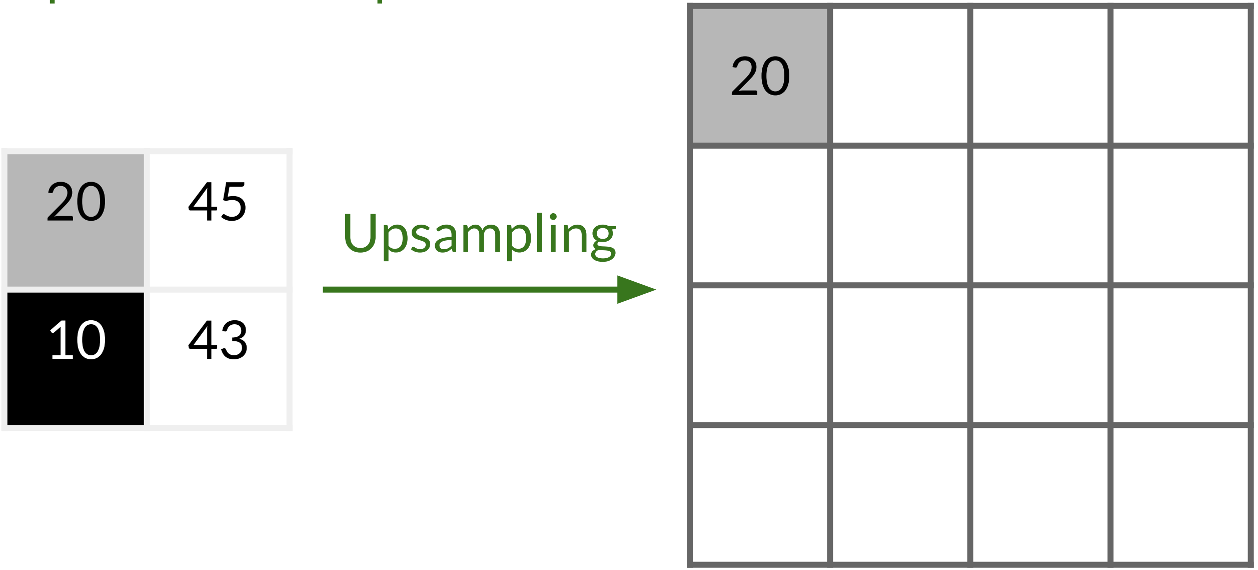
Upsampling: Nearest Neighbors

2x2 input to 4x4 output



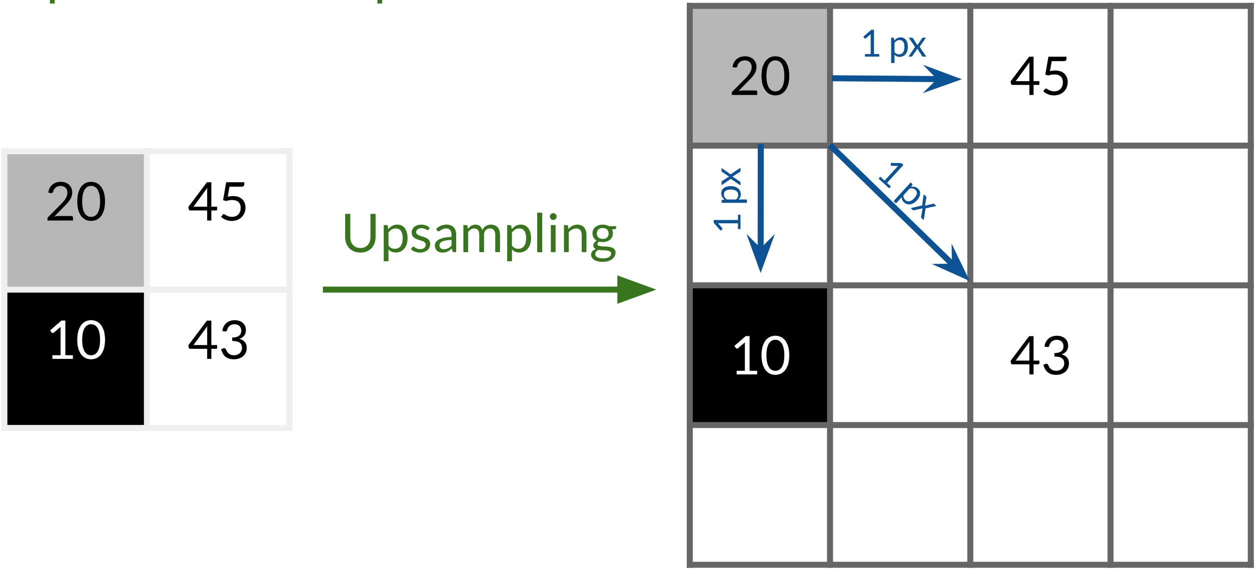
Upsampling: Nearest Neighbors

2x2 input to 4x4 output



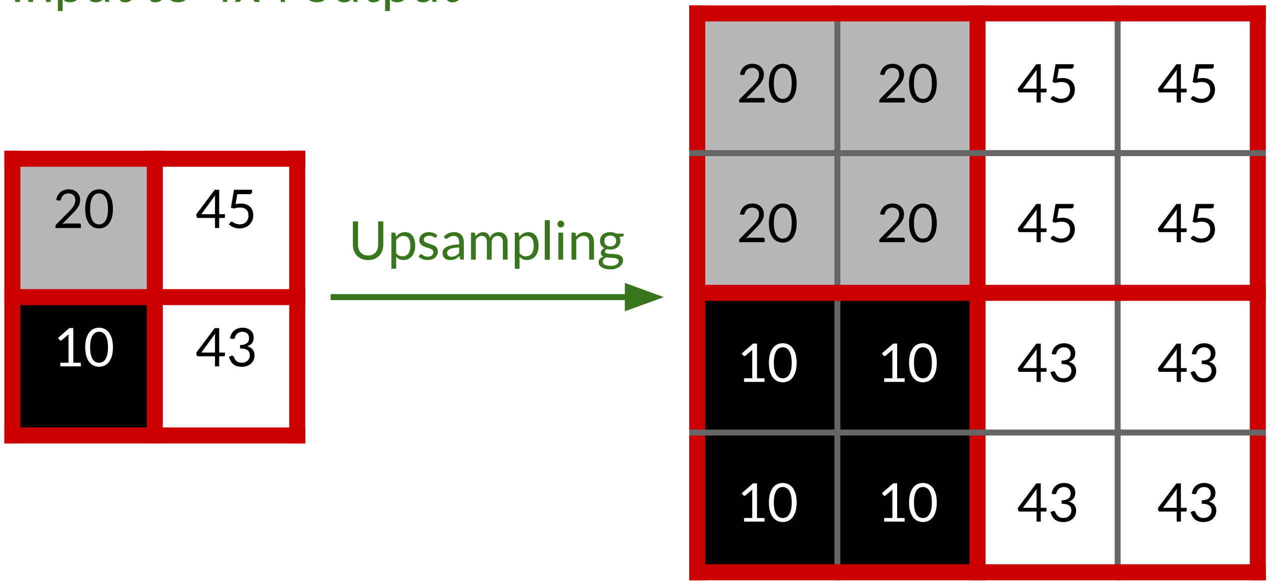
Upsampling: Nearest Neighbors

2x2 input to 4x4 output



Upsampling: Nearest Neighbors

2x2 input to 4x4 output



Upsampling: Nearest Neighbors

2x2 input to 4x4 output

20	45
10	43

Upsampling →

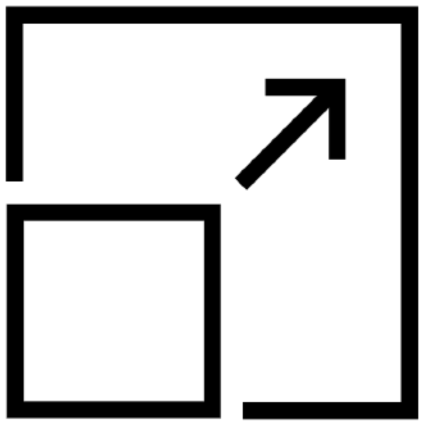
20	20	45	45
20	20	45	45
10	10	43	43
10	10	43	43

Other types include:

1. Linear interpolation
2. Bi-linear interpolation

Summary

- Pooling reduces the size of the input
- Upsampling increases the size of the input
- No learnable parameters!

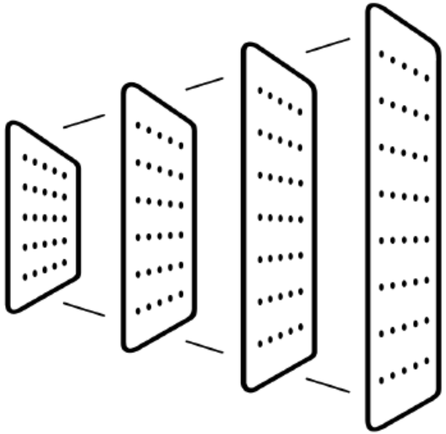


deeplearning.ai

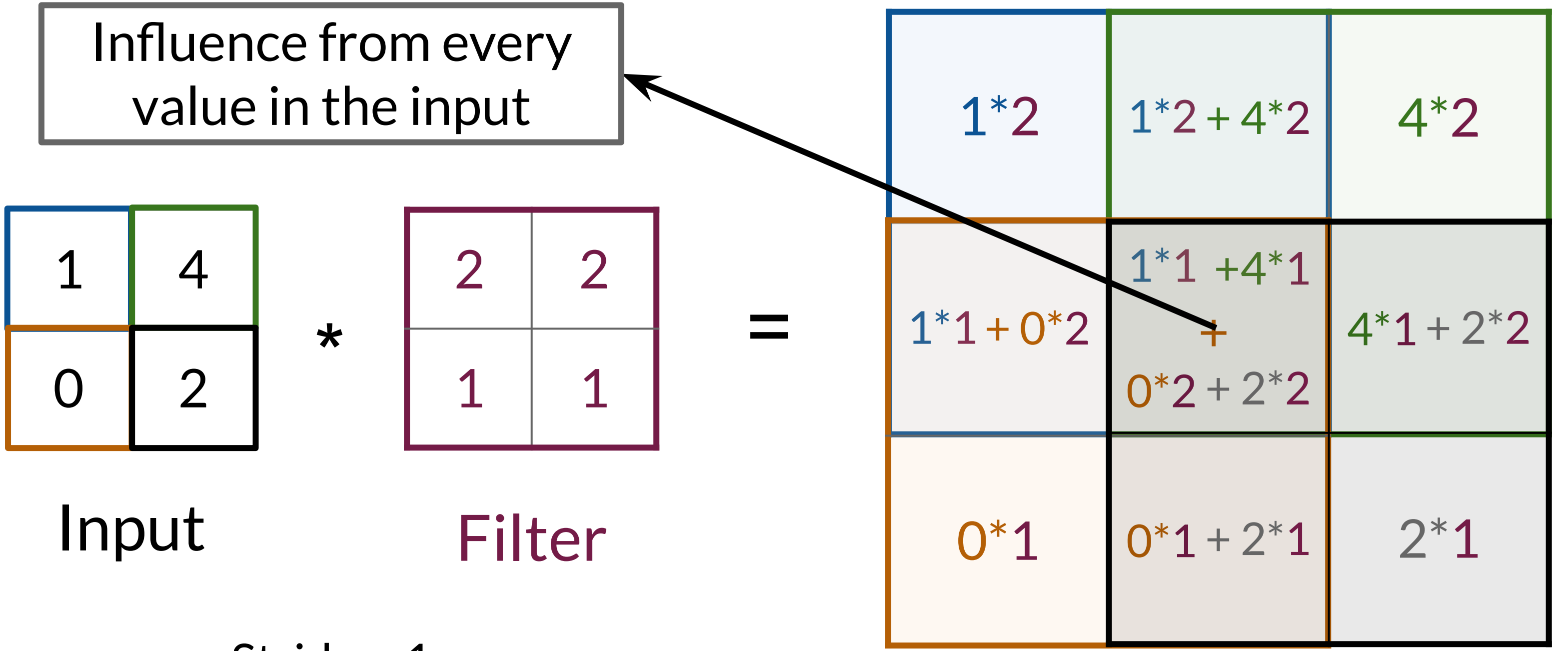
Transposed Convolutions

Outline

- Transposed convolutions as an upsampling technique
- Issues with transposed convolutions

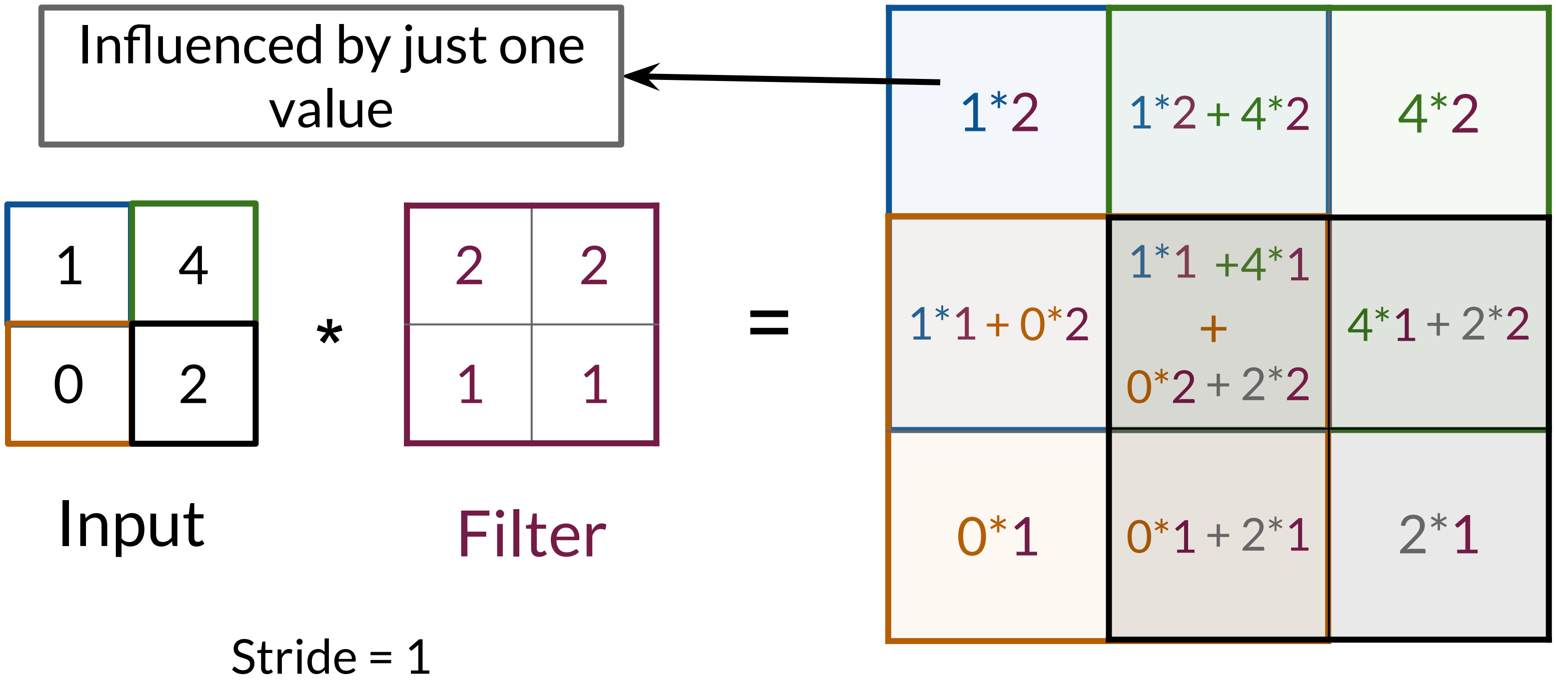


Transposed Convolution

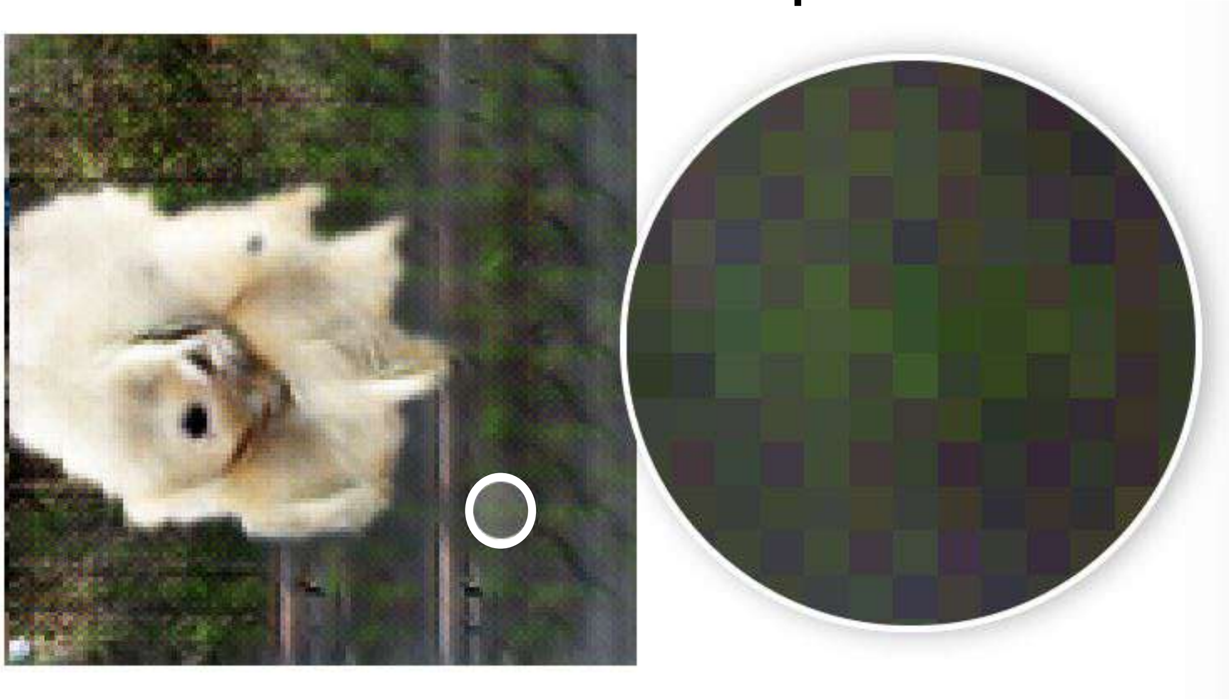


Stride = 1

Transposed Convolution



The Problems with Transposed Convolution

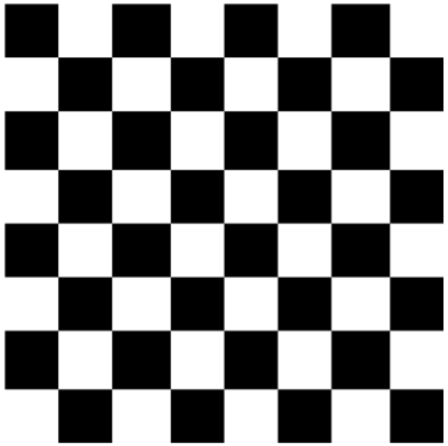


Checkerboard Pattern

Available from: http://doi.org/10.23915/distill.00003

Summary

- Transposed convolutions upsample
- They have learnable parameters
- Problem: results have a checkerboard pattern



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.

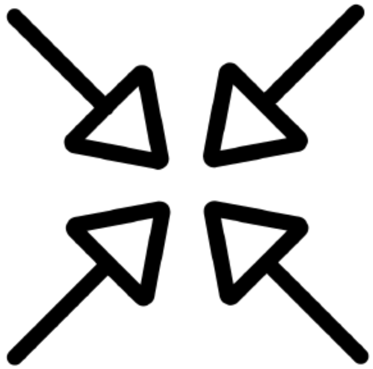
For the rest of the details of the license, see https://creativecommons.org/licenses/by-sa/2.0/legalcode

deeplearning.ai

Mode Collapse

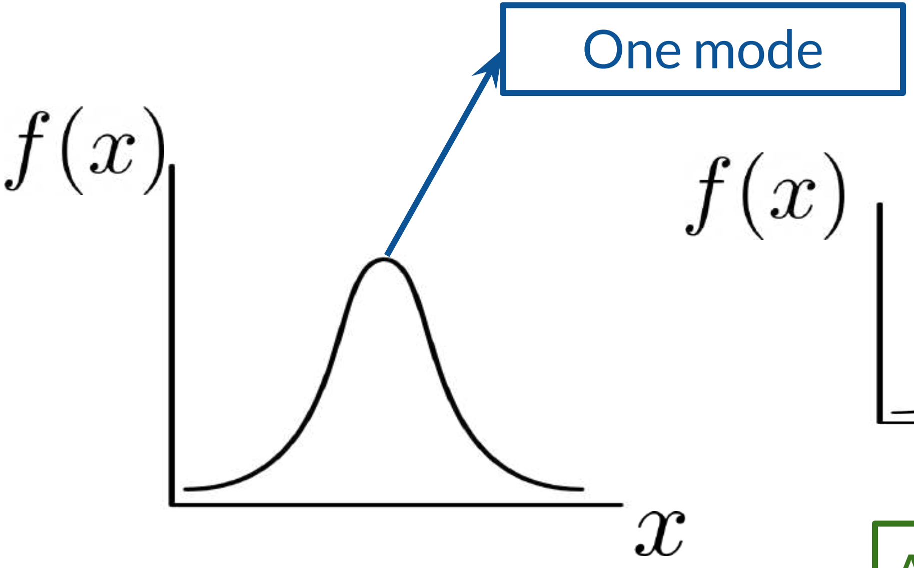
Outline

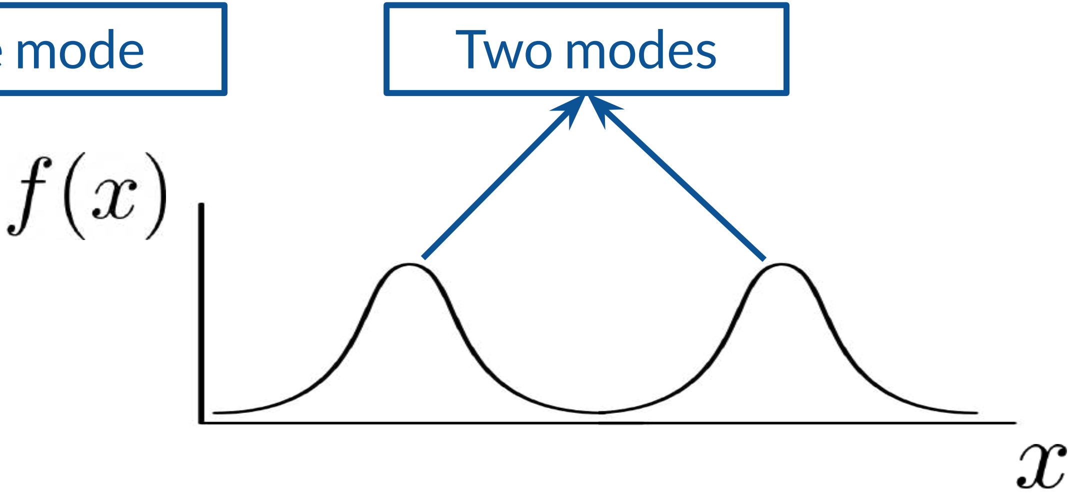
- Modes in distributions
- Mode collapse in GANs
- Intuition behind it during training



deeplearning.ai

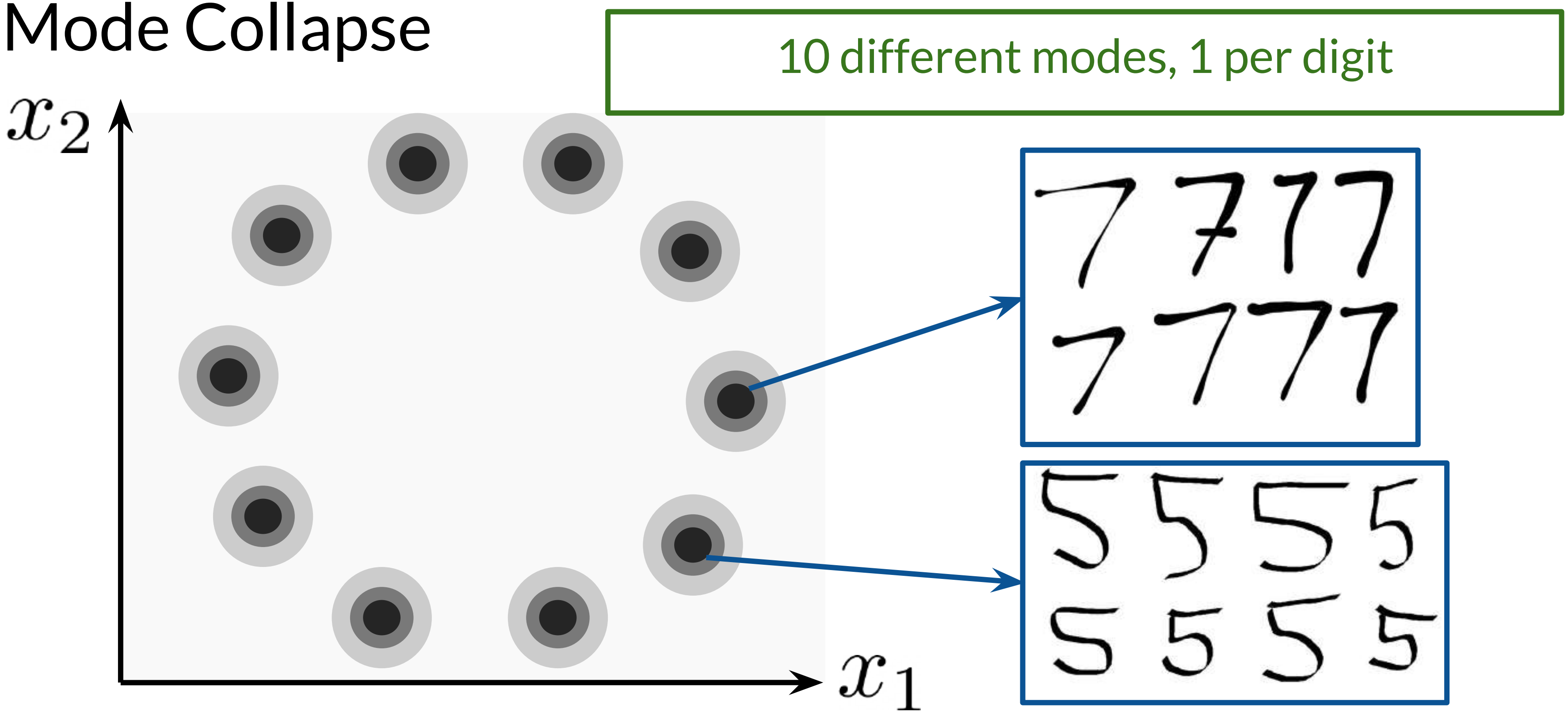
Mode Collapse





Any peak on the density function is a mode!

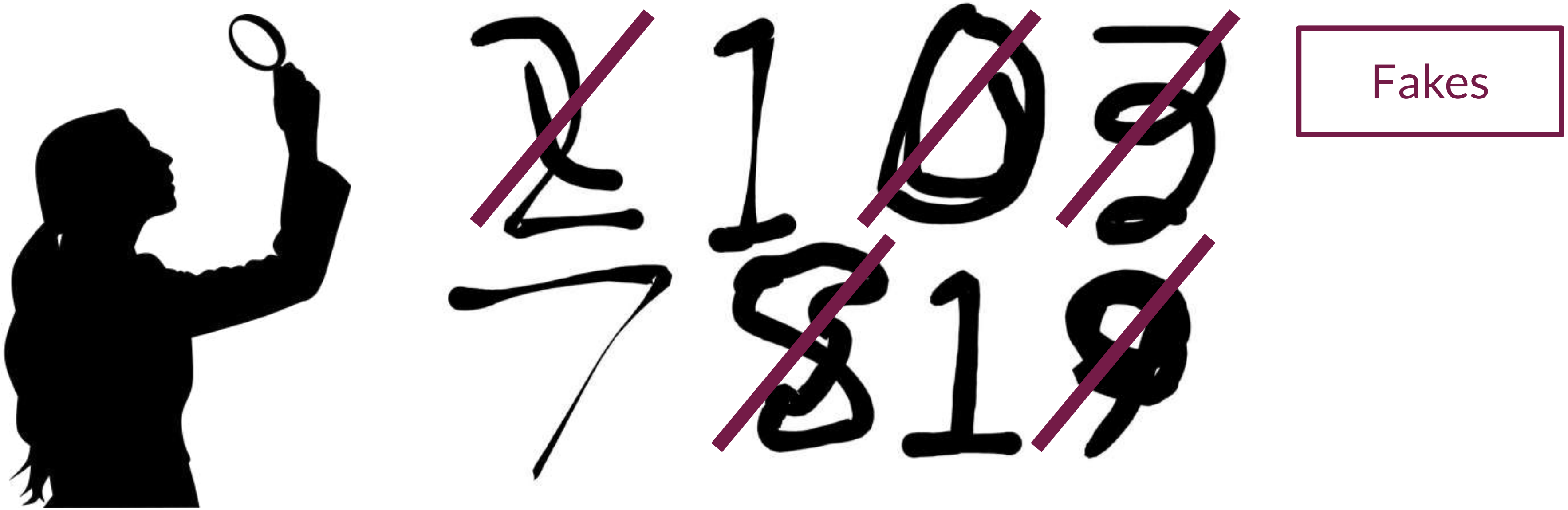
Mode Collapse



Mode Collapse

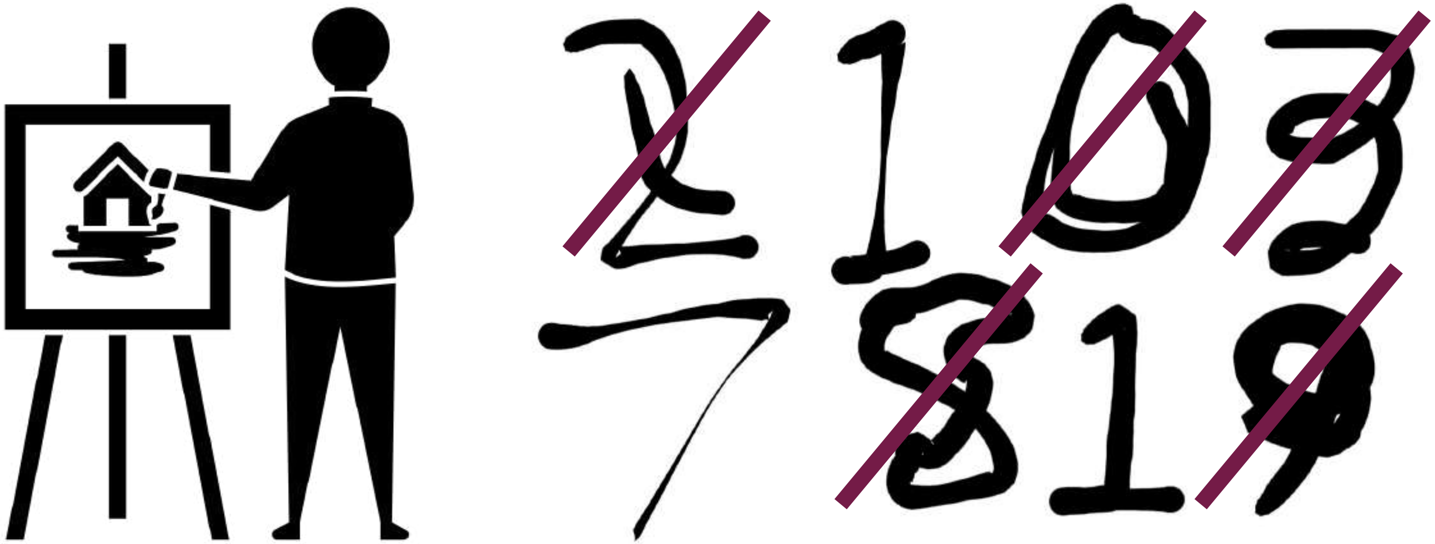
Discriminator

Mode Collapse



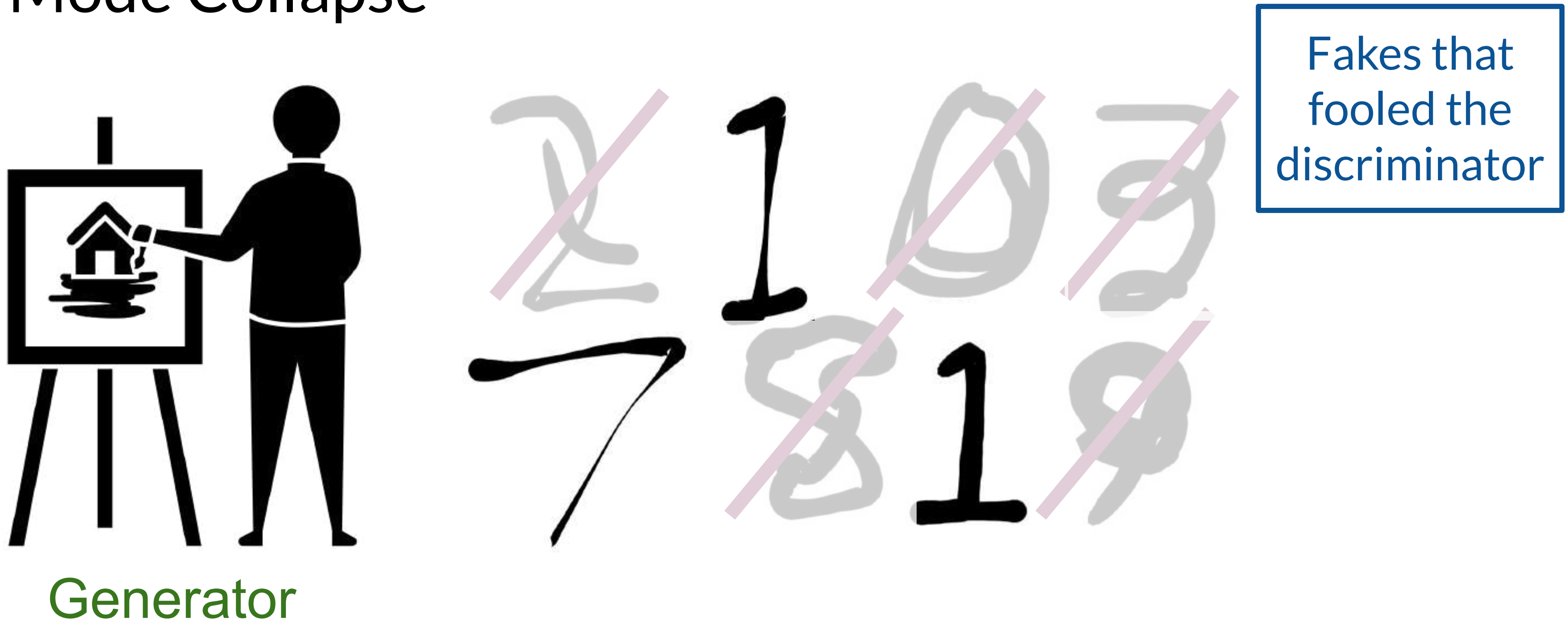
Discriminator

Mode Collapse



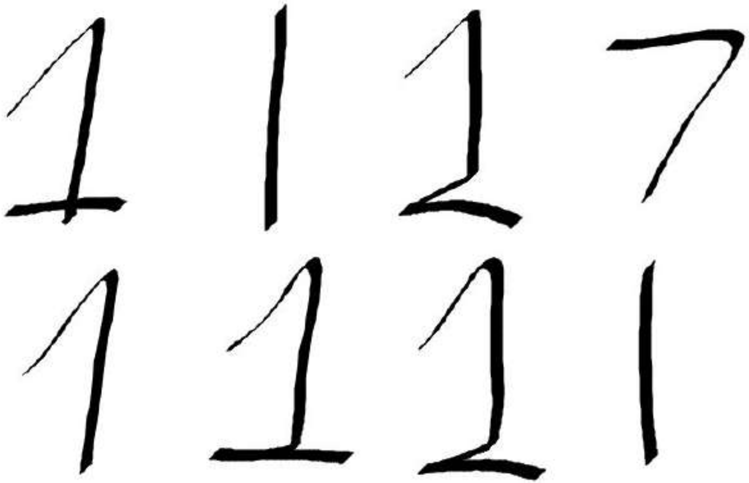
Generator

Mode Collapse

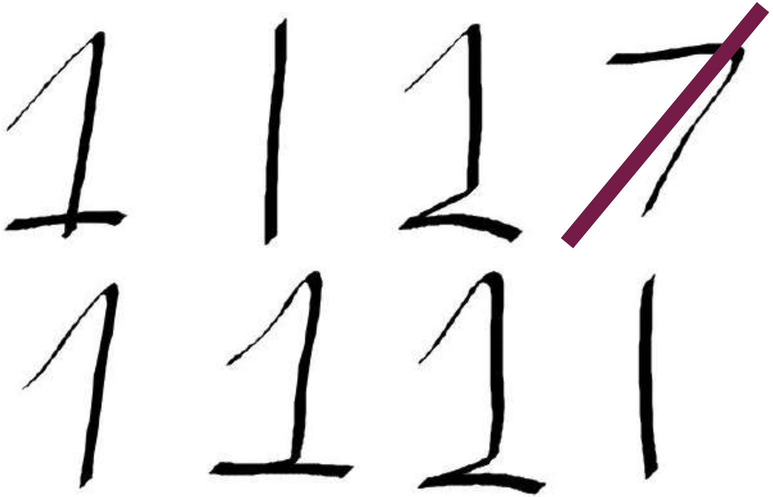


Mode Collapse

Generator



Mode Collapse

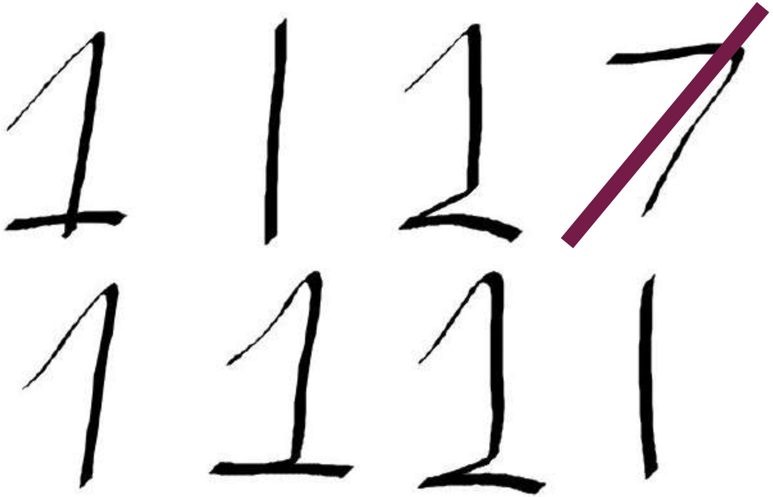


Fakes

Discriminator

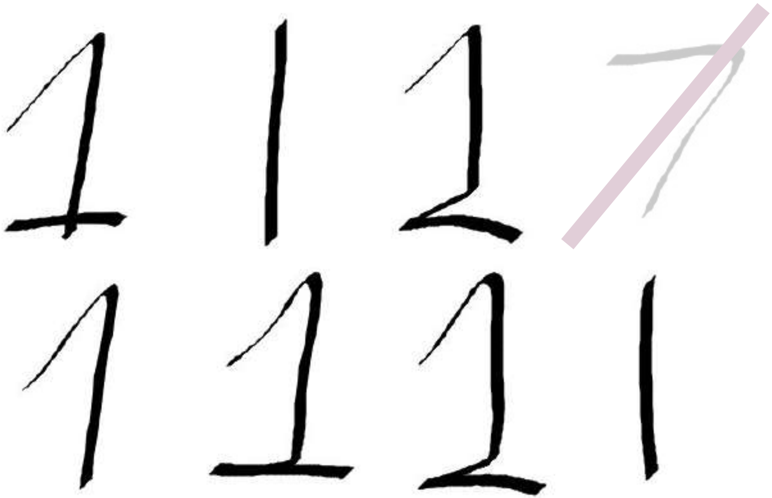
Mode Collapse

Generator



Mode Collapse

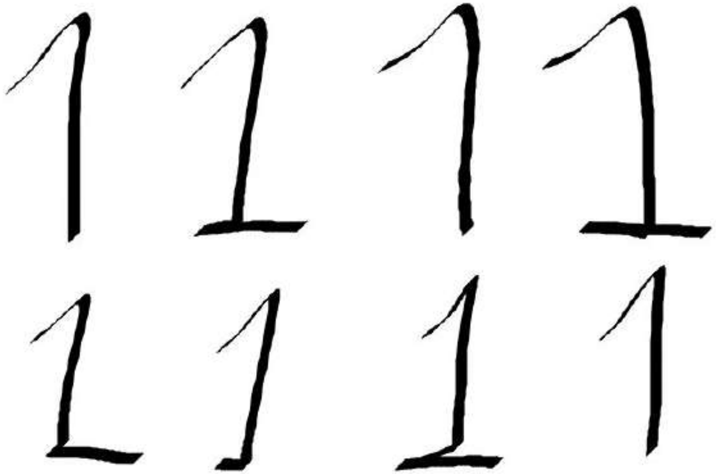
Generator



Fakes that fooled the discriminator

Mode Collapse

Generator



Summary

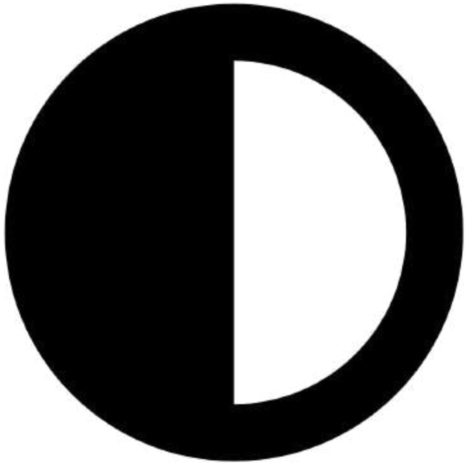
- Modes are peaks in the distribution of features
- Typical with real-world datasets
- Mode collapse happens when the generator gets stuck in one mode

deeplearning.ai

Problem with BCE Loss

Outline

- BCE Loss and the end objective in GANs
- Problem with BCE Loss



BCE Loss in GANs

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

Prediction

Label

Features

Parameters

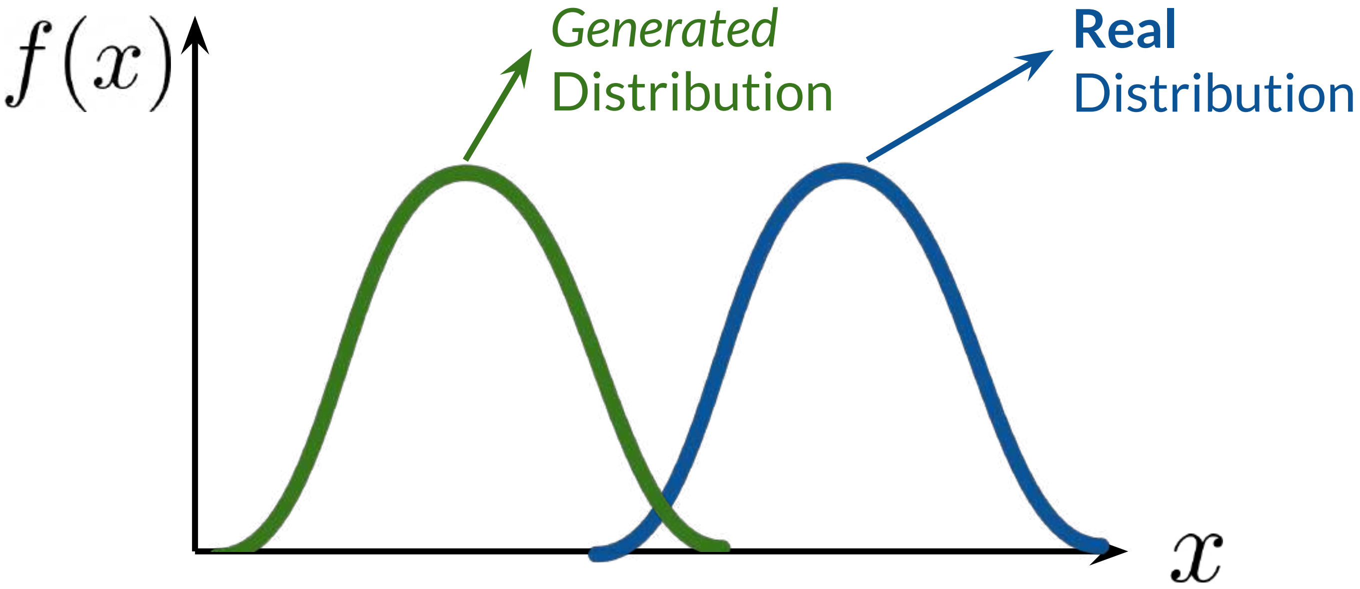
Generator

Maximize cost

Discriminator

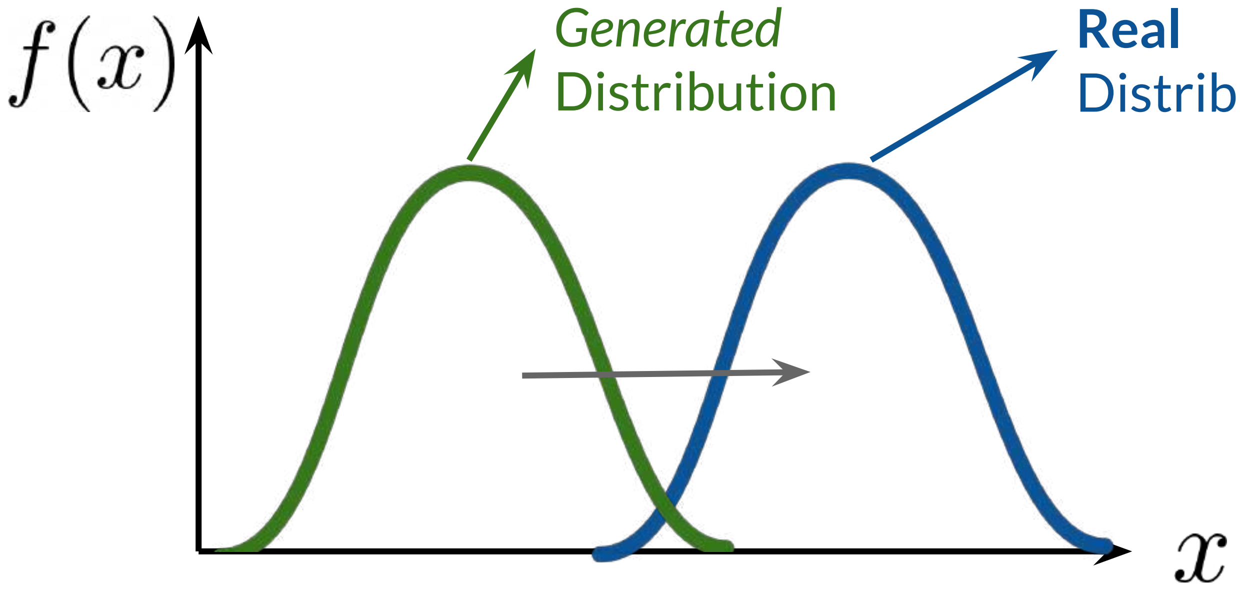
Minimize cost

Objective in GANs



Objective in GANs

Make the generated and real distributions look similar



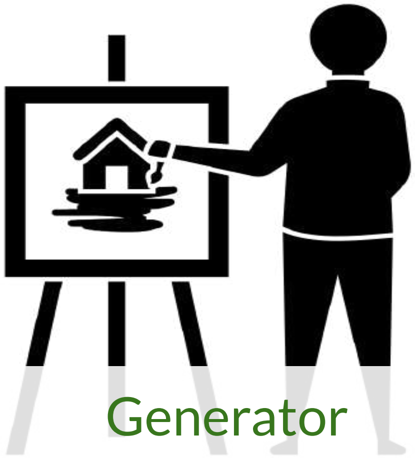
BCE Loss in GANs

Criticizing is more straightforward

Discriminator

Single output

Easier to train than the generator



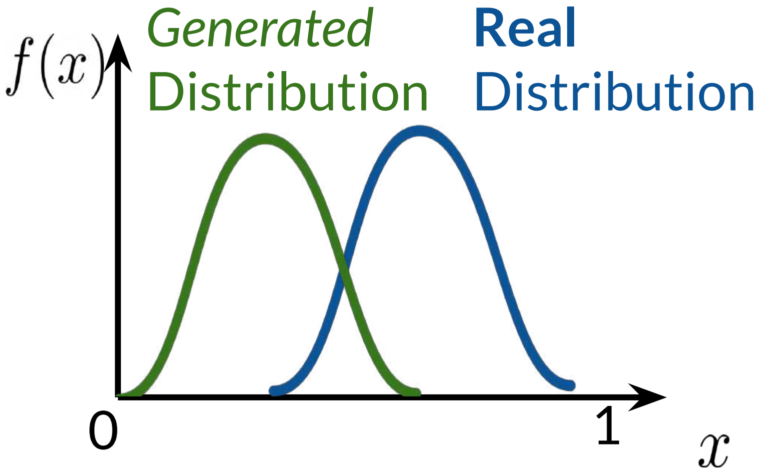
Generator

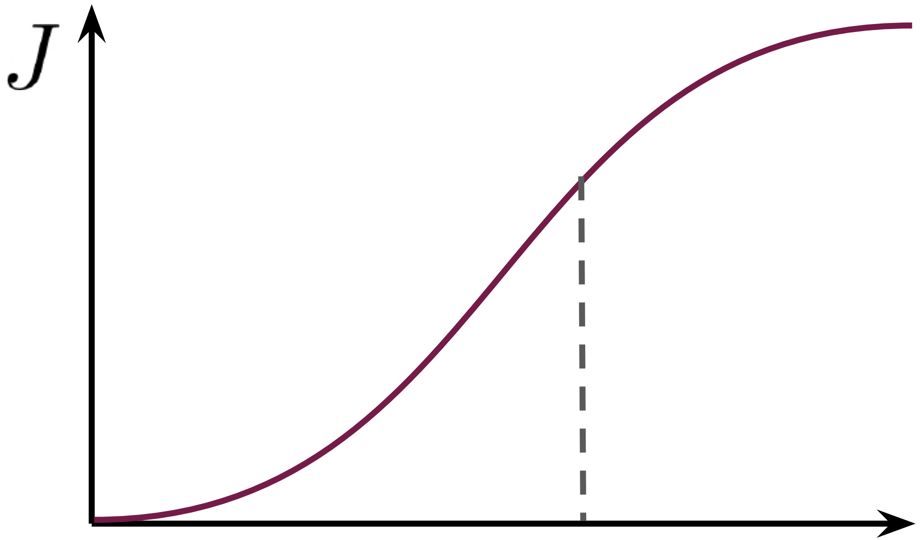
Complex output

Difficult to train

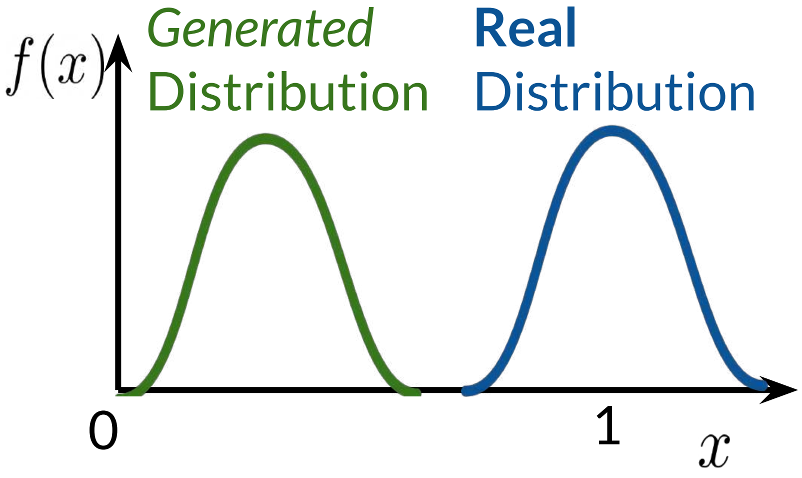
Often, the discriminator gets better than the generator

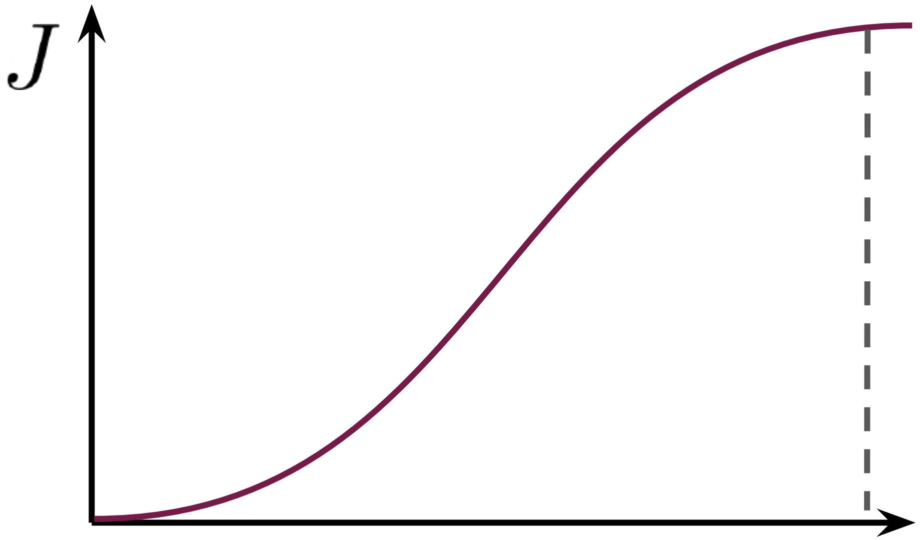
Problems with BCE Loss



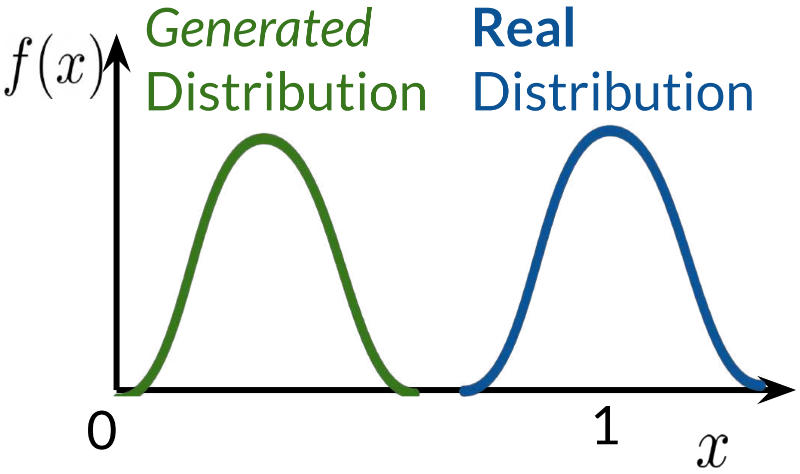


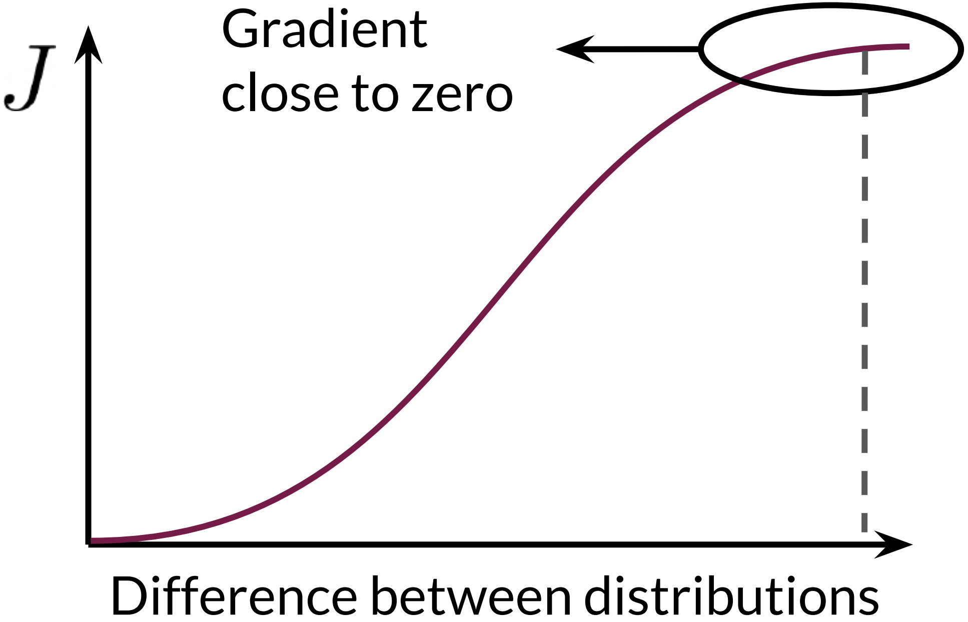
Problems with BCE Loss





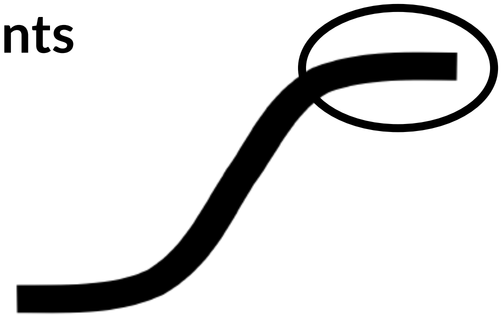
Problems with BCE Loss





Summary

- GANs try to make the real and generated distributions look similar
- When the discriminator improves too much, the function approximated by BCE Loss will contain flat regions
- Flat regions on the cost function = **vanishing gradients**

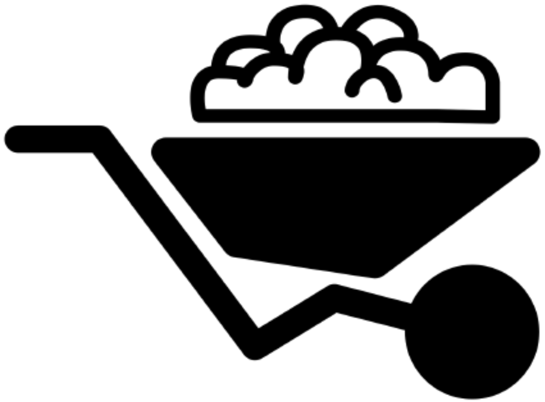


Earth Mover's Distance

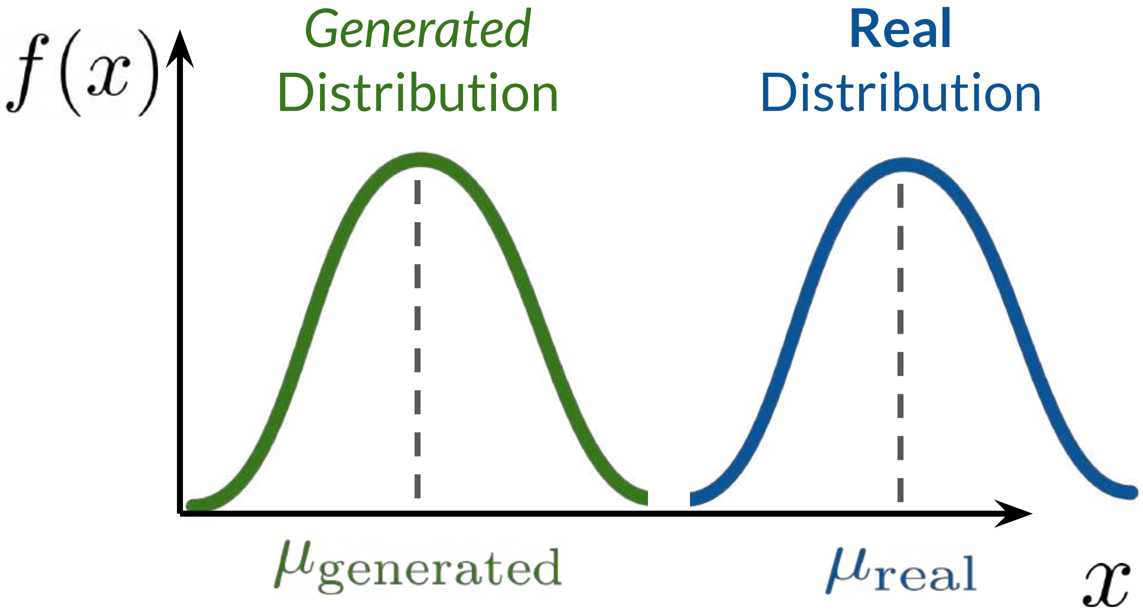
deeplearning.ai

Outline

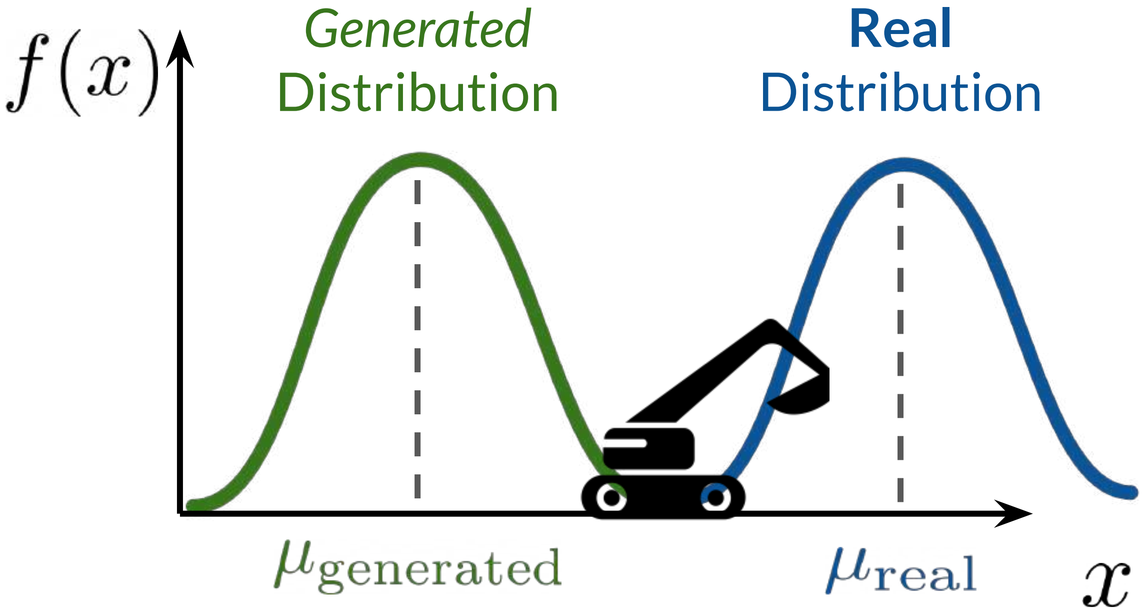
- Earth Mover's Distance (EMD)
- Why it solves the vanishing gradient problem of BCE Loss



Earth Mover's Distance



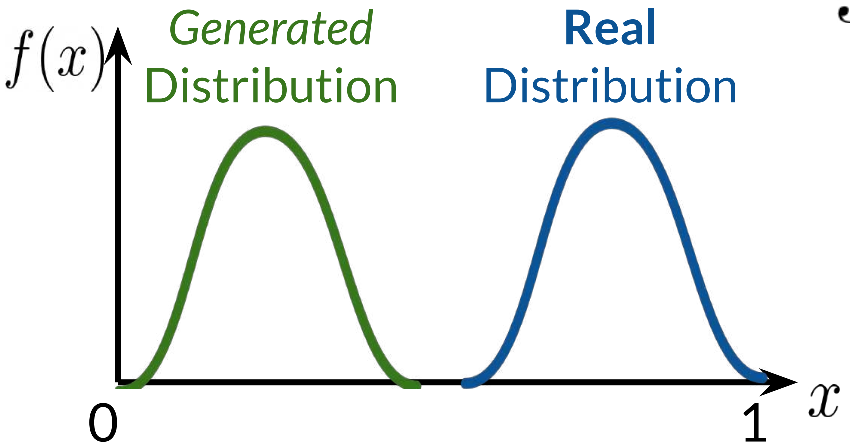
Earth Mover's Distance

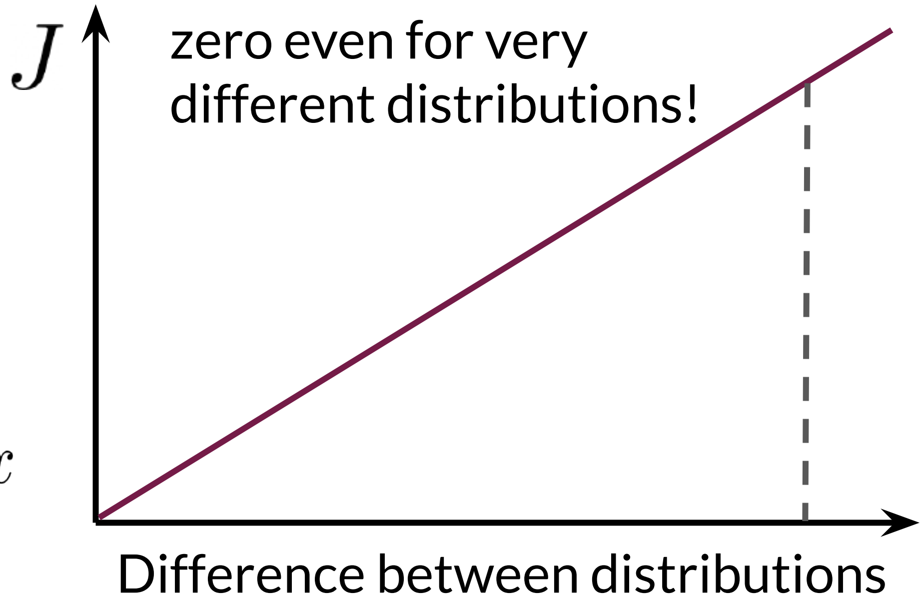


Effort to make the *generated* distribution equal to the **real** distribution

Depends on the distance and amount moved

Earth Mover's Distance





Summary

- Earth mover’s distance (EMD) is a function of amount and distance
- Doesn’t have flat regions when the distributions are very different
- Approximating EMD solves the problems associated with BCE

deeplearning.ai

Wasserstein Loss

Outline

- BCE Loss Simplified
- W-Loss and its comparison with BCE Loss

BCE Loss Simplified

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

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

Discriminator

Minimize cost

Generator

Maximize cost

BCE Loss Simplified

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

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

Minimize cost

Maximize cost

BCE Loss Simplified

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

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

Minimize cost

Maximize cost

W-Loss

W-Loss approximates the Earth Mover's Distance

W-Loss

W-Loss approximates the Earth Mover's Distance

$$\min_g \max_c \; \mathbb{E}(c(x)) - \mathbb{E}(c(g(z)))$$

W-Loss

W-Loss approximates the Earth Mover's Distance

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

Maximize
the
distance

W-Loss

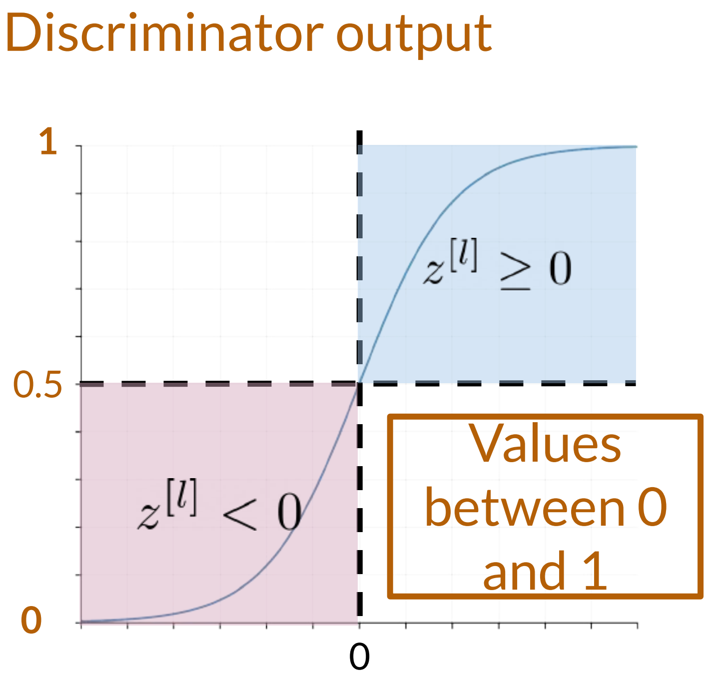
W-Loss approximates the Earth Mover's Distance

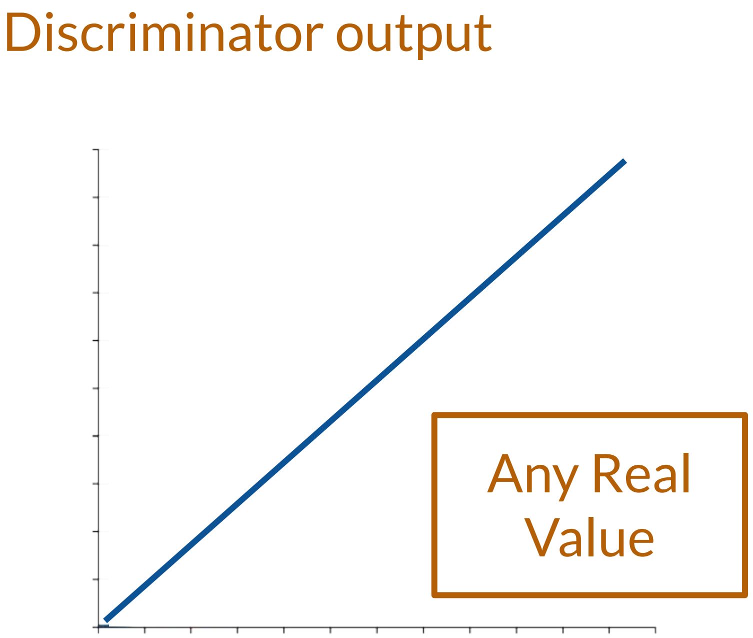
$$\min_{g} \max_{c} \; \mathbb{E}(c(x)) - \mathbb{E}(c(g(z)))$$

Minimize
the
distance

Maximize
the
distance

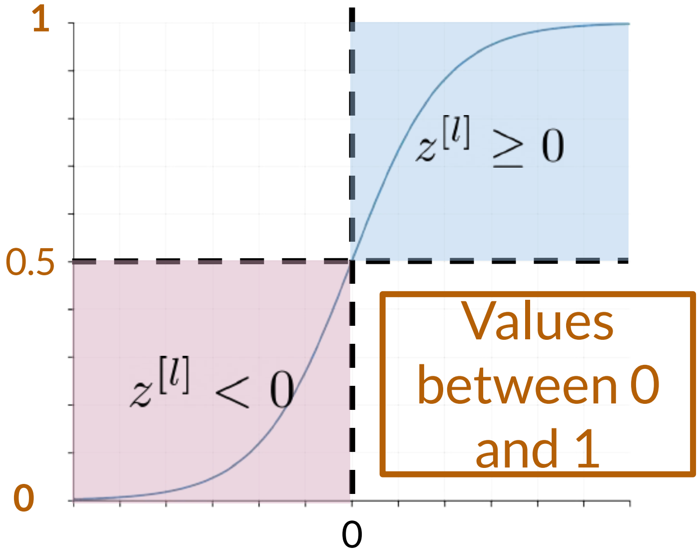
Discriminator Output



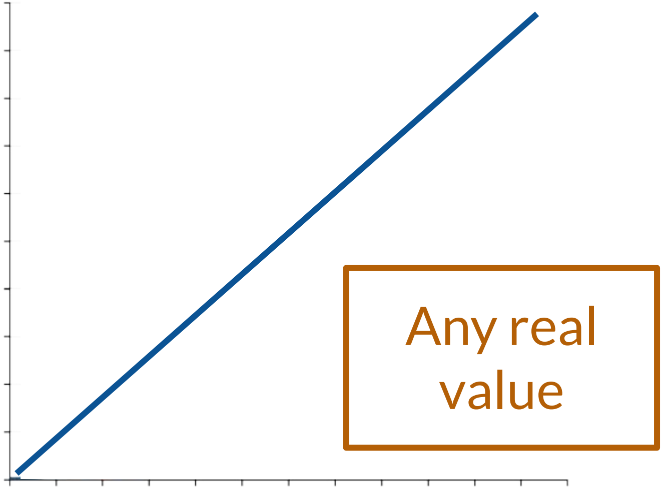


Discriminator Output

Discriminator output



~~Discriminator~~ output
Critic



deeplearning.ai

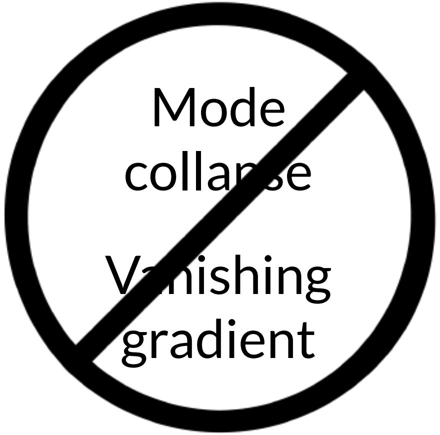
W-Loss vs BCE Loss

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

Summary

- W-Loss looks very similar to BCE Loss
- W-Loss prevents mode collapse and vanishing gradient problems



deeplearning.ai

Condition on Wasserstein Critic

Outline

- Continuity condition on the critic's neural network
- Why this condition matters

Condition on W-Loss

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

Condition on W-Loss

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

Condition on W-Loss

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

Condition on W-Loss

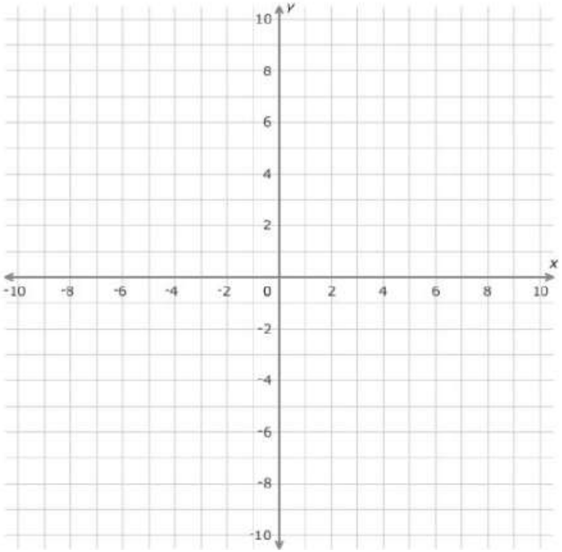
$$\min_{g} \max_{\boxed{c}} \mathbb{E}(\boxed{c}(x)) - \mathbb{E}(\boxed{c}(g(z)))$$

Needs to be 1-Lipschitz Continuous

Condition on W-Loss

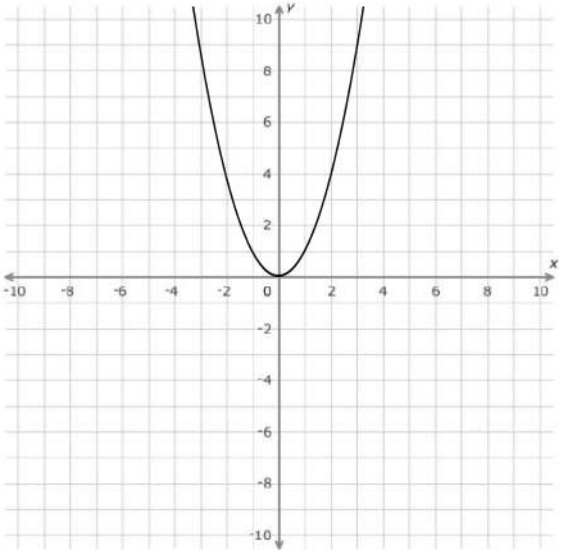
Critic needs to be **1**-L Continuous

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



Condition on W-Loss

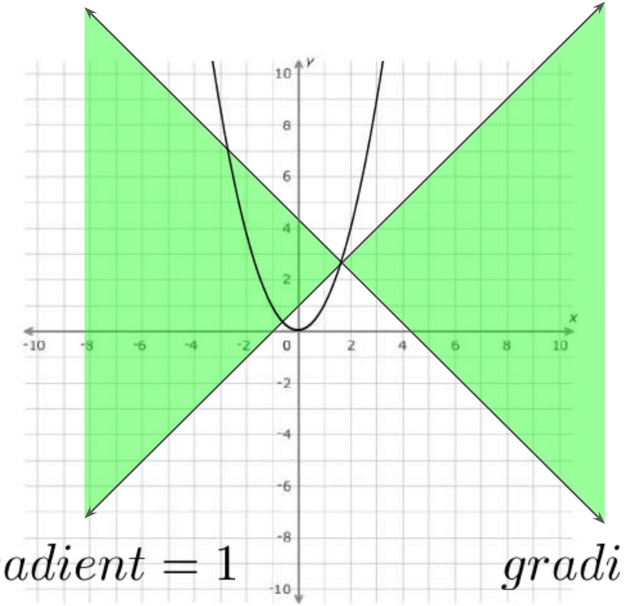
Critic needs to be **1**-L Continuous



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

Condition on W-Loss

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



$gradient = 1$

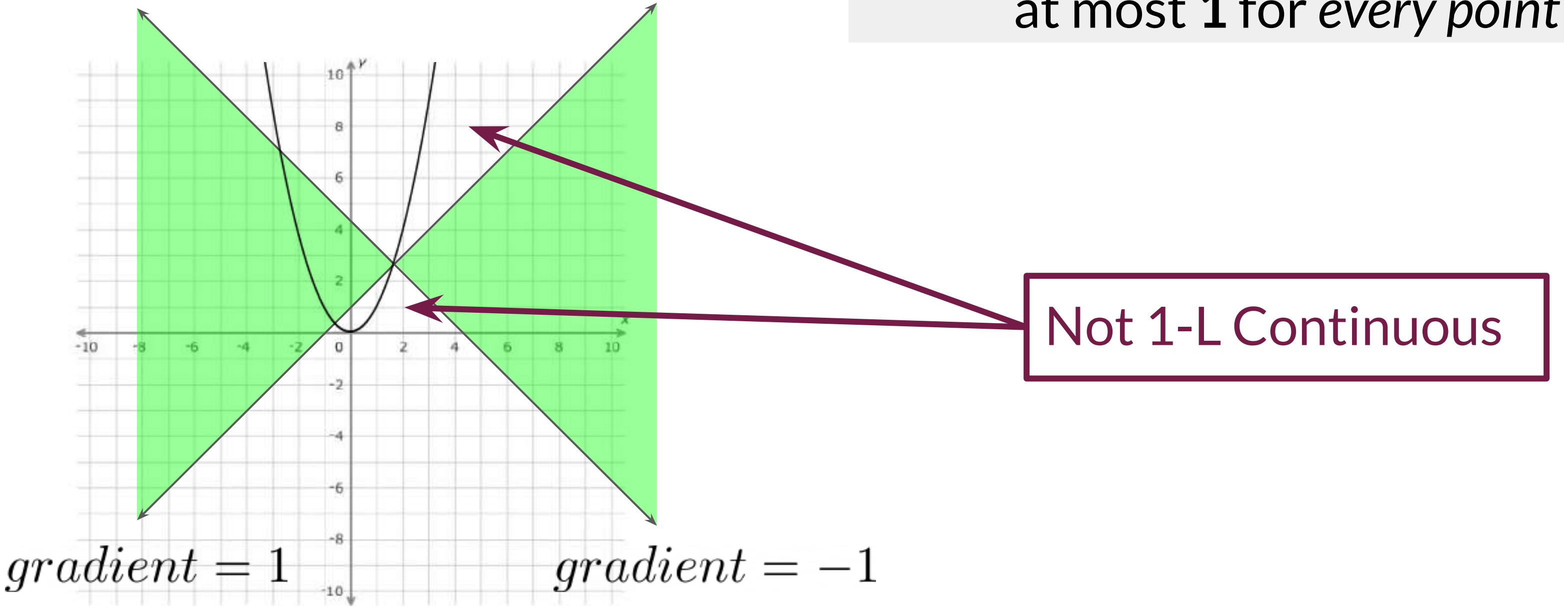
$gradient = -1$

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

Condition on W-Loss

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

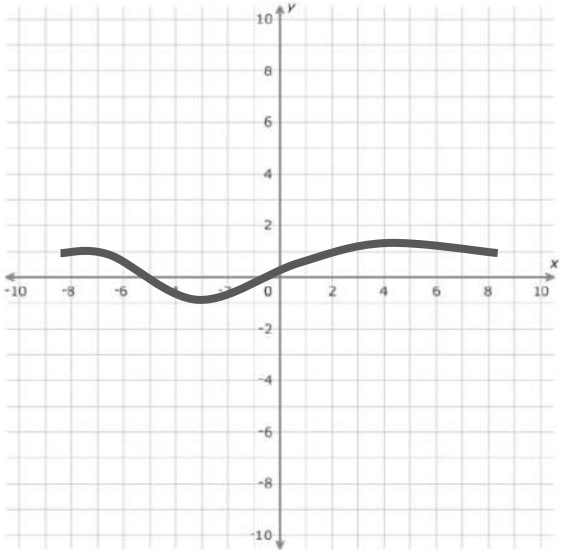
The norm of the gradient should be at most **1** for *every point*



Condition on W-Loss

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

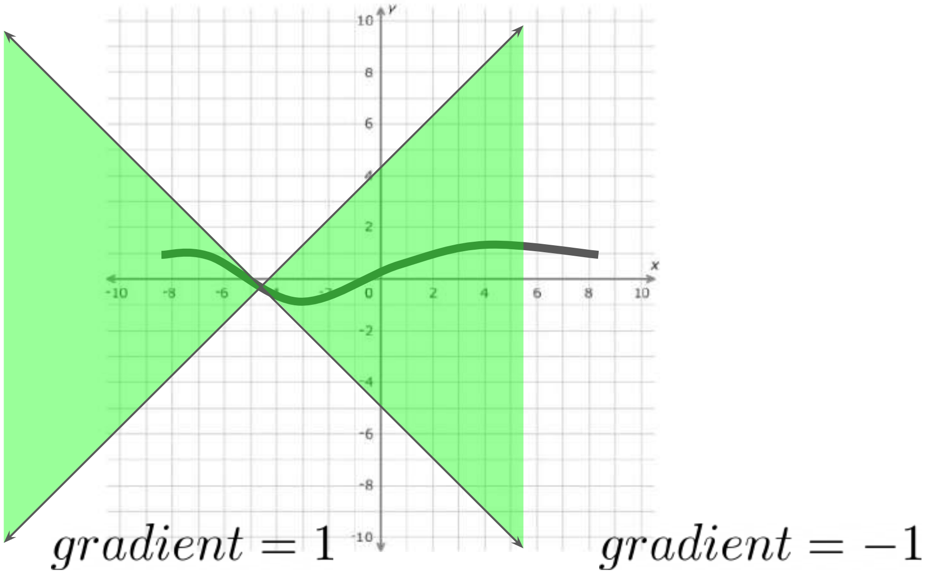
The norm of the gradient should be at most **1** for *every point*



Condition on W-Loss

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

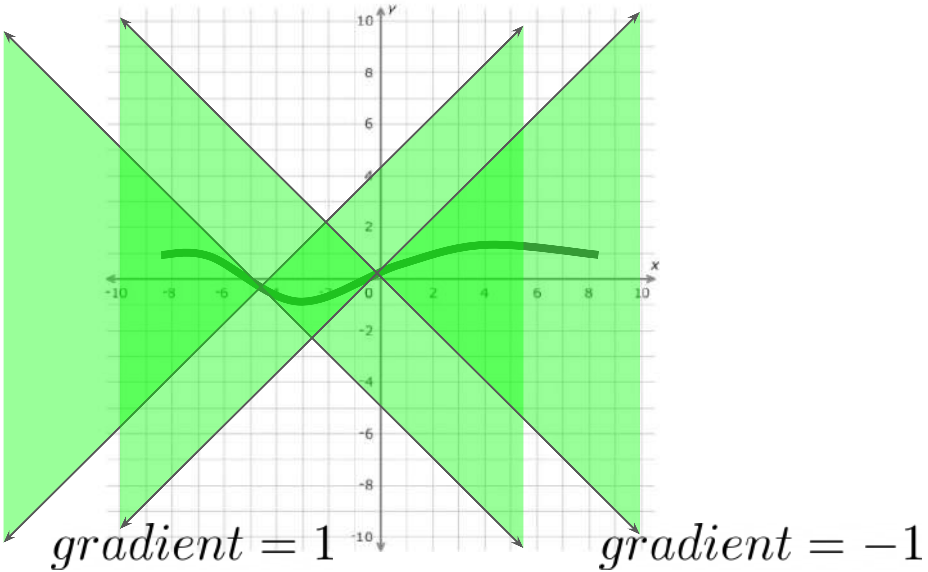
The norm of the gradient should be at most **1** for *every point*



Condition on W-Loss

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

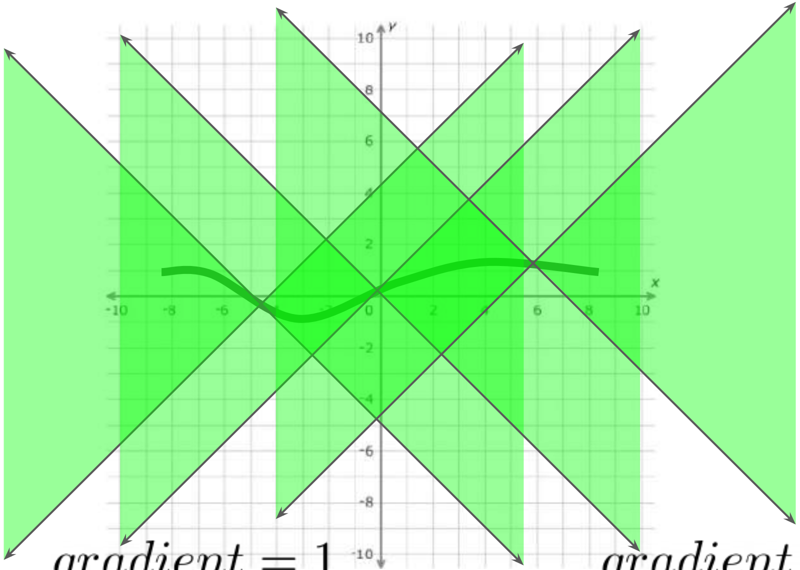
The norm of the gradient should be at most **1** for *every point*



Condition on W-Loss

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

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



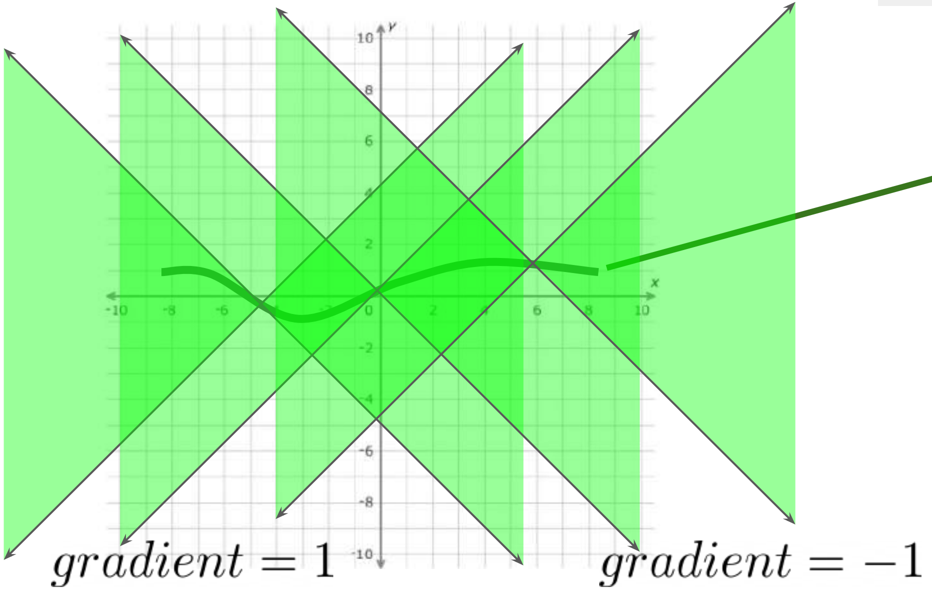
$gradient = 1$ $gradient = -1$

deeplearning.ai

Condition on W-Loss

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

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



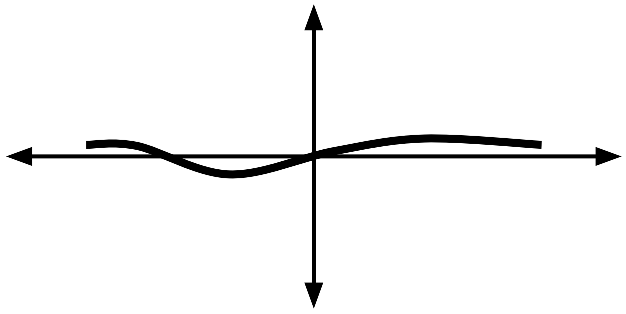
1-L Continuous

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



1-Lipschitz Continuity Enforcement

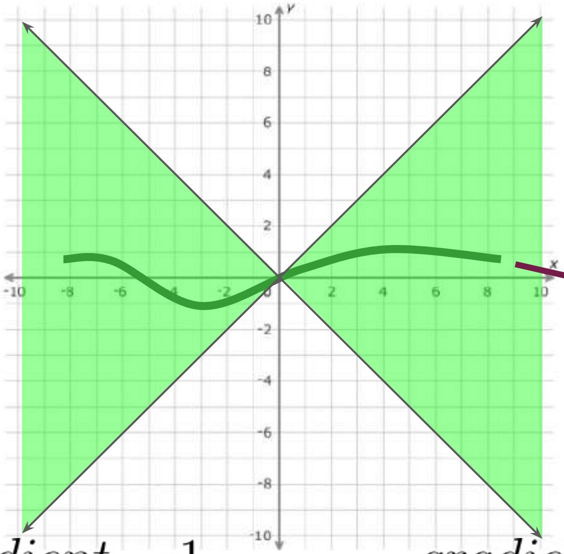
deeplearning.ai

Outline

- Weight clipping and gradient penalty
- Advantages of gradient penalty

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

1-L Enforcement: Weight Clipping

Weight clipping forces the weights of the critic to a fixed interval

Gradient descent to update weights

↓

Clip the critic's weights

Limits the learning ability of the critic

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

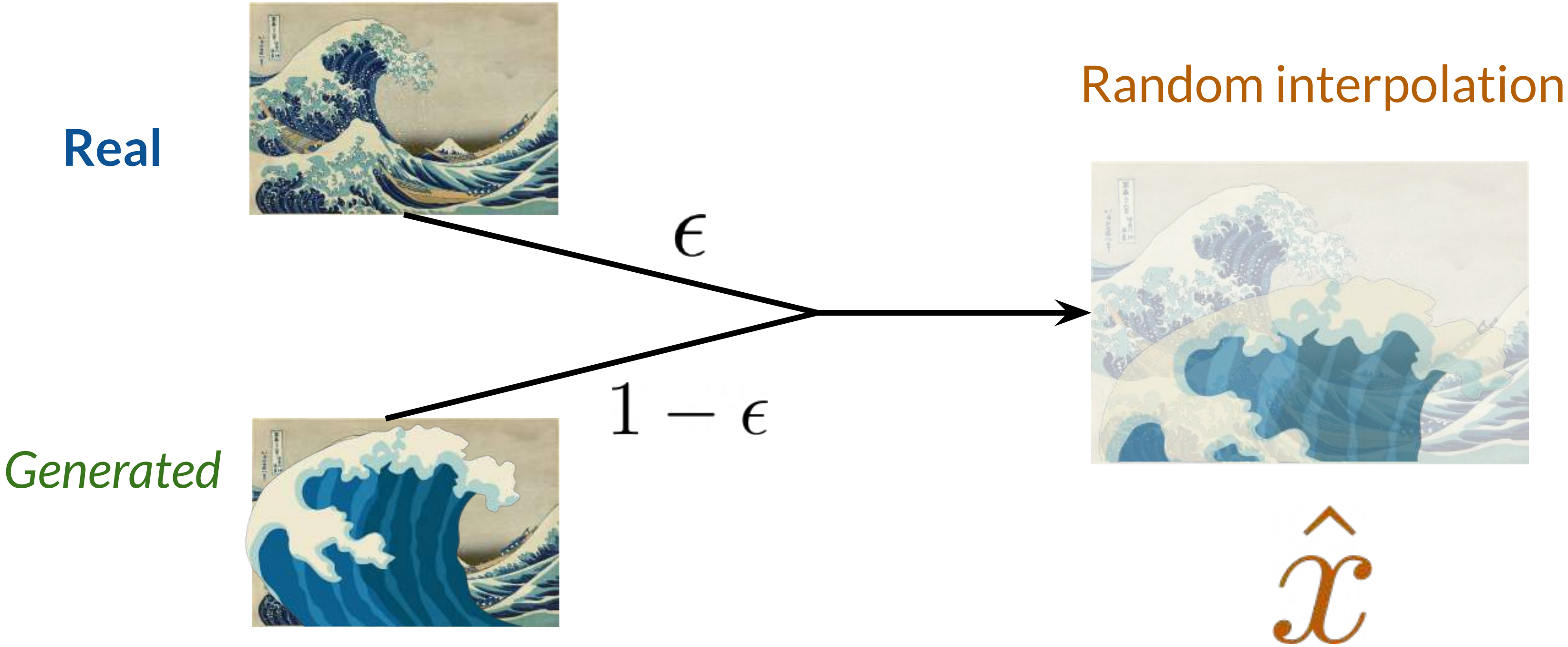
1-L Enforcement: Gradient Penalty

Real

ϵ

Random interpolation

1-L Enforcement: Gradient Penalty



deeplearning.ai

1-L Enforcement: Gradient Penalty

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

Regularization term

1-L Enforcement: Gradient Penalty

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

Regularization term

1-L Enforcement: Gradient Penalty

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

Regularization term

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

Interpolation

1-L Enforcement: Gradient Penalty

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

Regularization term

$$\epsilon \boxed{x} + (1 - \epsilon) g(z)$$

Real

Interpolation

1-L Enforcement: Gradient Penalty

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

Regularization term

$$\epsilon \boxed{x} + (1 - \epsilon)\boxed{g(z)}$$

Interpolation

Real

Generated

Putting It All Together

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

Putting It All Together

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

Makes the GAN less prone to **mode collapse** and **vanishing gradient**

Putting It All Together

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

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

deeplearning.ai

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

Conditional Generation: Intuition

Outline

- Unconditional generation
- Conditional vs. unconditional generation

deeplearning.ai

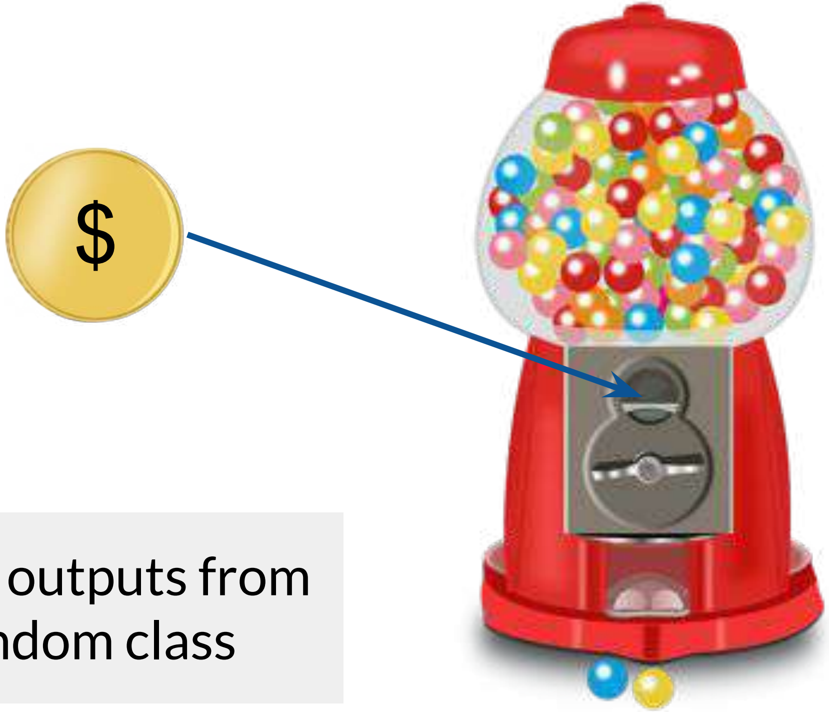
Unconditional Generation

You get outputs from a random class

Unconditional Generation

You get outputs from a random class

Unconditional Generation

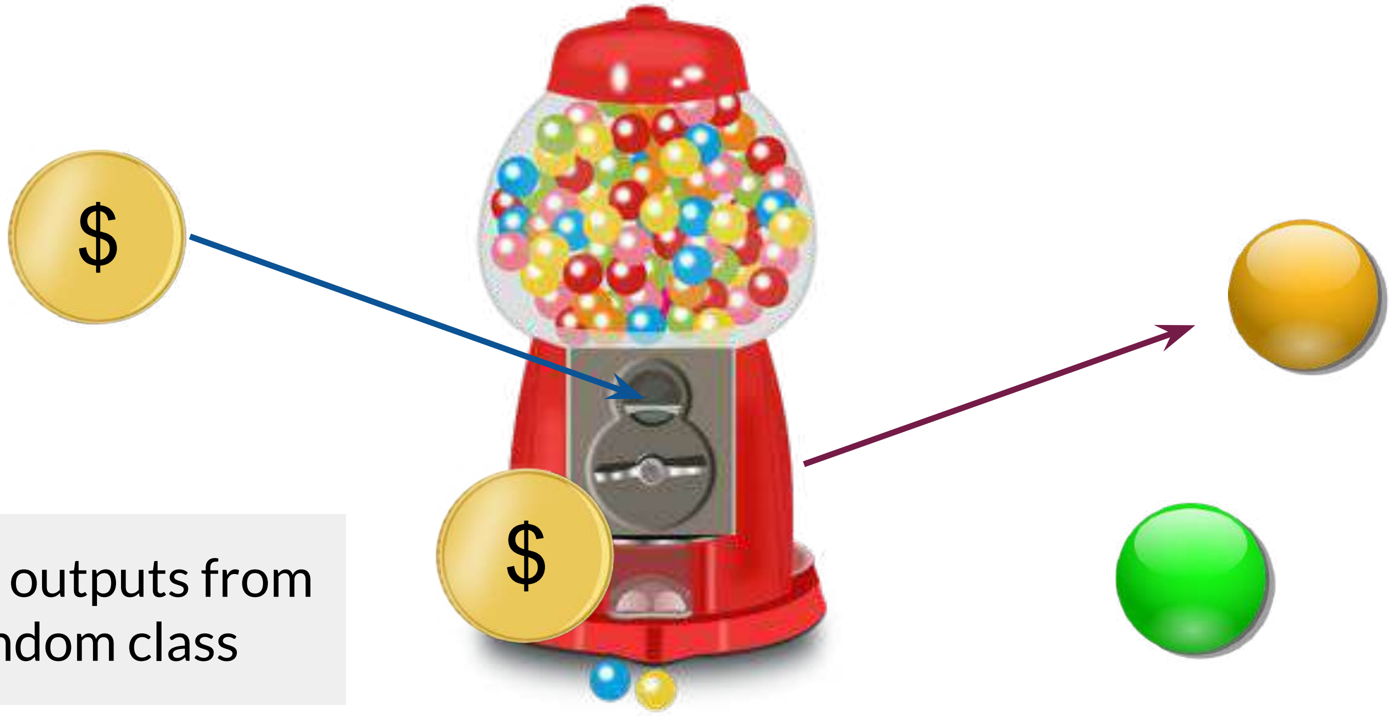


You get outputs from a random class

Unconditional Generation

You get outputs from a random class

Unconditional Generation



You get outputs from a random class

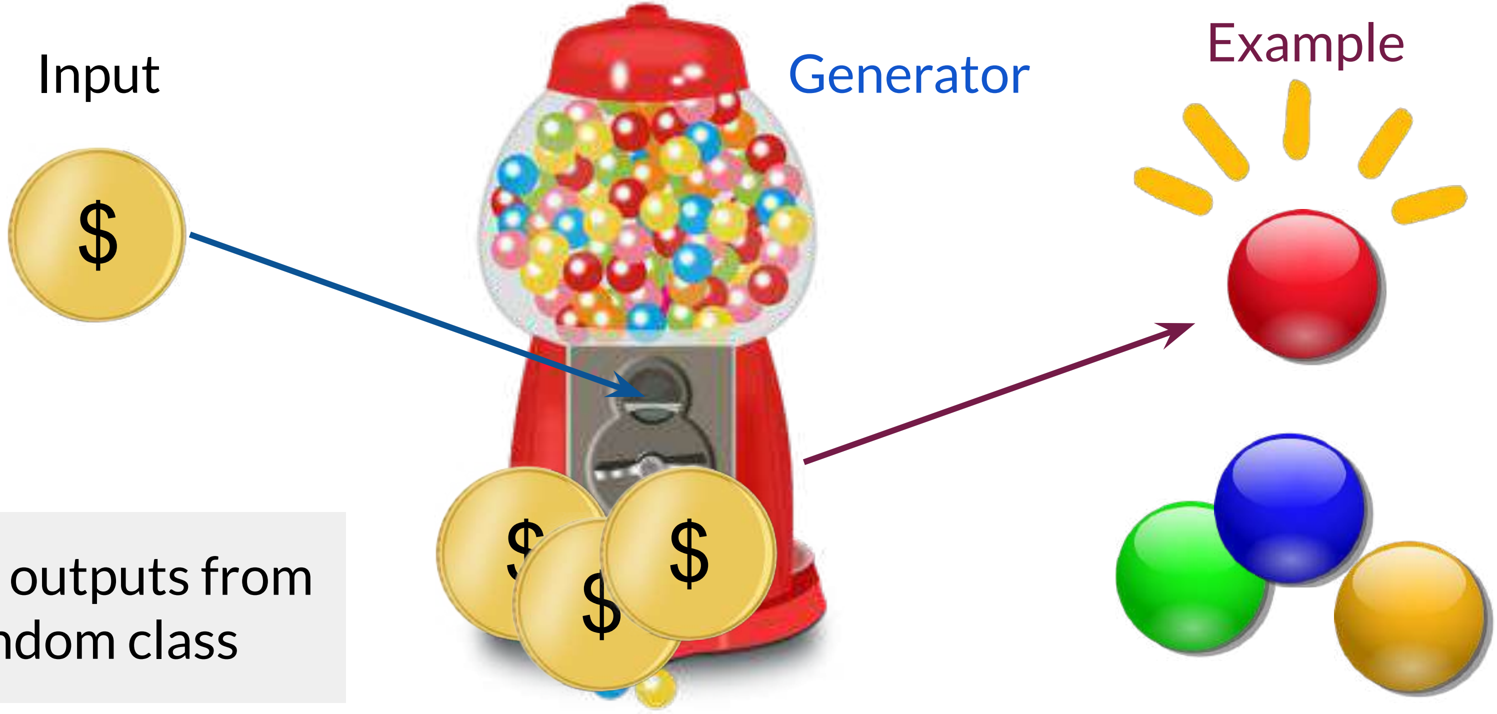
Unconditional Generation

You get outputs from a random class

Unconditional Generation

You get outputs from a random class

Unconditional Generation

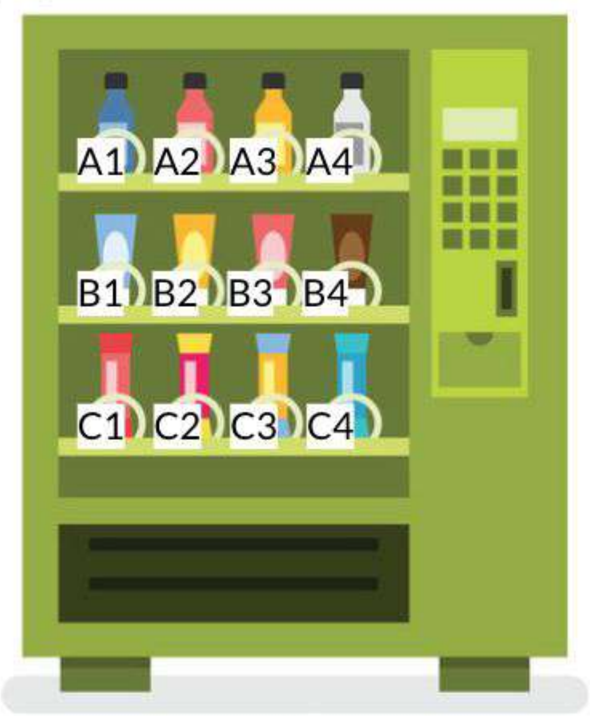


You get outputs from a random class

Conditional Generation

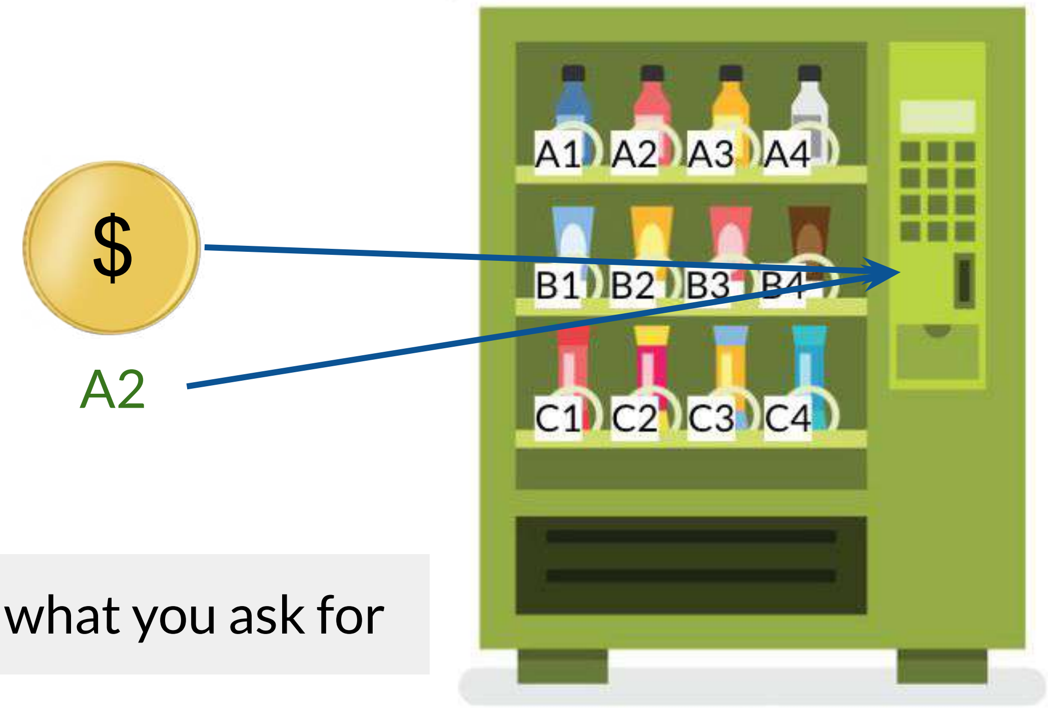
You get what you ask for

Conditional Generation



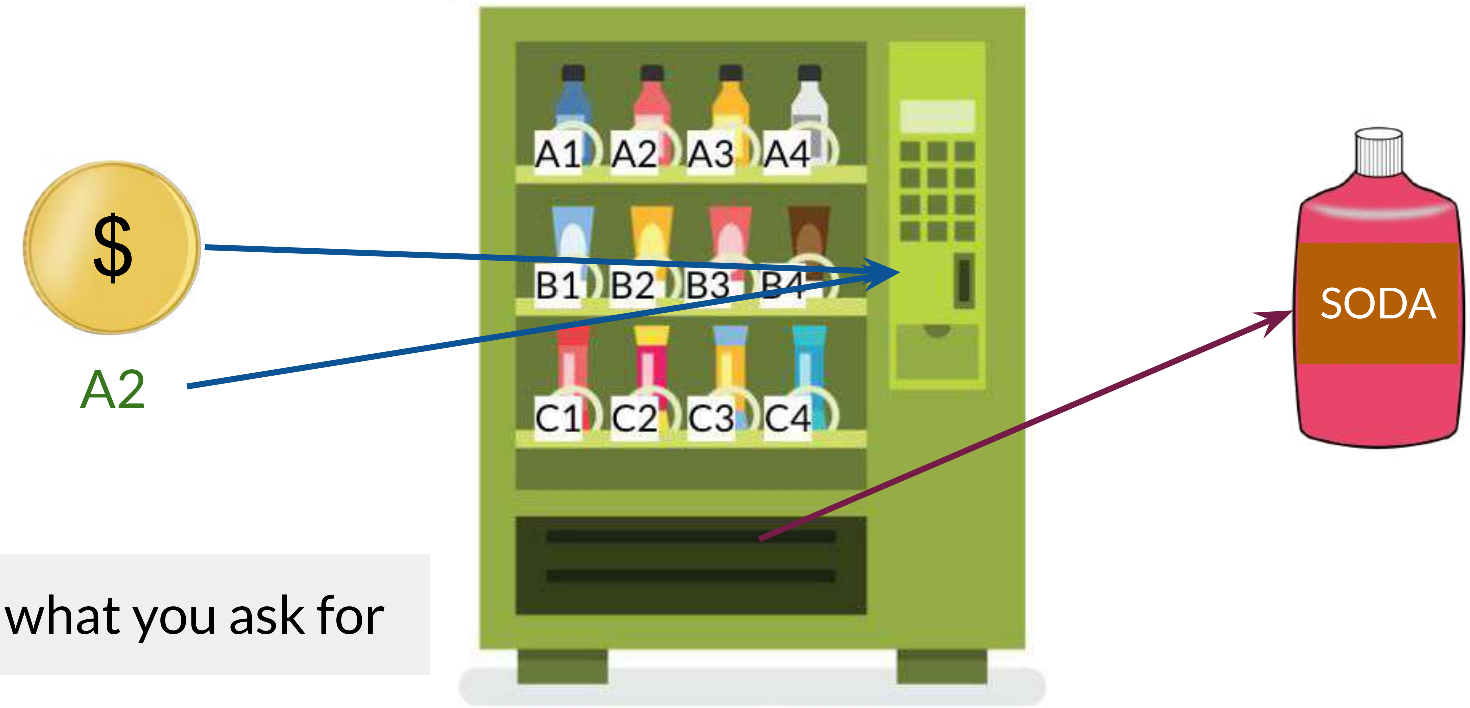
You get what you ask for

Conditional Generation



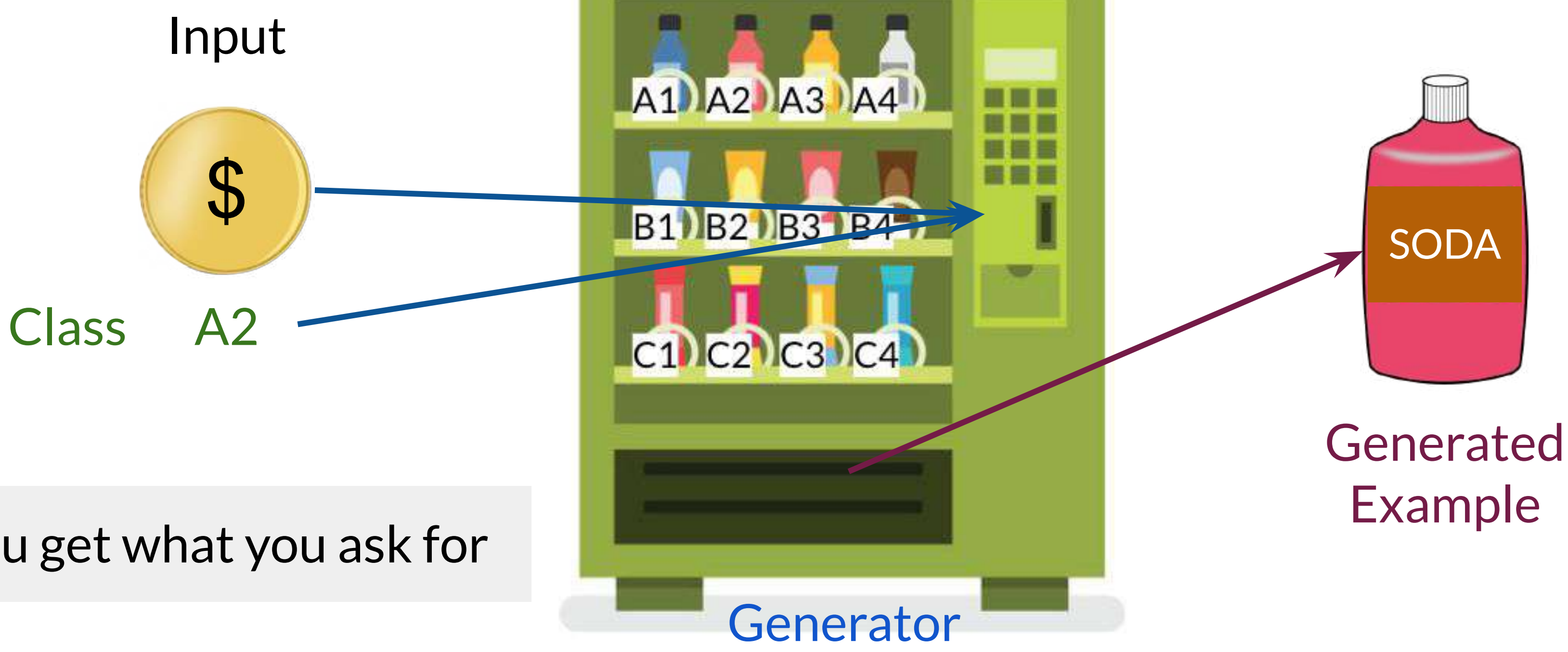
You get what you ask for

Conditional Generation



You get what you ask for

Conditional Generation



You get what you ask for

Conditional vs. Unconditional Generation

Conditional	Unconditional

Conditional vs. Unconditional Generation

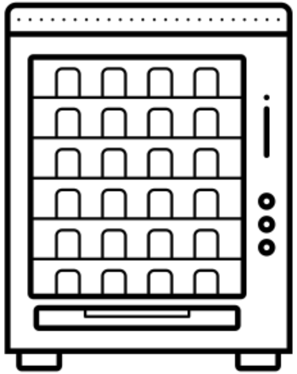
Conditional	Unconditional
Examples from **the classes you want**	Examples from *random classes*

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*

Summary

- Conditional generation requires labeled datasets
- Examples can be generated for the selected class

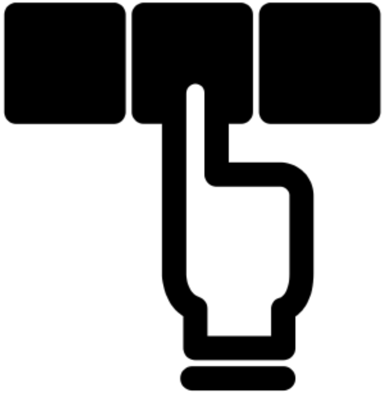


deeplearning.ai

Conditional Generation: Inputs

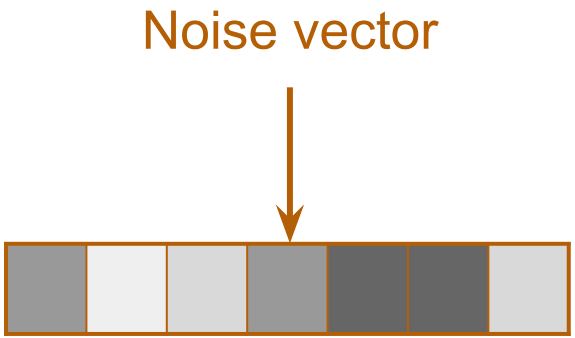
Outline

- How to tell the generator what type of example to produce
- Input representation for the discriminator

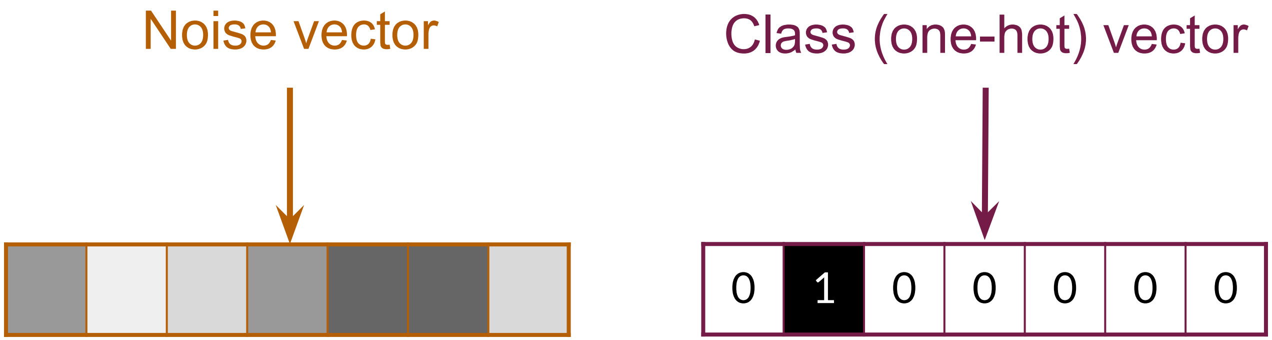


deeplearning.ai

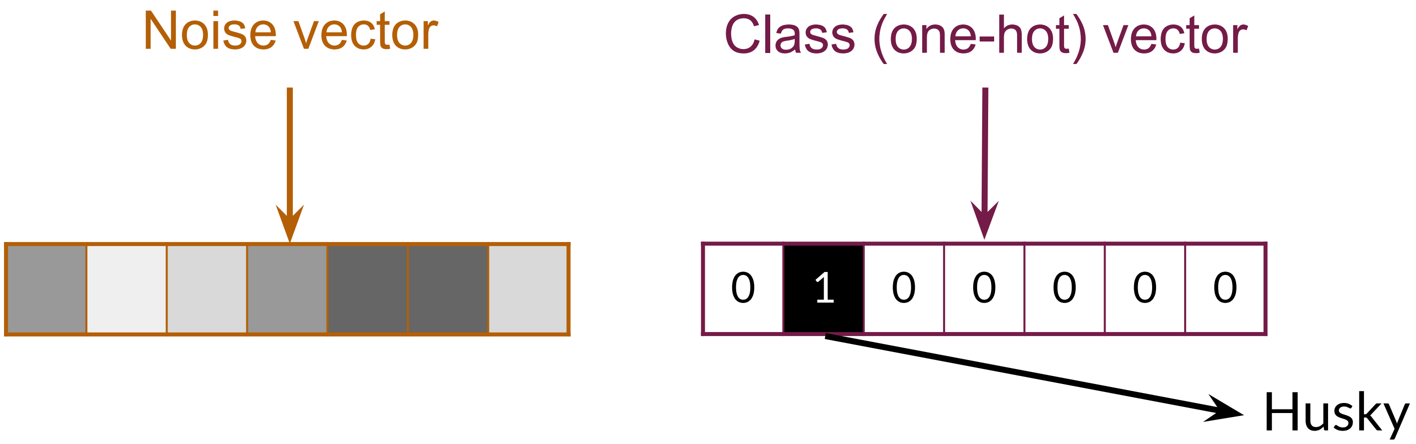
Generator Input



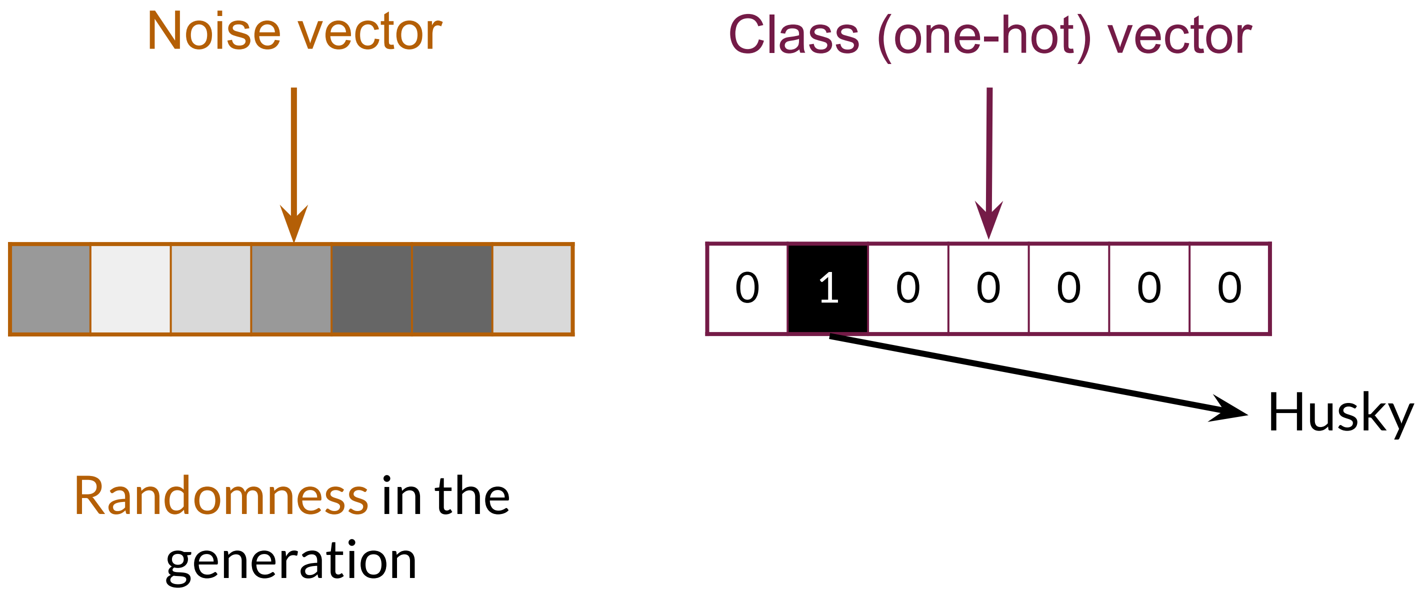
Generator Input



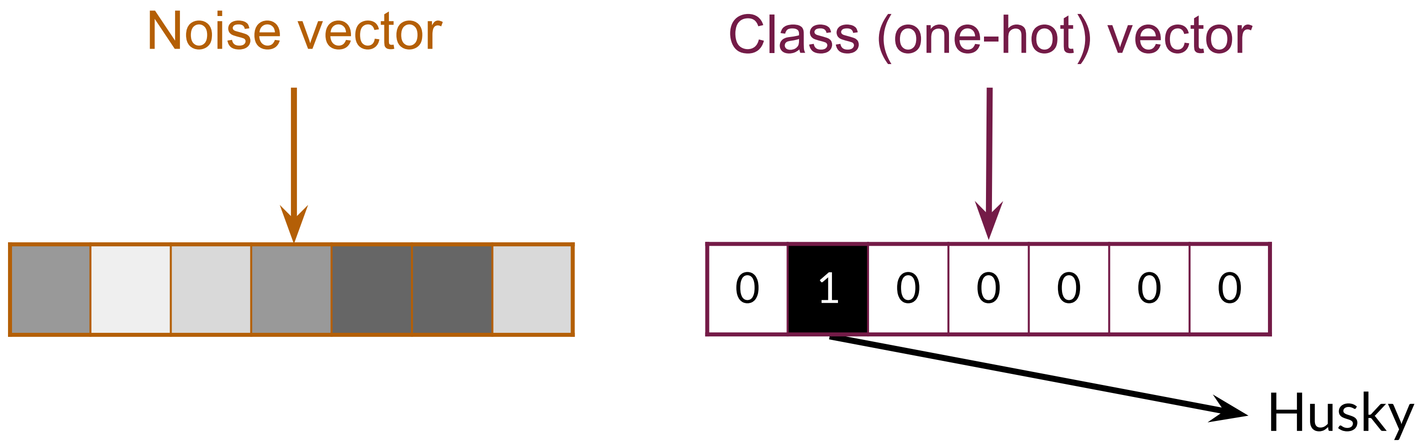
Generator Input



Generator Input



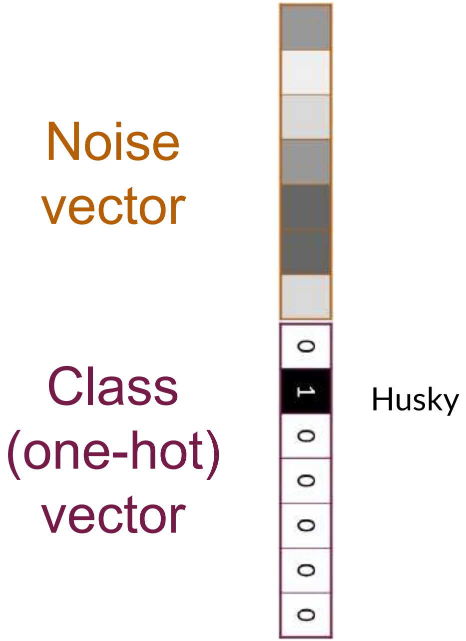
Generator Input



Randomness in the generation

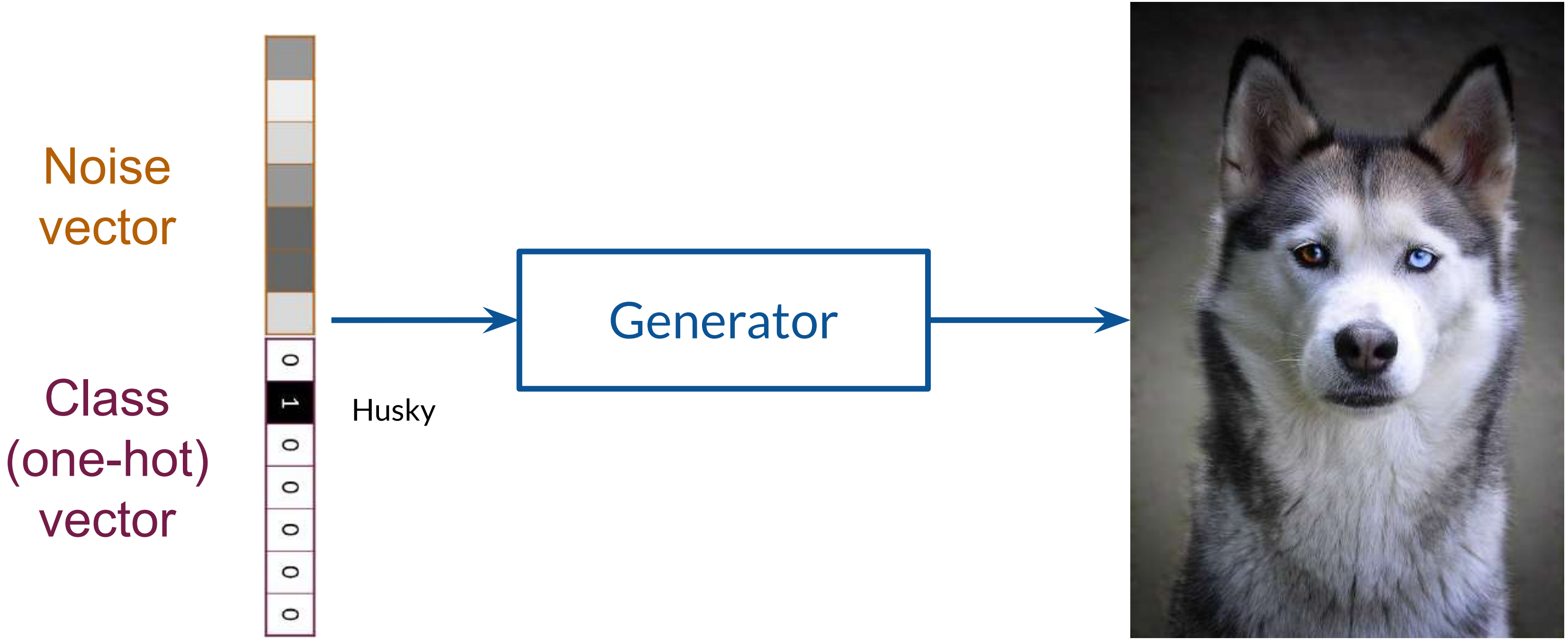
Control in the generation

Generator Input

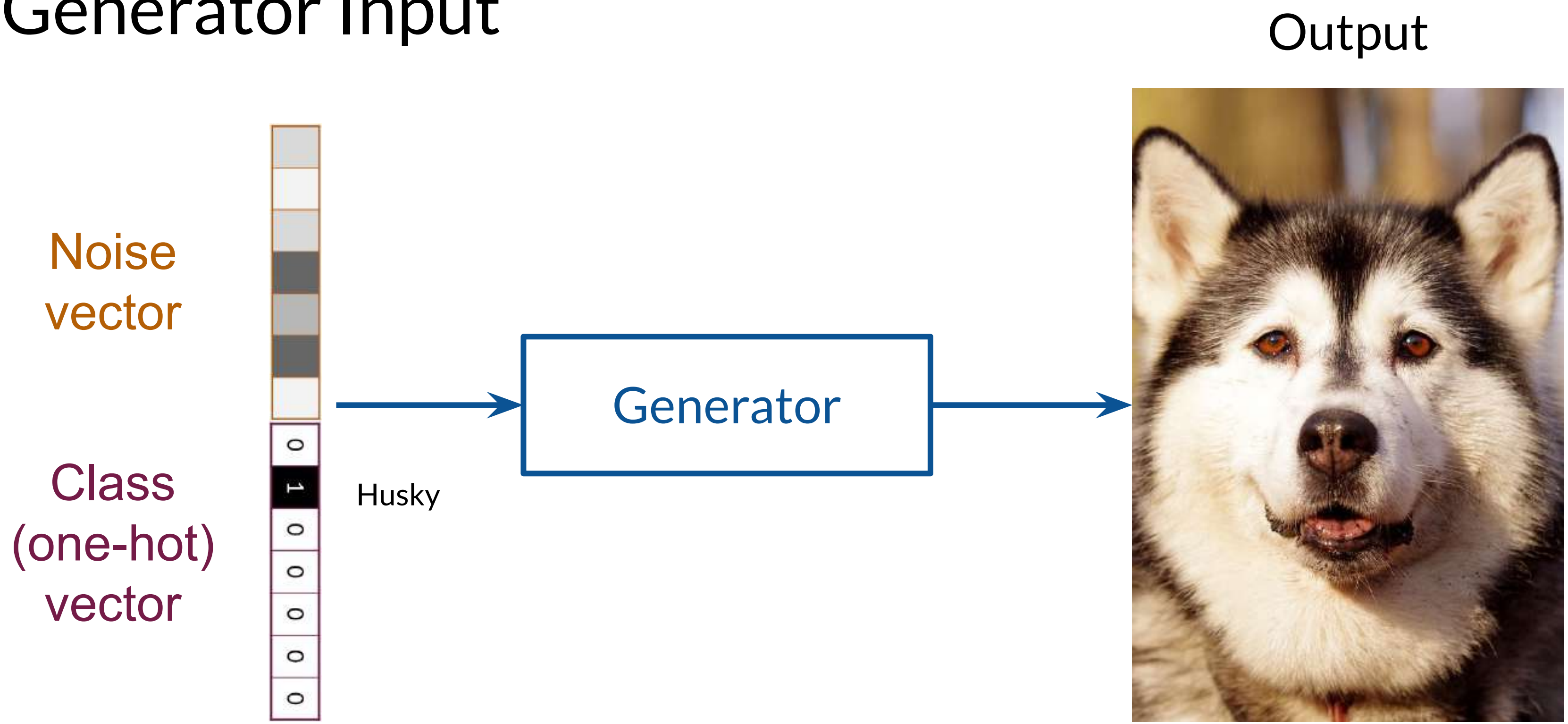


Generator Input

Output



Generator Input



Discriminator Input

Discriminator

Discriminator Input

Discriminator

Golden Retriever

Discriminator Input

Golden Retriever

Discriminator

Output

5%
Real

Discriminator Input

Golden Retriever

Discriminator

Discriminator Input

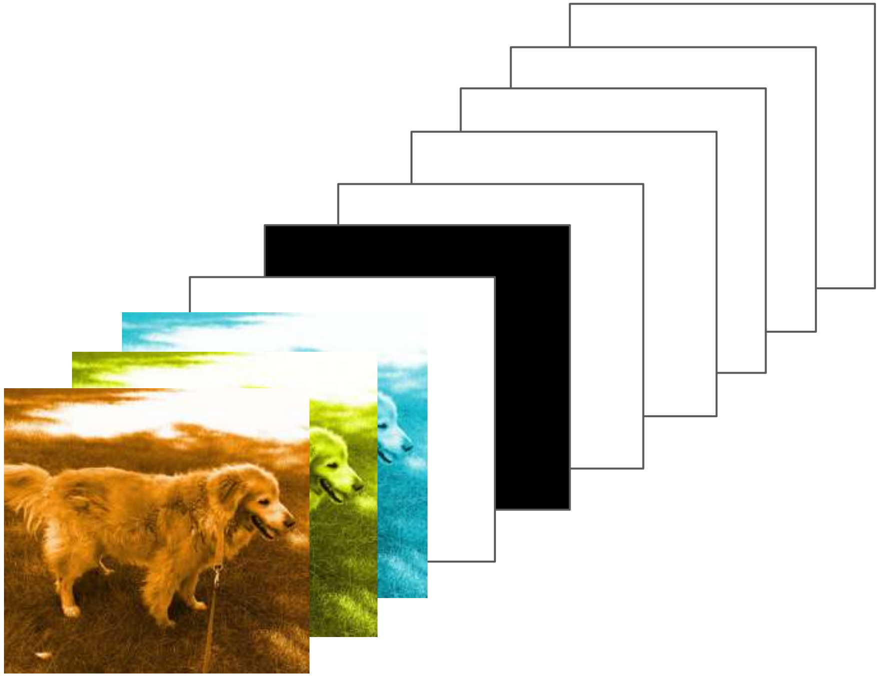
Golden Retriever

Discriminator

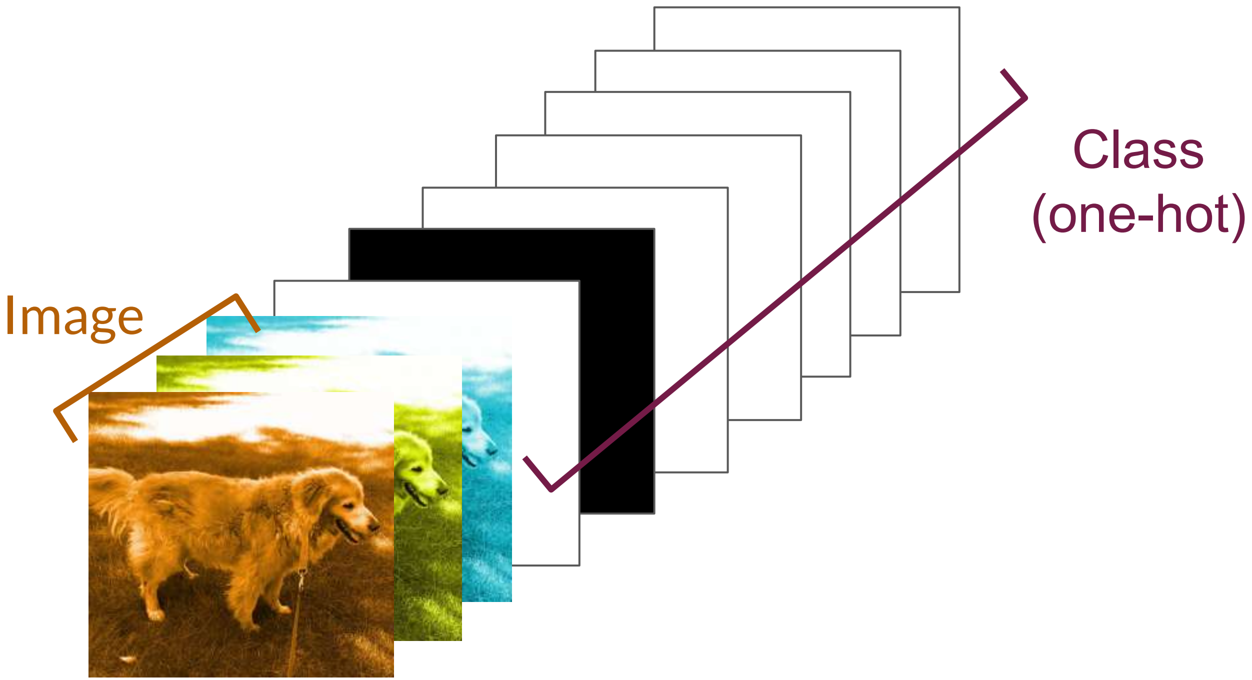
Output

98%
Real

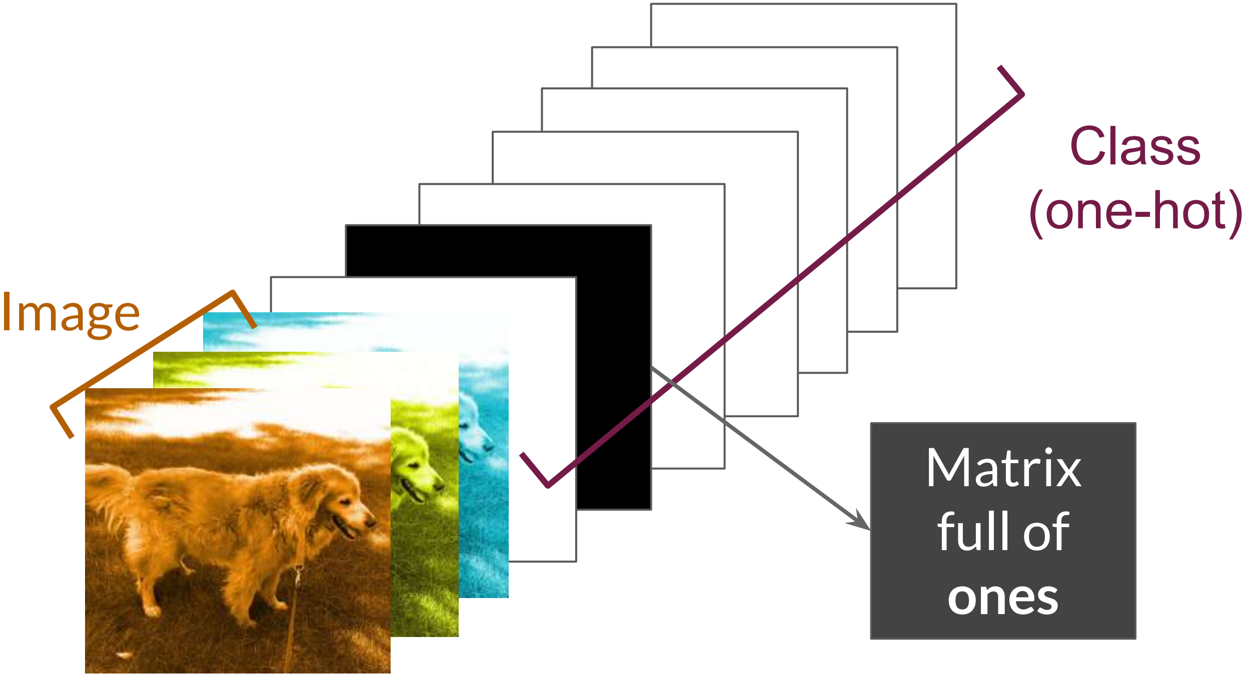
Discriminator Input



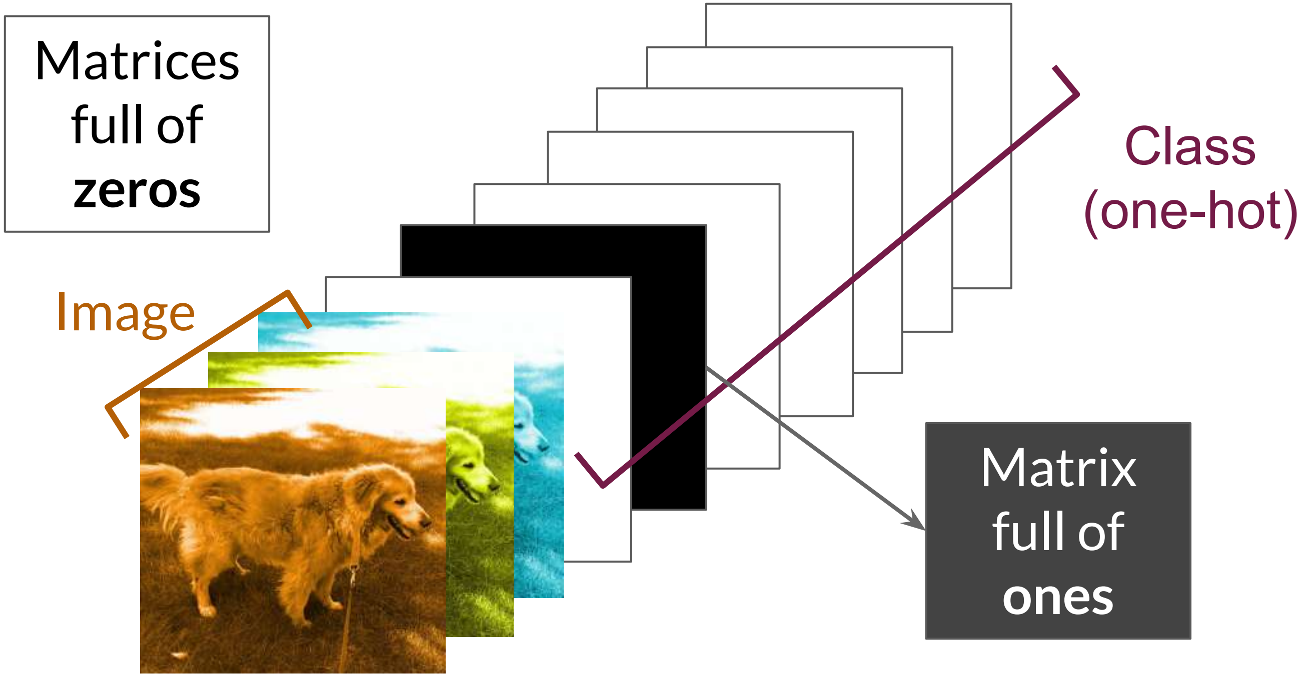
Discriminator Input



Discriminator Input



Discriminator Input



Summary

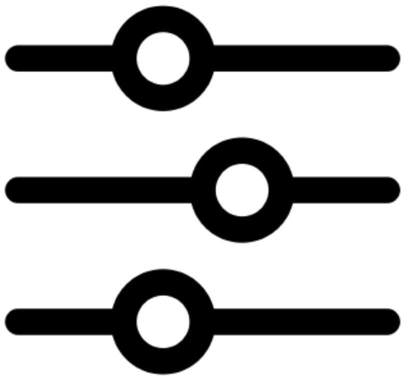
- The class is passed to the generator as one-hot vectors
- The class is passed to the discriminator as one-hot matrices
- The size of the vector and the number of matrices represent the number of classes

deeplearning.ai

Controllable Generation

Outline

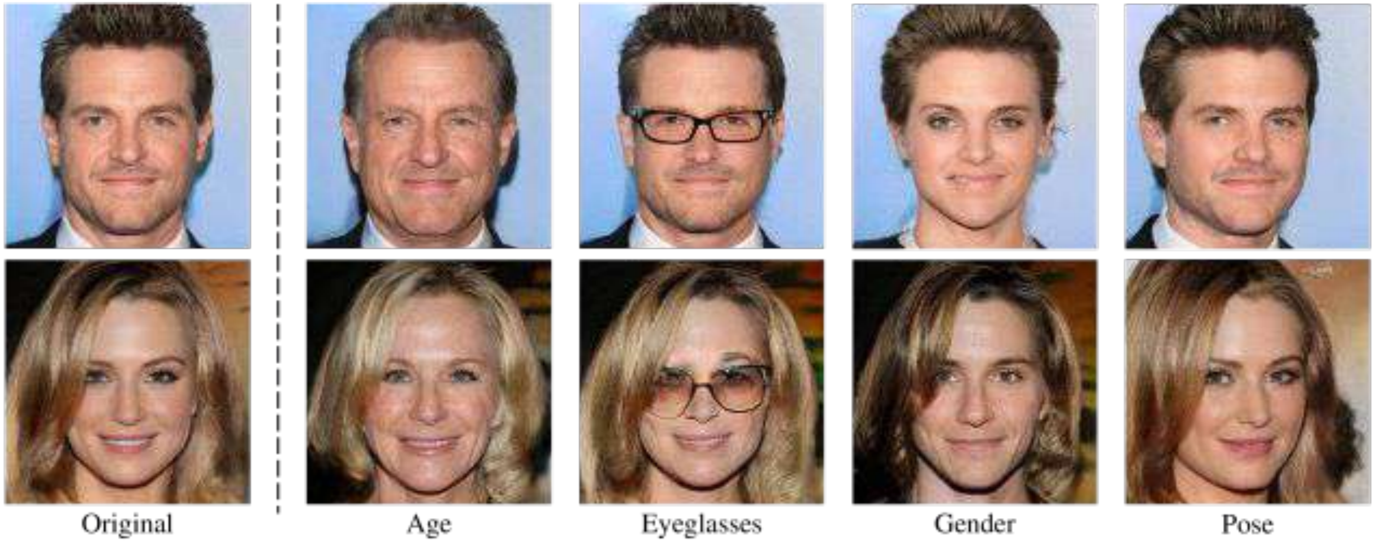
- What is controllable generation
- How it compares to conditional generation



Controllable Generation

Change specific features of the output

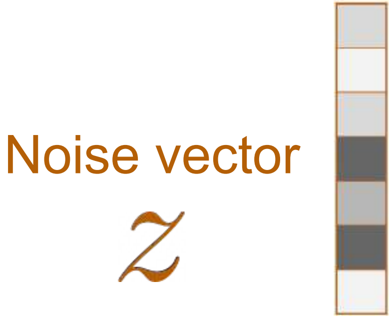
Controllable Generation



Change specific features of the output

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

Controllable Generation



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

Controllable Generation

Noise vector

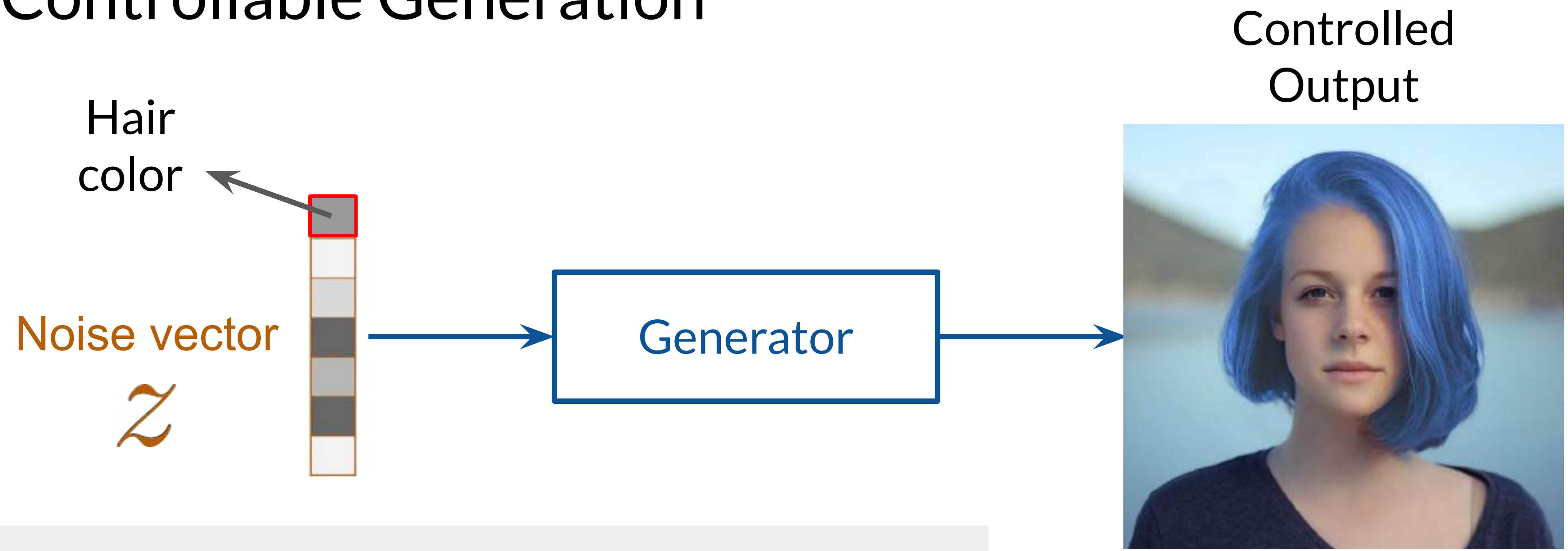
z

Generator

Controlled Output

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

Controllable Generation



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

Controllable Generation vs. Conditional Generation

Controllable	Conditional

Controllable Generation vs. Conditional Generation

Controllable	Conditional
Examples with the **features that you want**	Examples from *the classes you want*

Controllable Generation vs. Conditional Generation

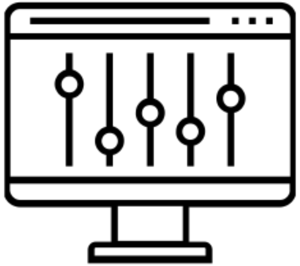
Controllable	Conditional
Examples with the **features that you want**	Examples from *the classes you want*
Training dataset **doesn't need to be labeled**	Training dataset *needs to be labeled*

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

Summary

- Controllable generation lets you control the features of the generated outputs
- It does not need a labeled training dataset
- The input vector is tweaked to get different features on the output



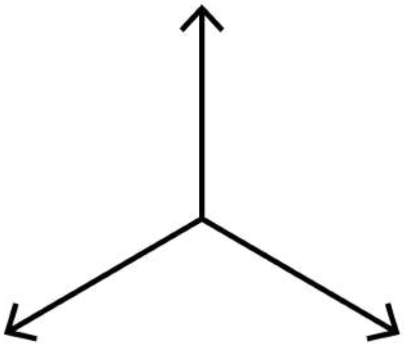
deeplearning.ai

Vector Algebra in the Z-Space

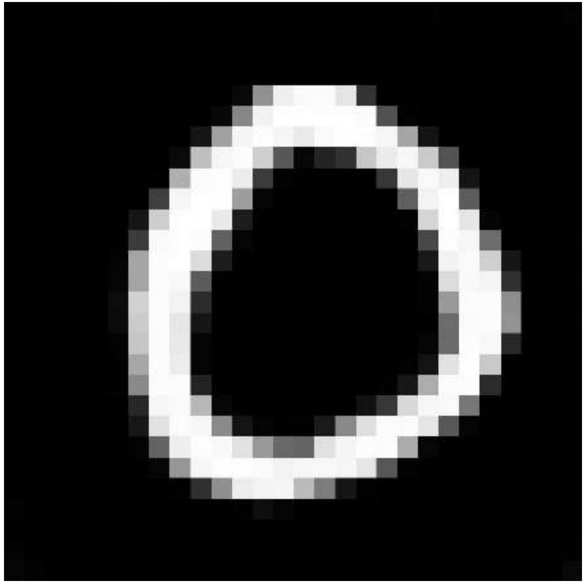
Outline

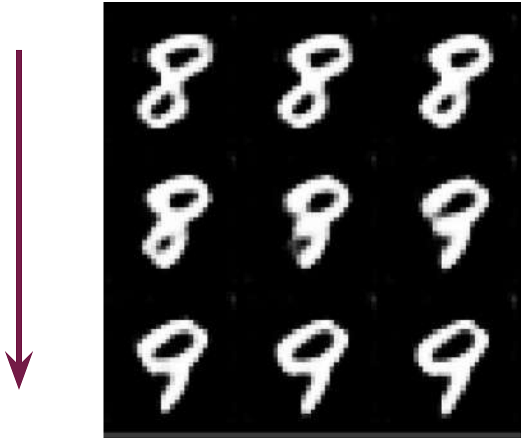
- Interpolation in the Z-space

- Modifying the noise vector z to control desired features



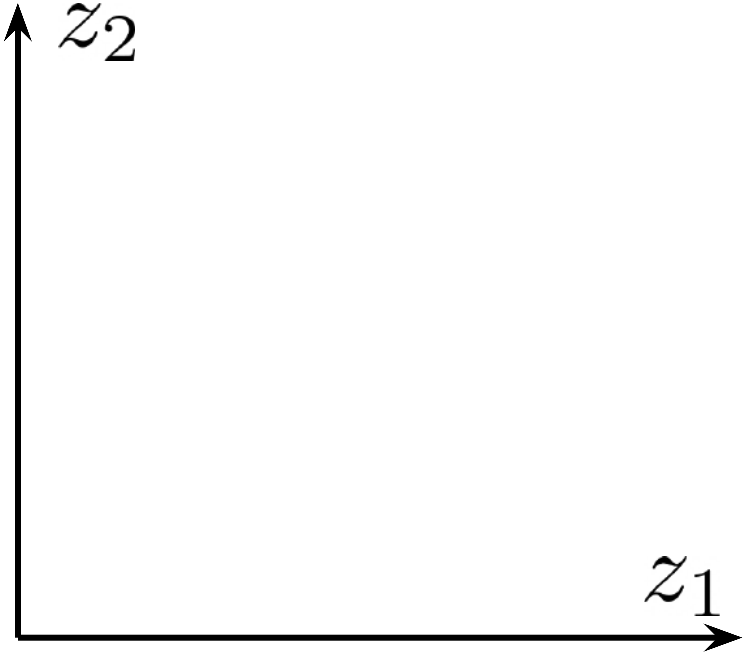
Interpolation Using the Z-Space





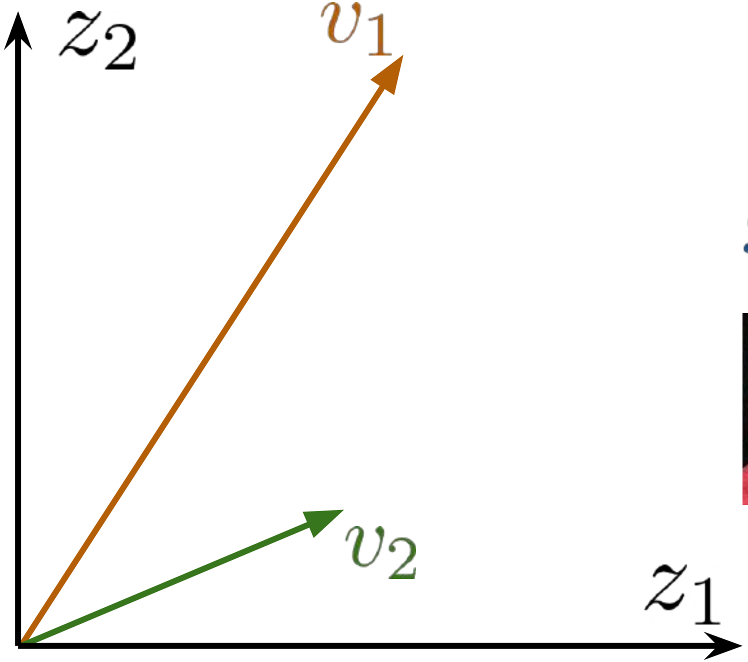
How an image morphs into another

Interpolation Using the Z-Space



Z-Space with Noise Vectors

Interpolation Using the Z-Space

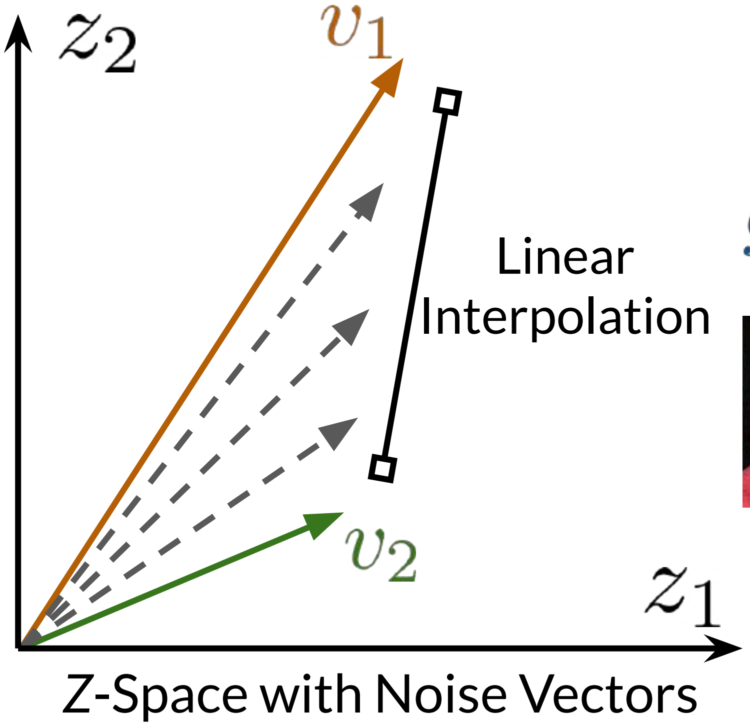


Z-Space with Noise Vectors

$g(v_1)$

$g(v_2)$

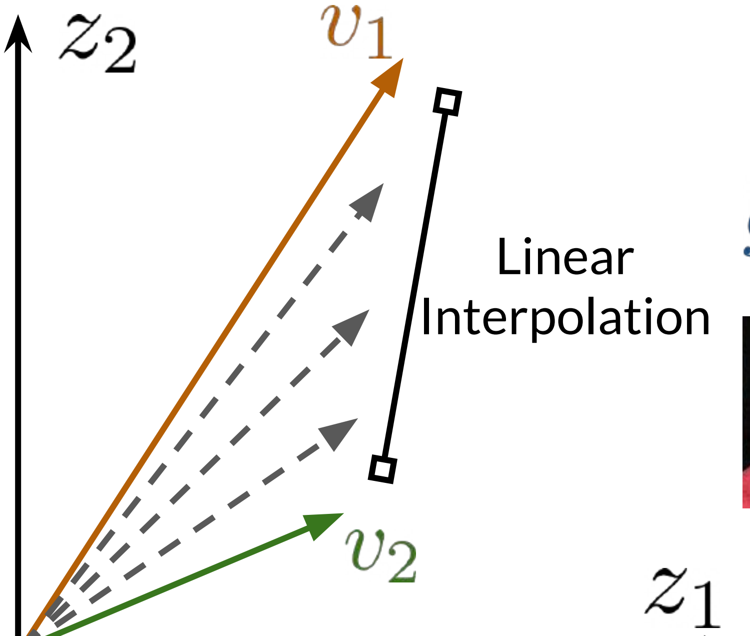
Interpolation Using the Z-Space



$g(v_1)$

$g(v_2)$

Interpolation Using the Z-Space

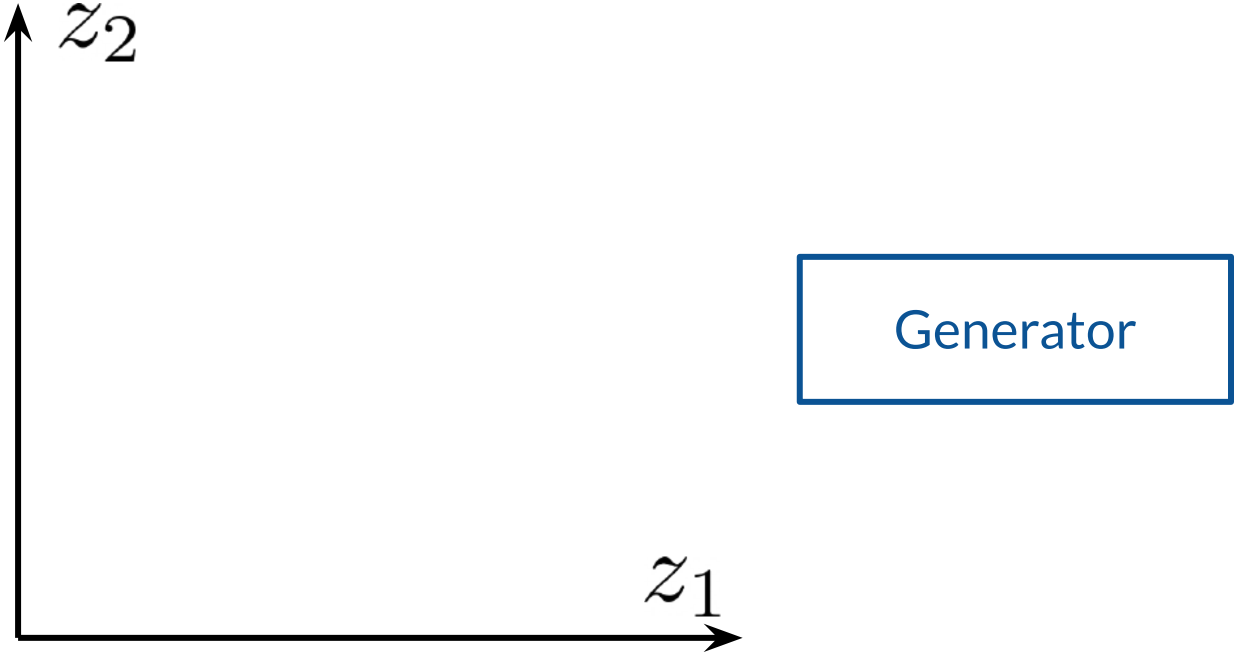


Z-Space with Noise Vectors

Intermediate images using the Z-space

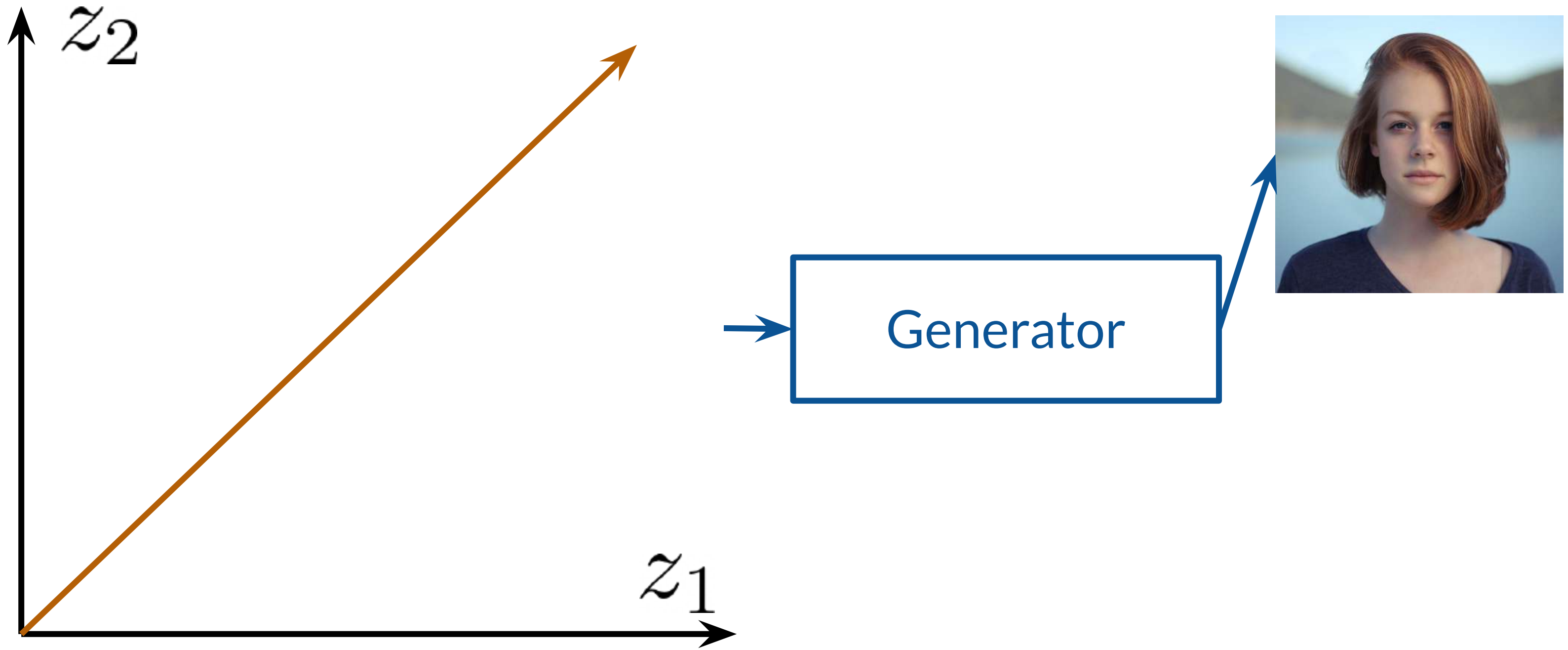
$g(v_1) \longleftrightarrow g(v_2)$

Z-Space and Controllable Generation



Z-Space with Noise Vectors

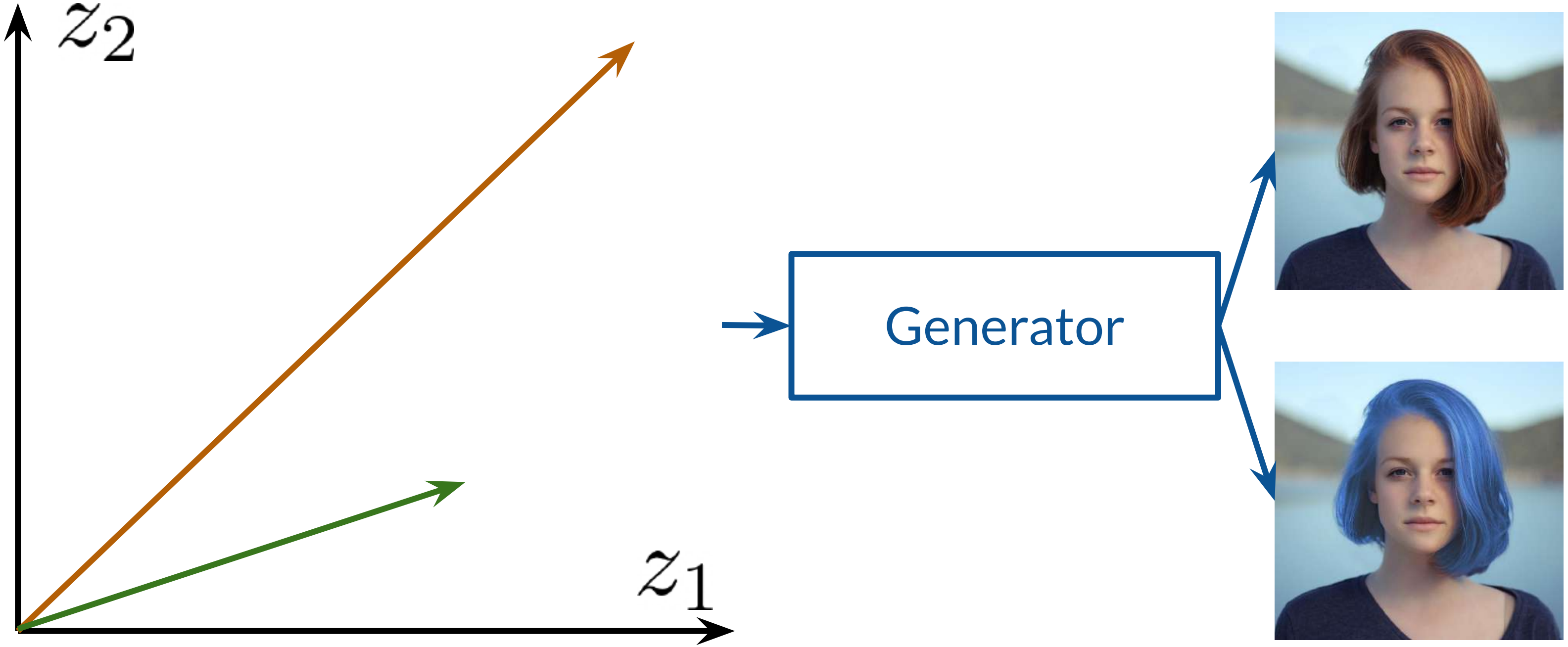
Z-Space and Controllable Generation



Z-Space with Noise Vectors

Output

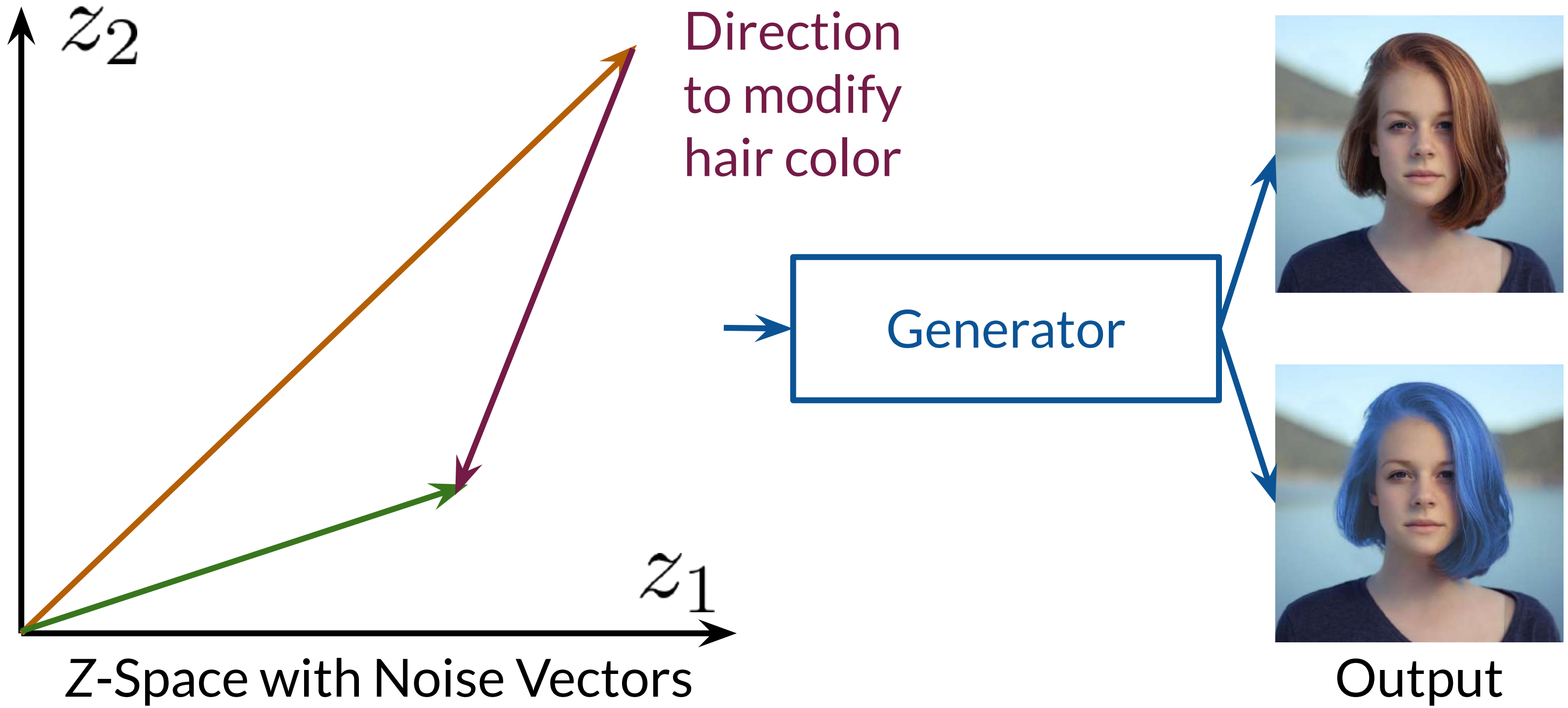
Z-Space and Controllable Generation



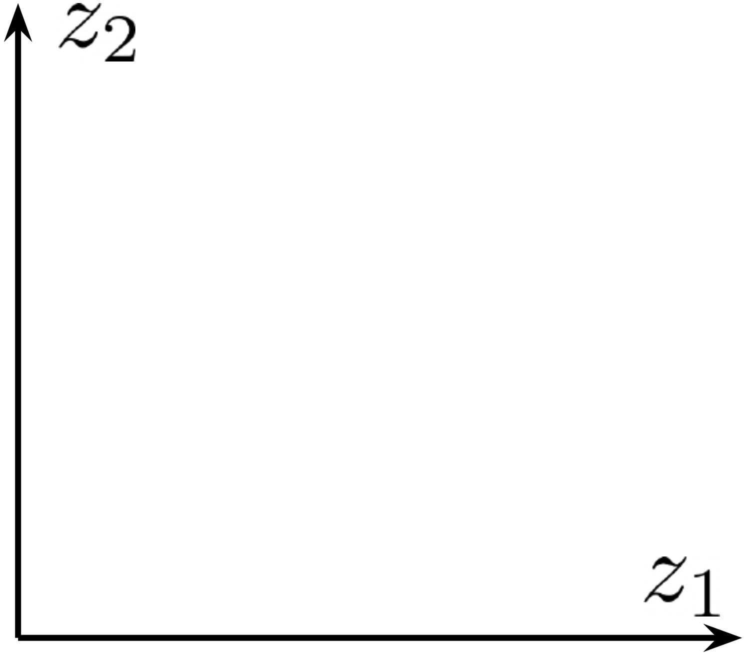
Z-Space with Noise Vectors

Output

Z-Space and Controllable Generation

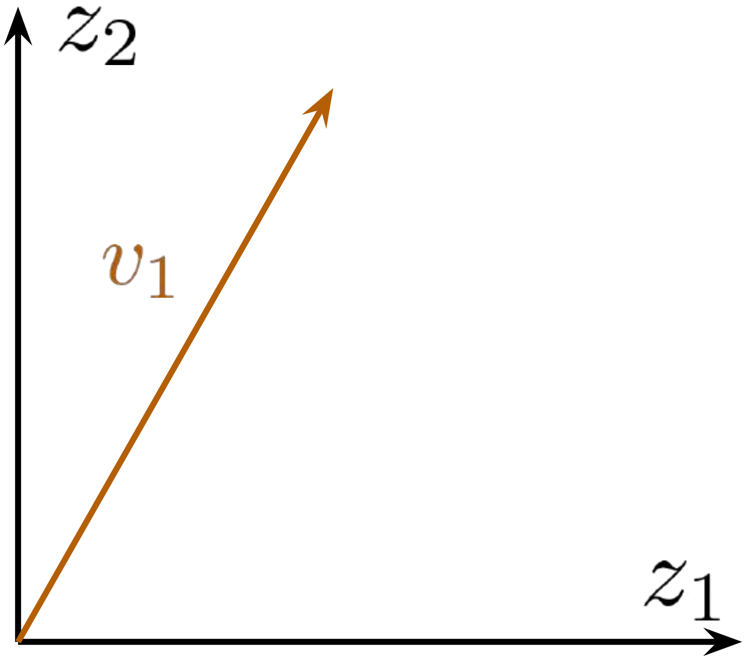


Z-Space and Controllable Generation



Z-Space with Noise Vectors

Z-Space and Controllable Generation

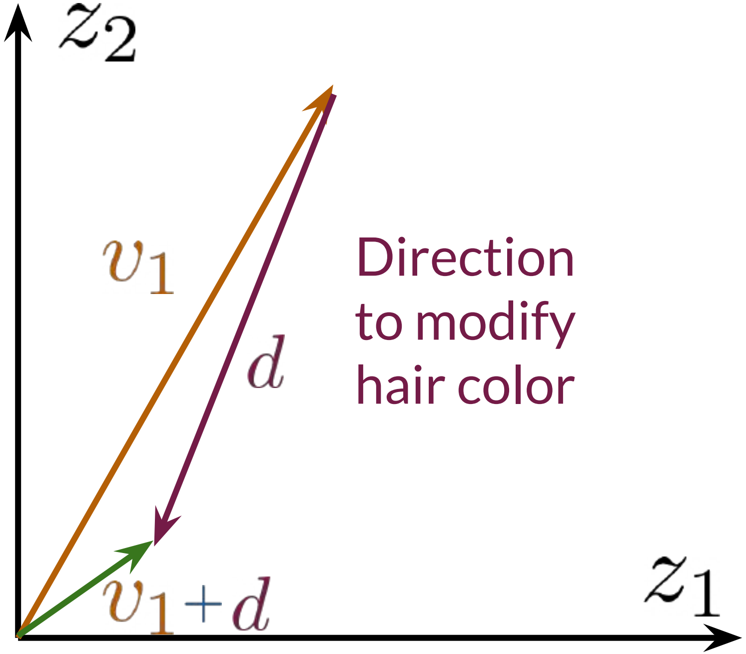


Z-Space with Noise Vectors

Original output

$g(v_1)$

Z-Space and Controllable Generation

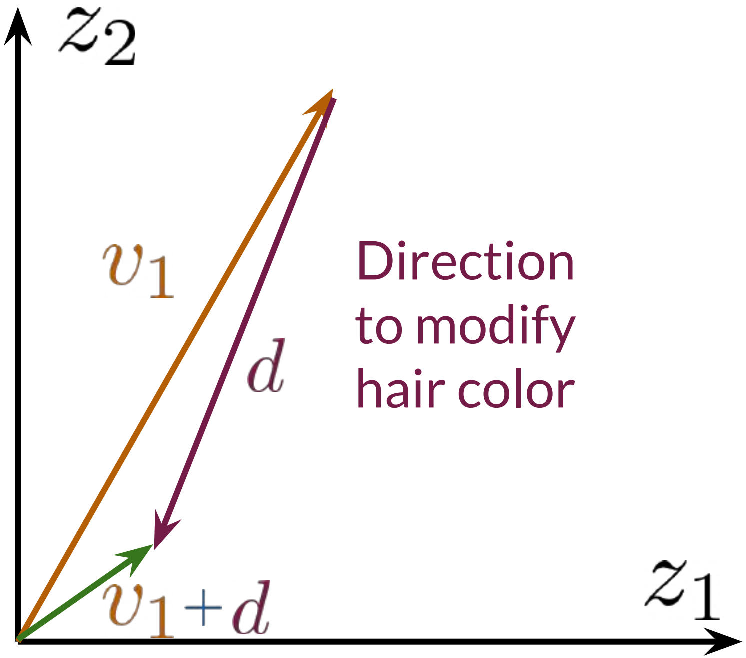


Z-Space with Noise Vectors

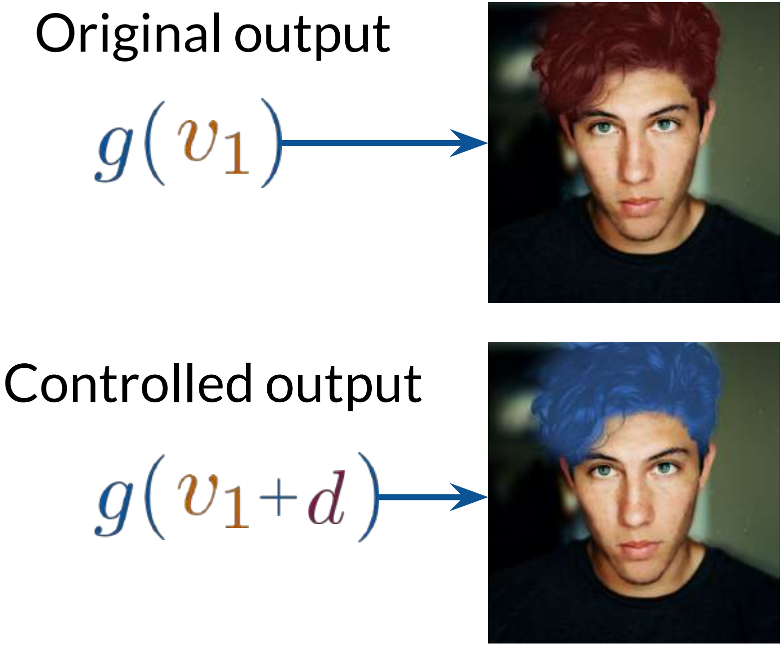
Original output

$g(v_1)$

Z-Space and Controllable Generation



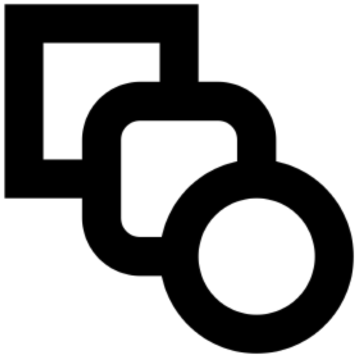
Z-Space with Noise Vectors



deeplearning.ai

Summary

- To control output features, you need to find directions in the Z-space
- To modify your output, you move around in the Z-space



Challenges with Controllable Generation

deeplearning.ai

Outline

- Output feature correlation
- *Z*-space entanglement

Feature Correlation

Uncorrelated
Features

Add beard

Feature Correlation

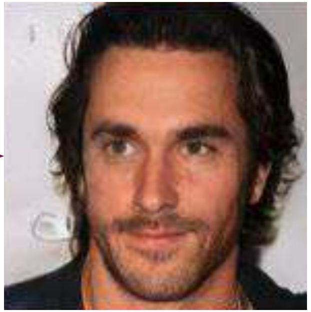
Uncorrelated Features

Add beard

Correlated Features

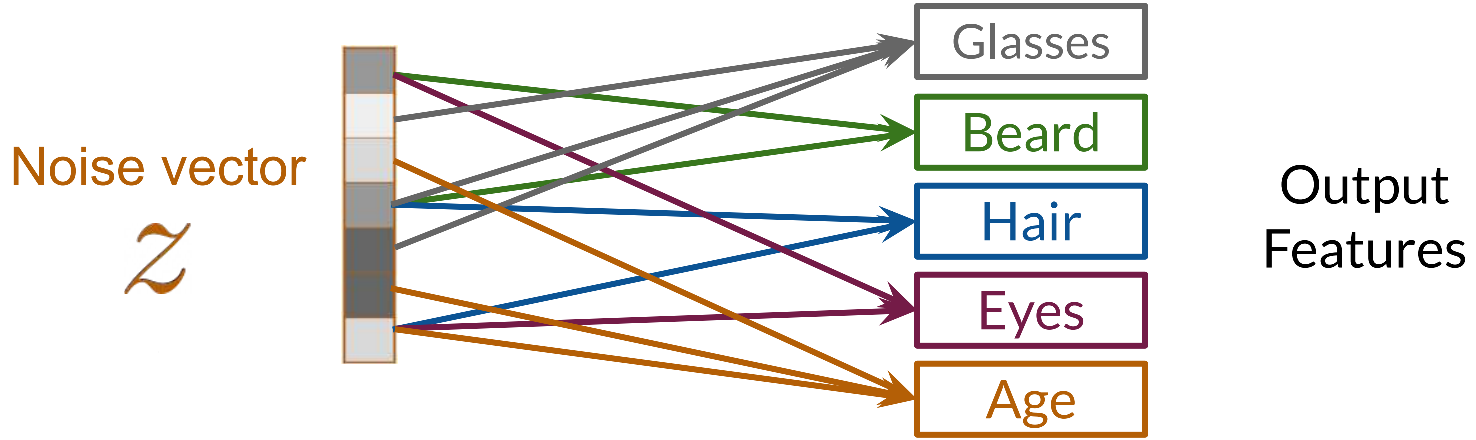
Add beard

Make more masculine

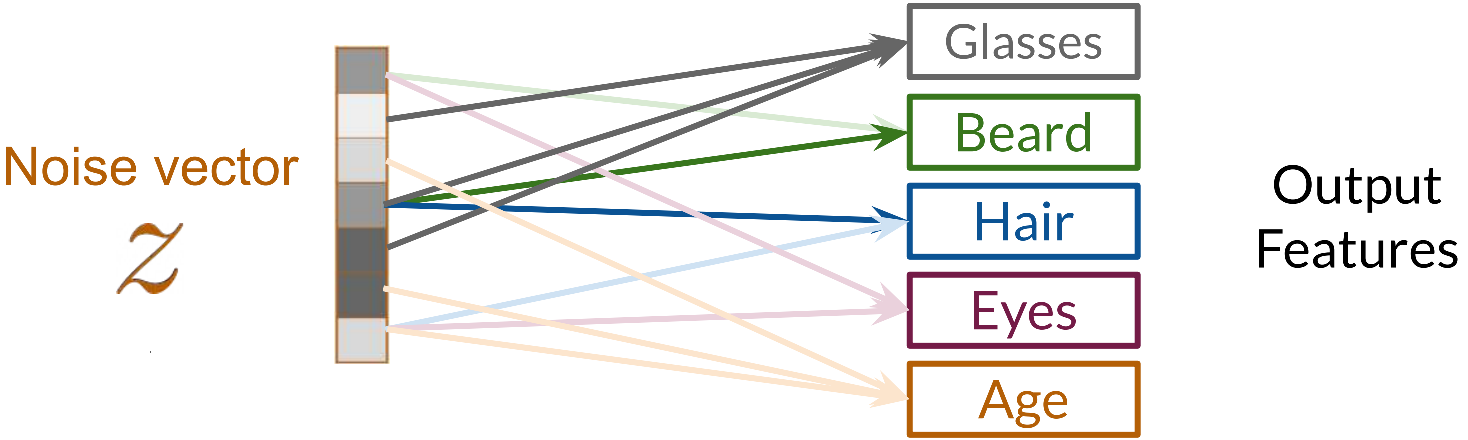


Z-Space Entanglement

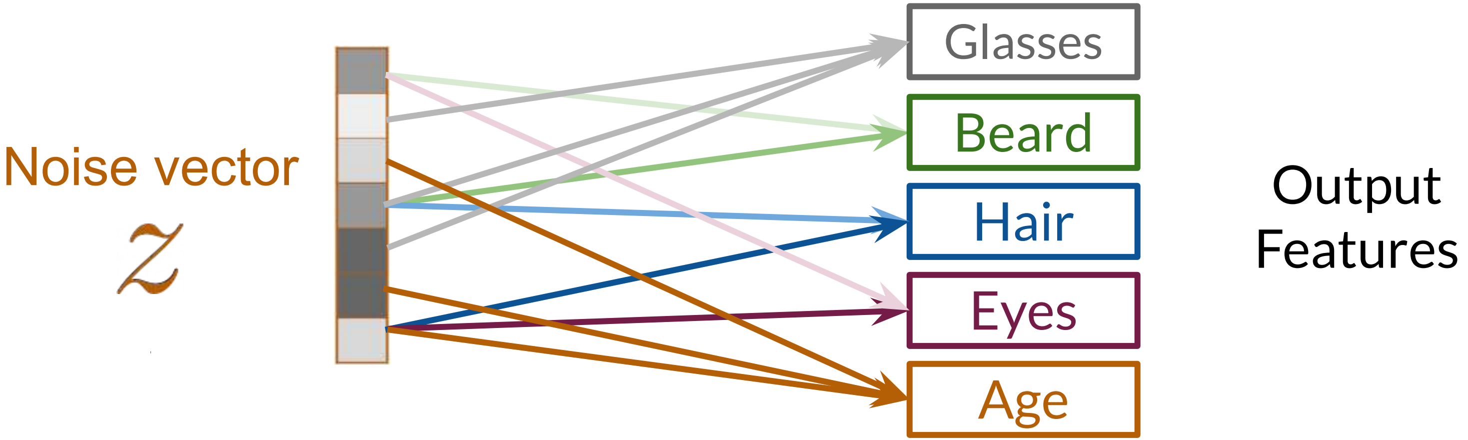
Z-Space Entanglement



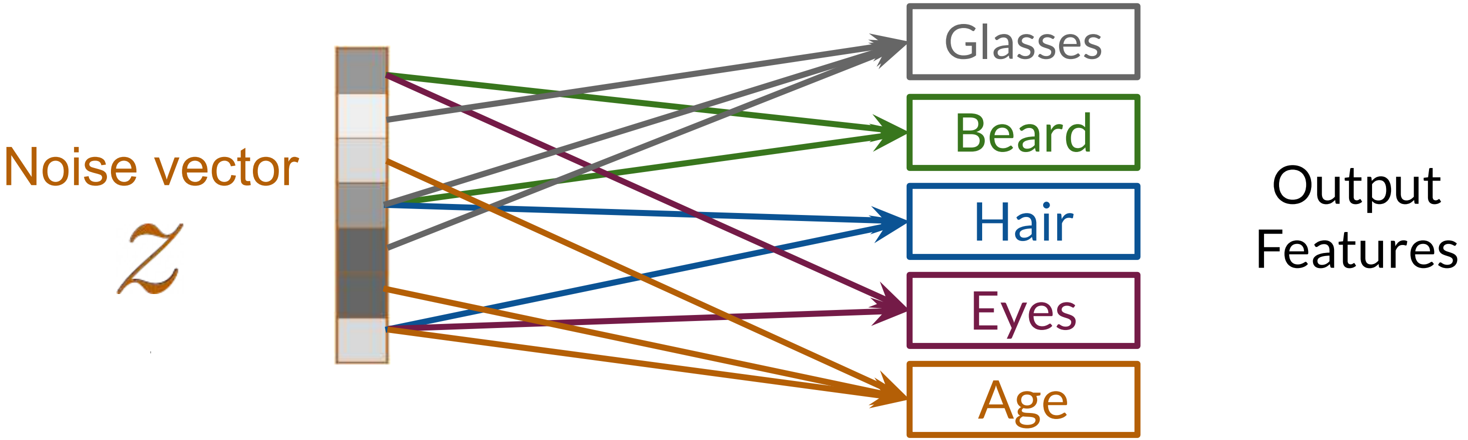
Z-Space Entanglement



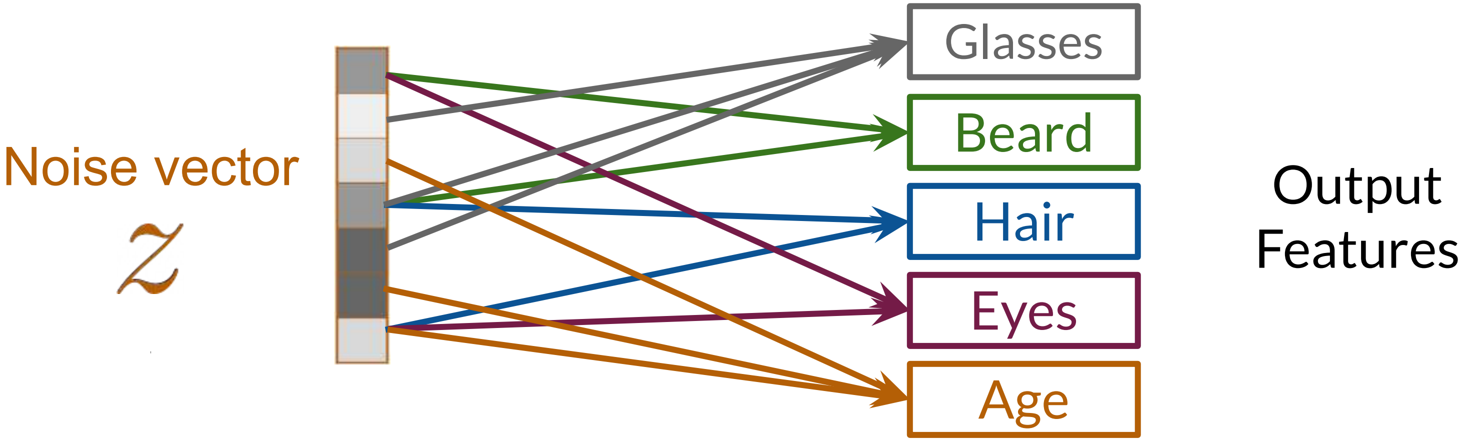
Z-Space Entanglement



Z-Space Entanglement

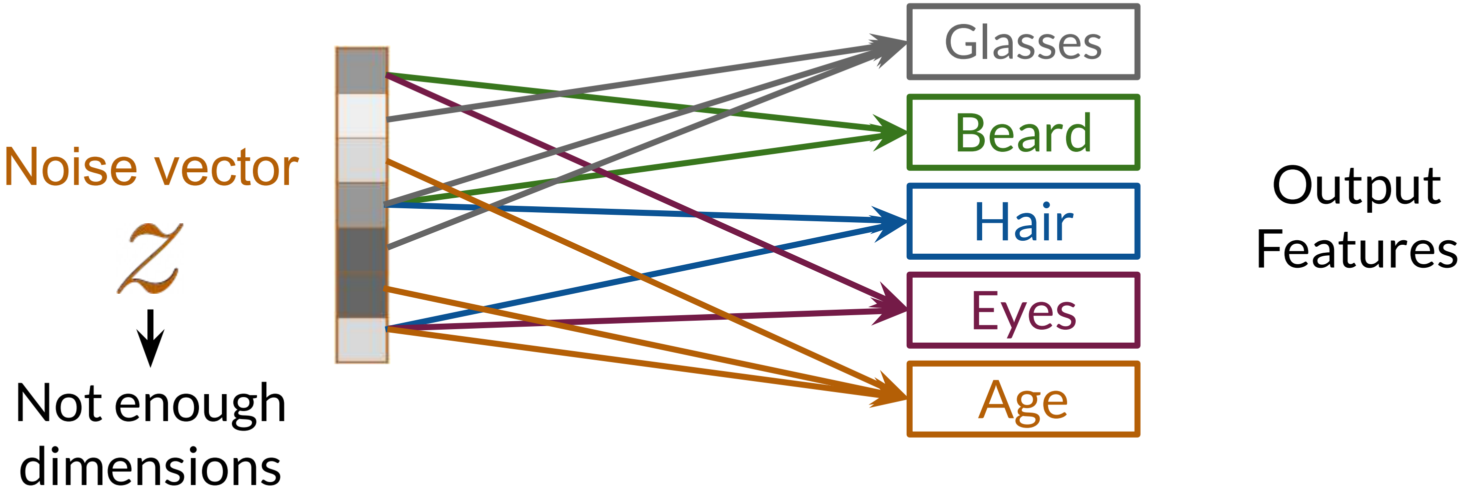


Z-Space Entanglement



It is not possible to control single output features

Z-Space Entanglement



It is not possible to control single output features

Summary

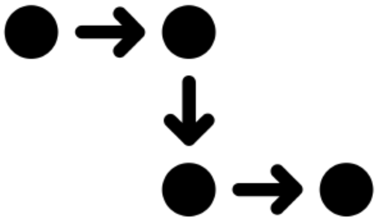
- When trying to control one feature, others that are correlated change
- *Z*-space entanglement makes controllability difficult, if not impossible
- Entanglement happens when *z* does not have enough dimensions

deeplearning.ai

Classifier Gradients

Outline

- How to use classifiers to find directions in the Z-space
- Requirements to use this method



Classifier Gradients

Classifier Gradients

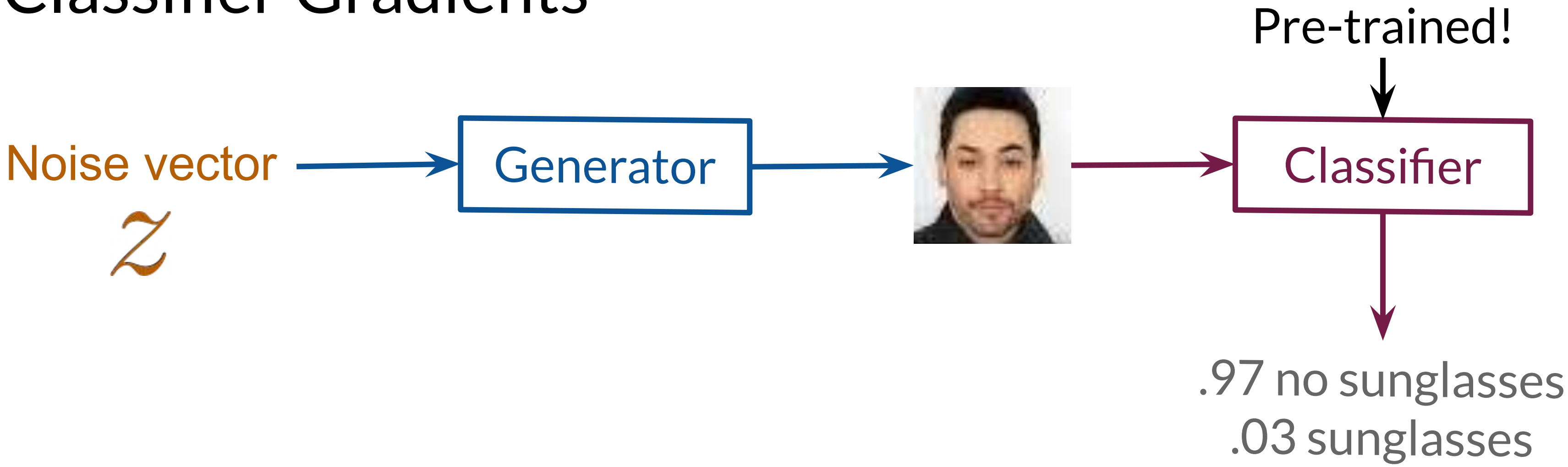
Noise vector

z

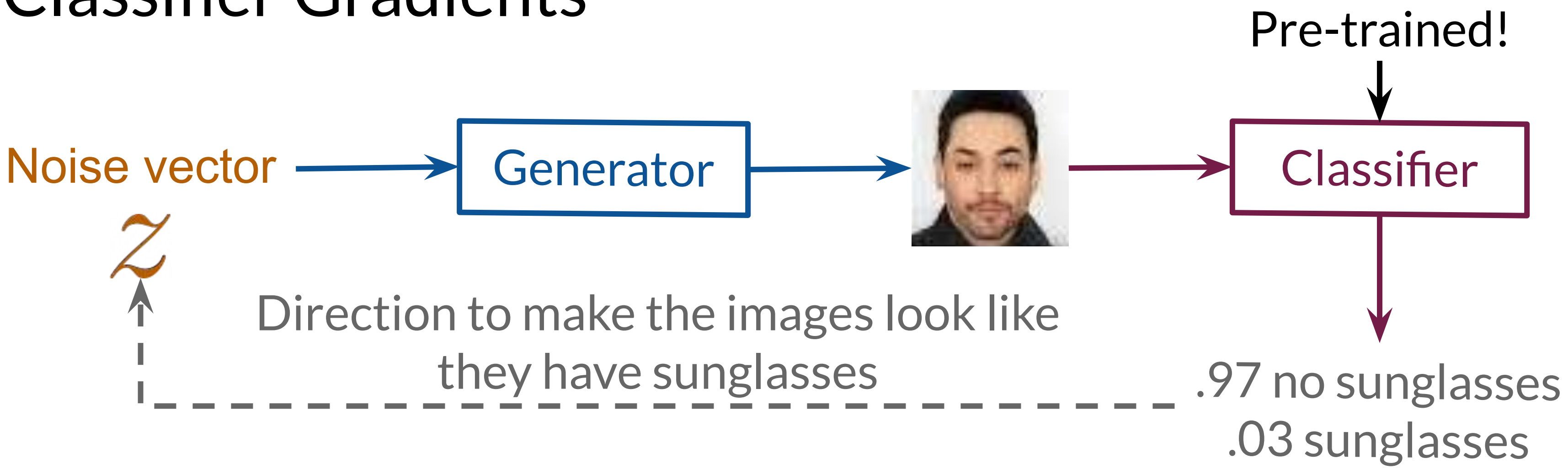
Classifier Gradients

Noise vector z → Generator →

Classifier Gradients



Classifier Gradients



Classifier Gradients

Pre-trained!

Noise vector → Generator →  → Classifier

z

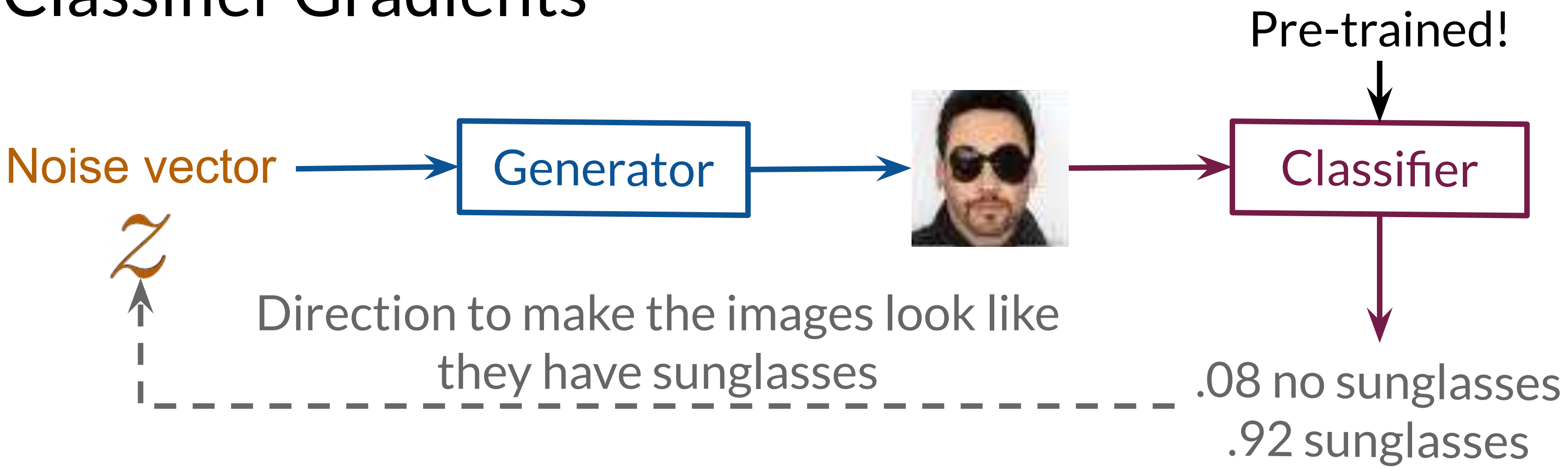
Direction to make the images look like they have sunglasses

.25 no sunglasses
.75 sunglasses

Modify **just** the **noise vector** until the feature emerges

deeplearning.ai

Classifier Gradients



Modify **just** the noise vector until the feature emerges

deeplearning.ai

Summary

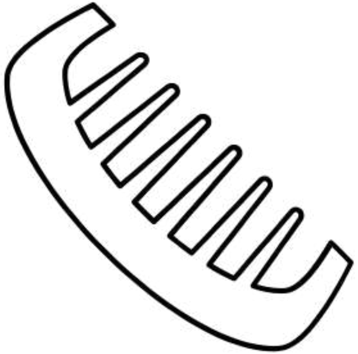
- Classifiers can be used to find directions in the Z-space
- To find directions, the updates are done just to the noise vector

deeplearning.ai

Disentanglement

Outline

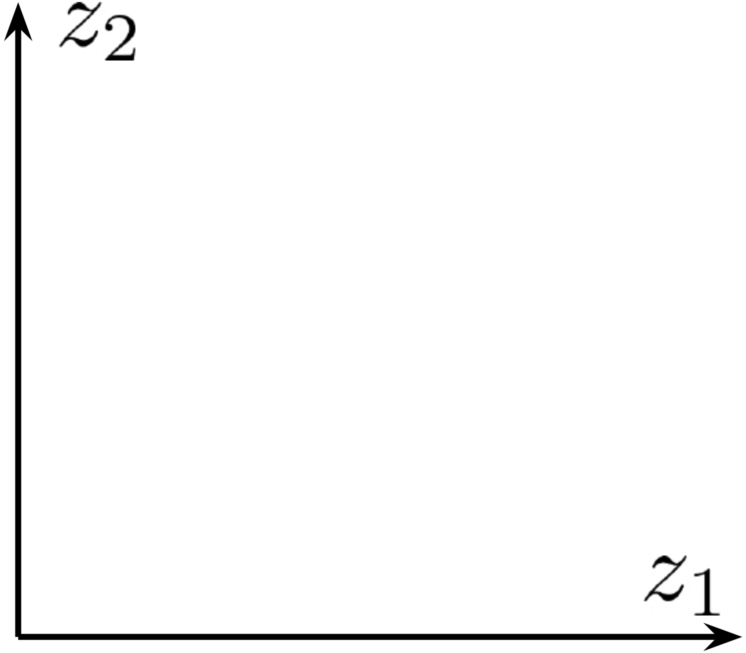
- What a disentangled Z-space means
- Ways to encourage disentangled Z-spaces



Disentangled Z-Space

$$v_1 = [\ 1,\ 2,\ 3, \ldots\ ]$$

$$v_2 = [\ 5,\ 6,\ 7, \ldots\ ]$$

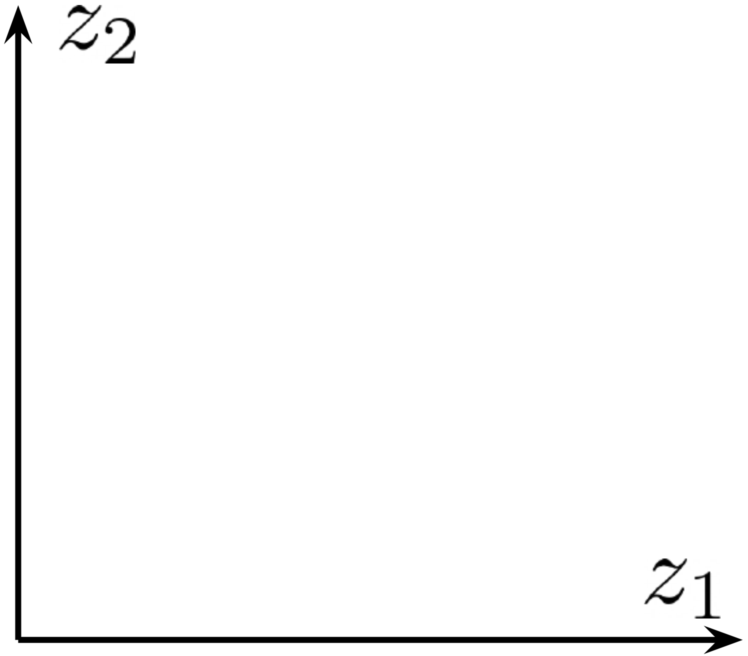


Z-Space with Noise Vectors

Disentangled Z-Space

$$\begin{array}{ll} & \quad z_1 \\ v_1 = & [\ 1\ ,\ 2\ ,\ 3\ ,\ \dots\ ] \\ v_2 = & [\ 5\ ,\ 6\ ,\ 7\ ,\ \dots\ ] \end{array}$$

Hair color



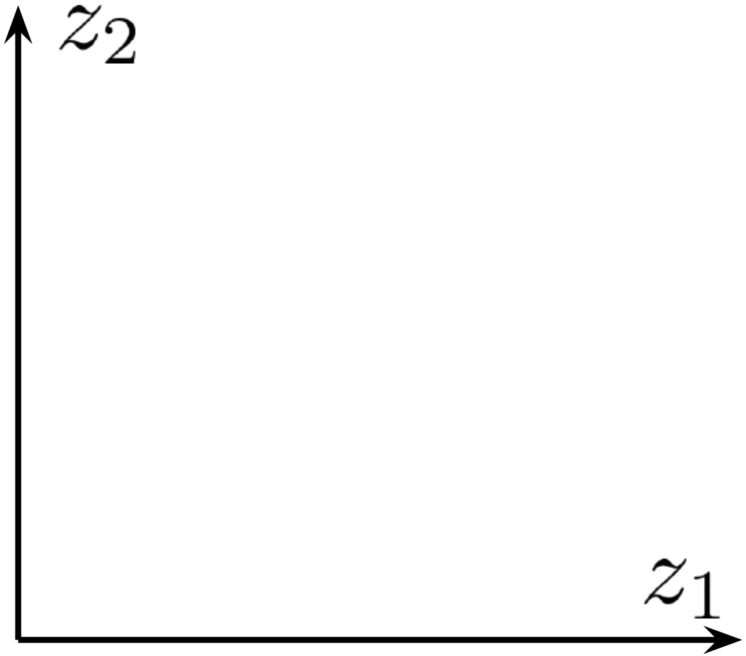
Z-Space with Noise Vectors

Disentangled Z-Space

$$\begin{array}{lccl} & z_1 & z_2 & \\ v_1 = [& 1, & 2, & 3, \ldots] \\ v_2 = [& 5, & 6, & 7, \ldots] \end{array}$$

Hair color

Hair length



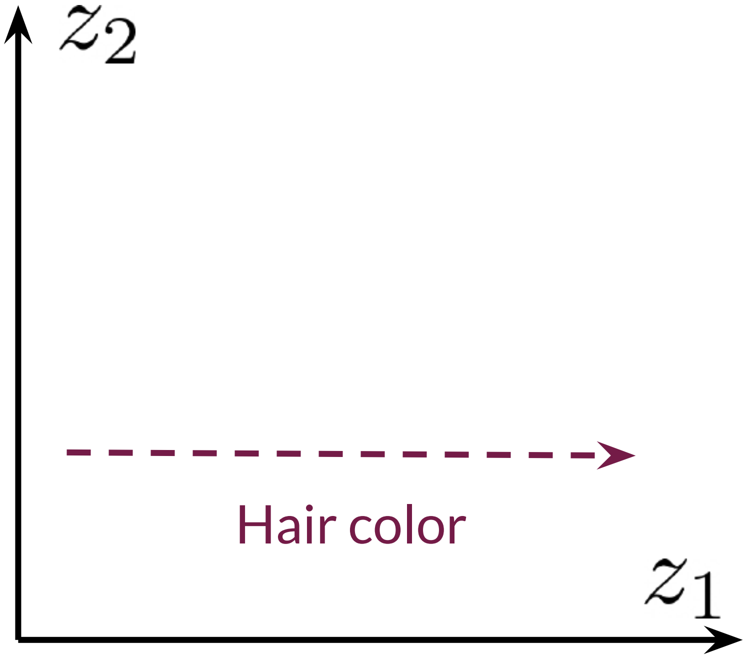
Z-Space with Noise Vectors

Disentangled Z-Space

$$\begin{array}{cccc} & z_1 & z_2 & \\ v_1 = [& 1, & 2, & 3, \ldots] \\ v_2 = [& 5, & 6, & 7, \ldots] \end{array}$$

Hair color

Hair length



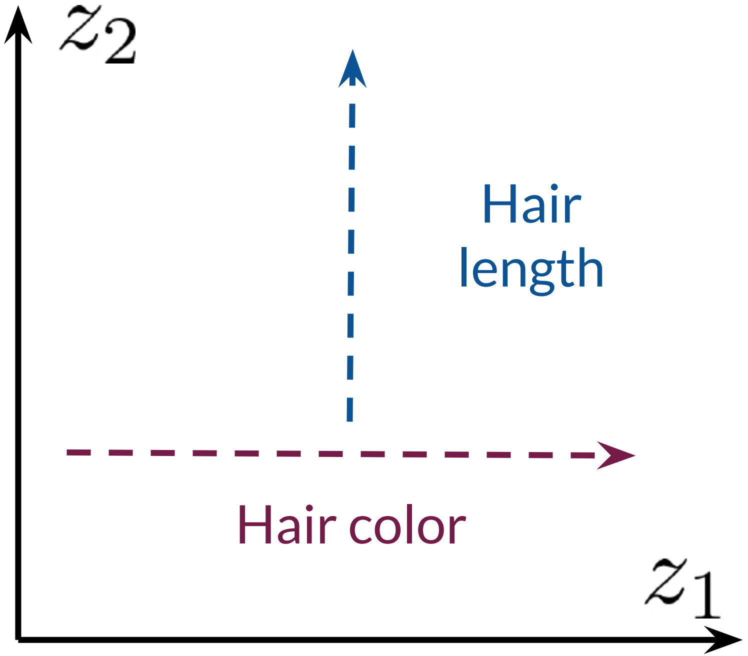
Z-Space with Noise Vectors

Disentangled Z-Space

$$\begin{array}{cccc} & z_1 & z_2 & \\ v_1 = [& 1, & 2, & 3, \dots] \\ v_2 = [& 5, & 6, & 7, \dots] \end{array}$$

Hair color

Hair length



Z-Space with Noise Vectors

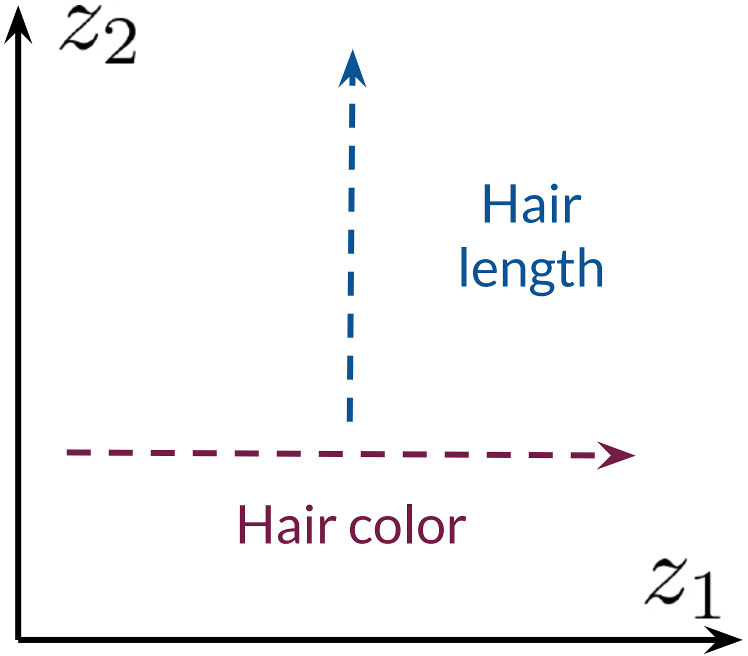
Disentangled Z-Space

$$\begin{array}{cccccc} & & z_1 & z_2 & & \\ v_1 & = [& 1, & 2, & 3, \ldots &] \\ v_2 & = [& 5, & 6, & 7, \ldots &] \end{array}$$

Hair color

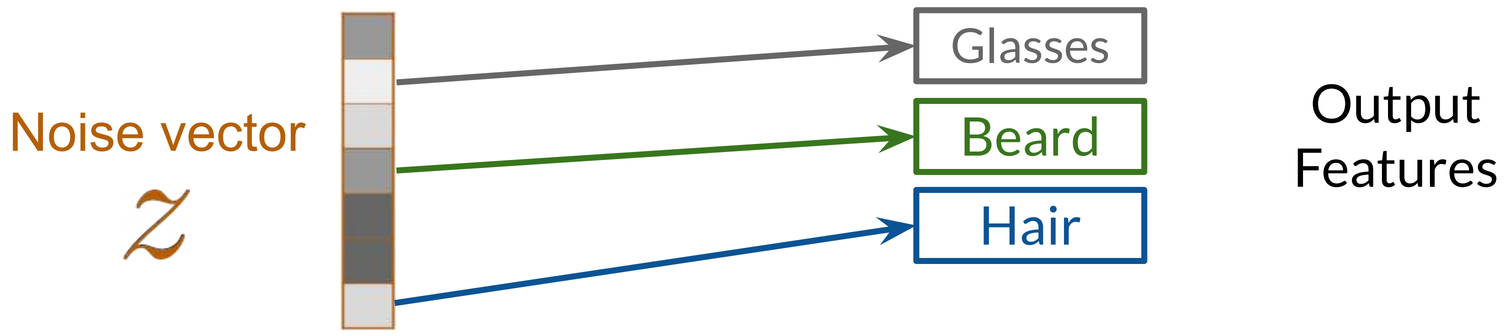
Hair length

Latent factors of variation

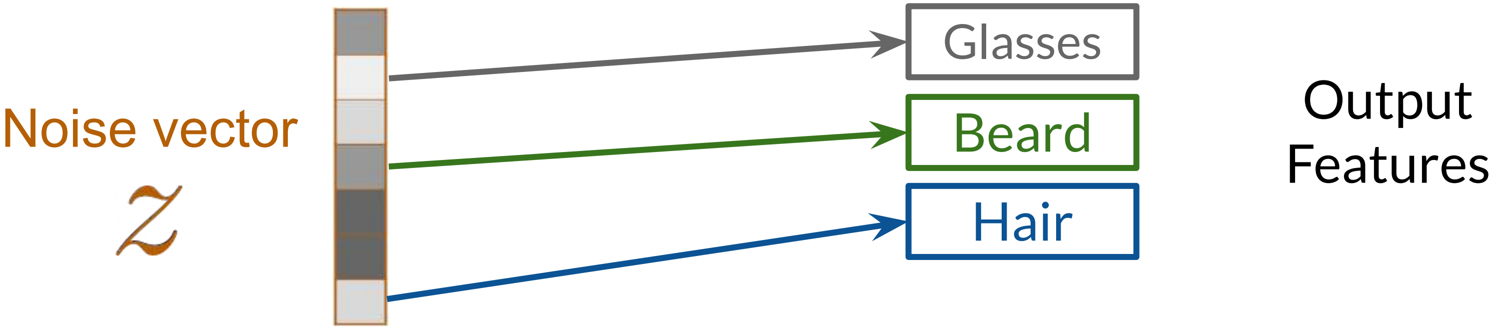


Z-Space with Noise Vectors

Disentangled Z-Space

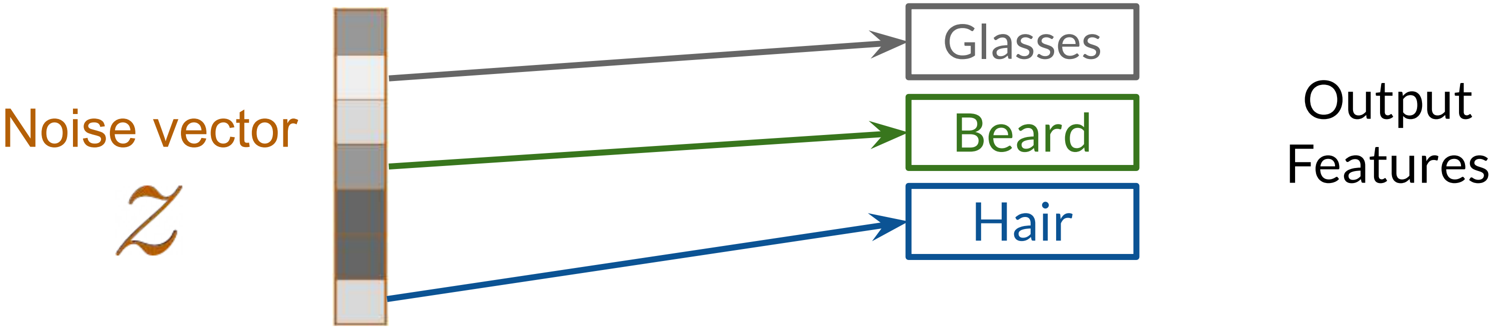


Disentangled Z-Space

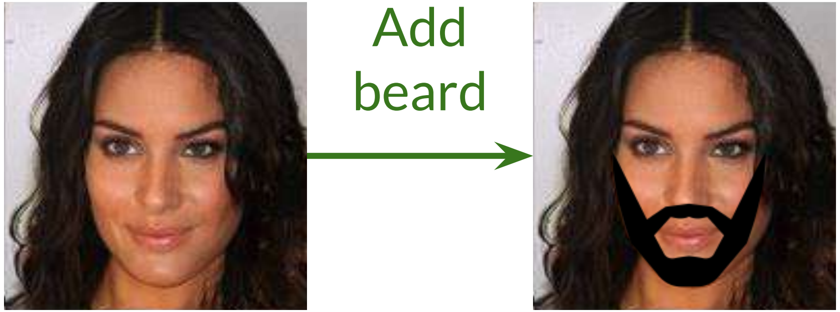


Changes to one feature don't affect the others

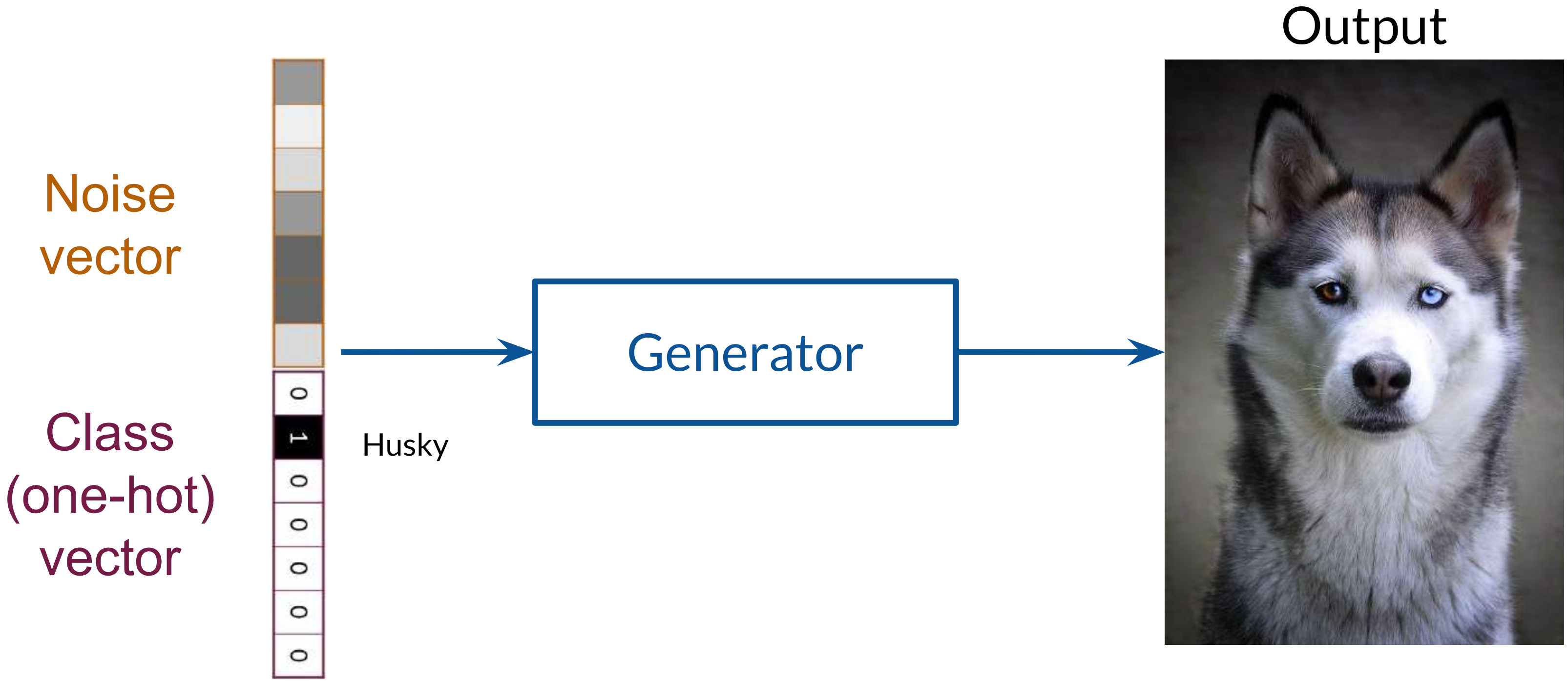
Disentangled Z-Space



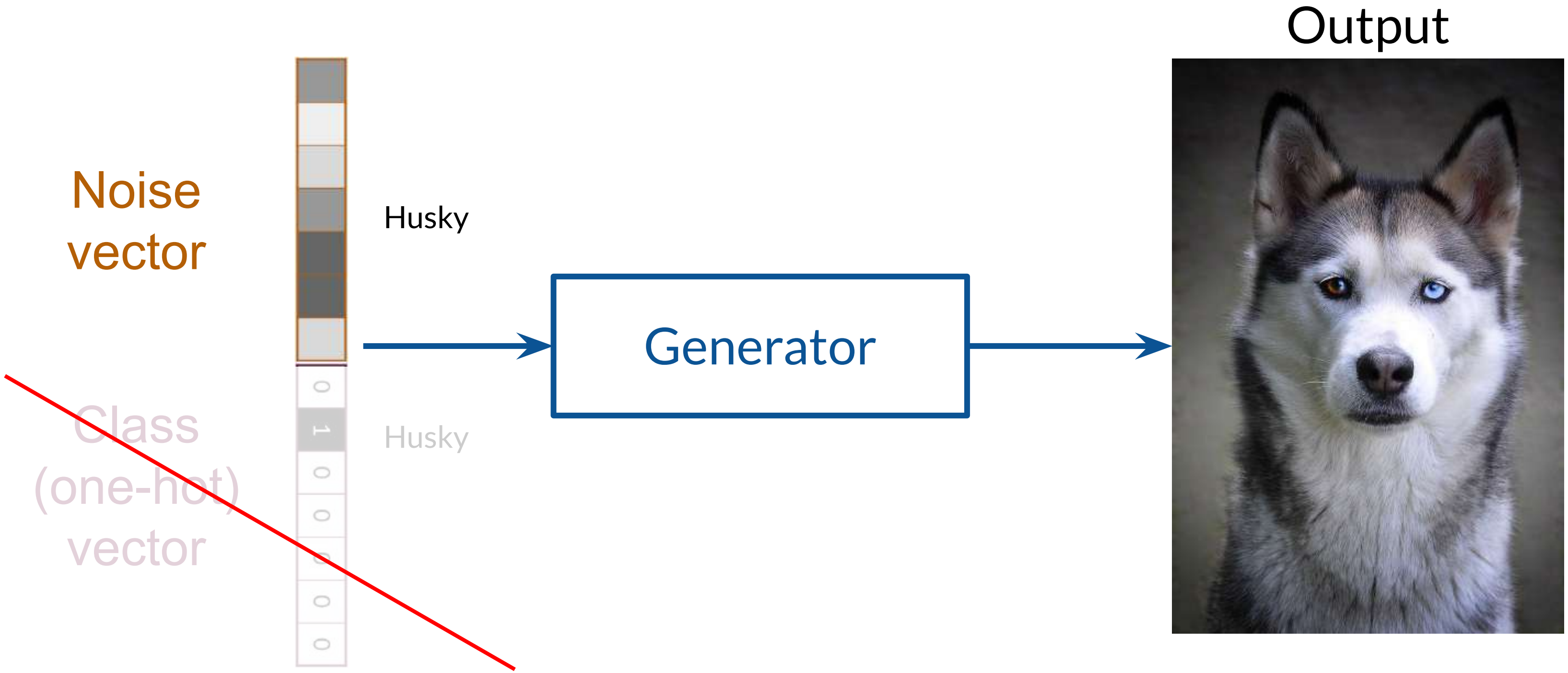
Changes to one feature don’t affect the others



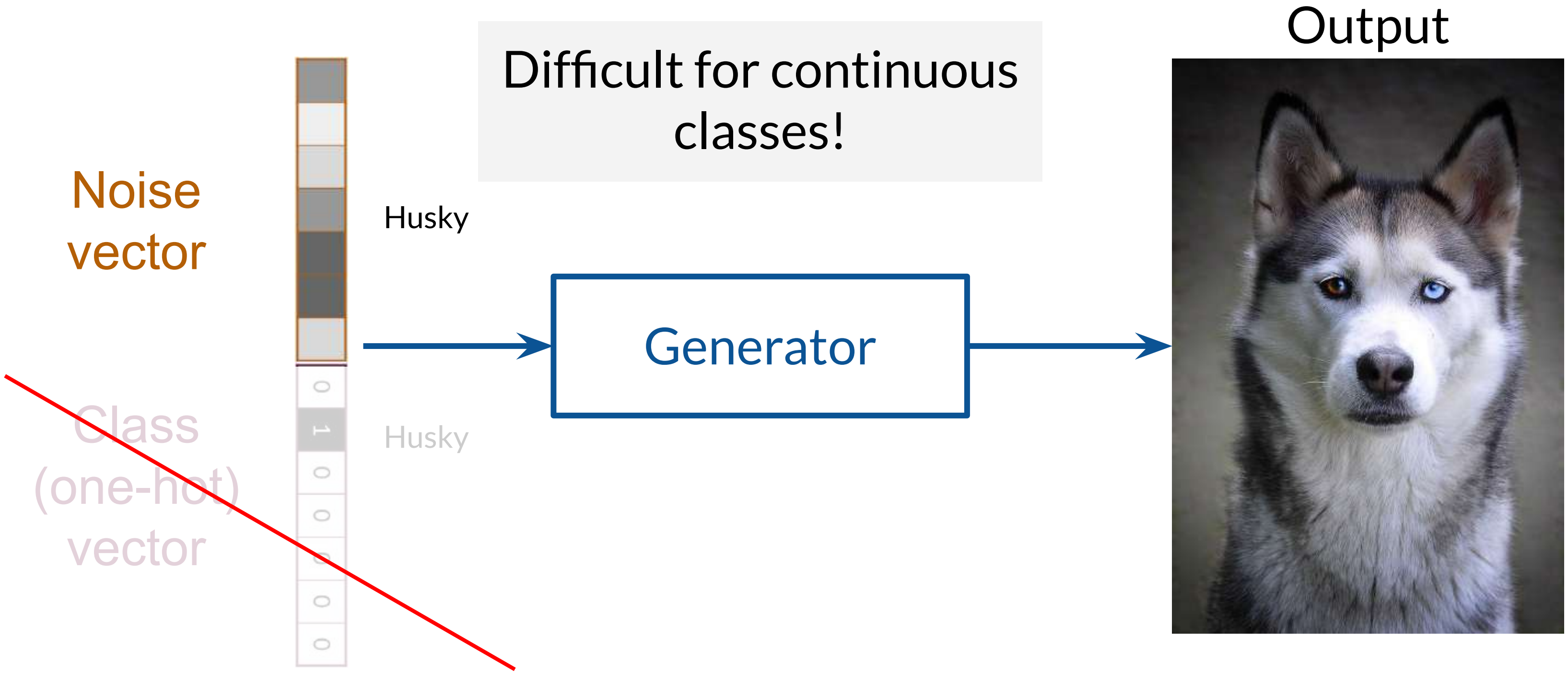
Encourage Disentanglement: Supervision



Encourage Disentanglement: Supervision



Encourage Disentanglement: Supervision



Encourage Disentanglement: Loss Function

$$v_1 = [\ 1,\ 2,\ 3, \ldots\ ]$$

$$v_2 = [\ 5,\ 6,\ 7, \ldots\ ]$$

Encourage Disentanglement: Loss Function

$$L_{\text{new}} = \underbrace{\boxed{L_{\text{original}}}}_{\text{Original loss}} + \underbrace{\boxed{\text{reg}_d}}_{\text{Regularization}}$$

$$v_1 = [\ 1,\ 2,\ 3, \ldots\ ]$$

$$v_2 = [\ 5,\ 6,\ 7, \ldots\ ]$$

Encourage Disentanglement: Loss Function

$$L_{\text{new}} = \underbrace{\boxed{L_{\text{original}}}}_{\text{Original loss}} + \underbrace{\boxed{\text{reg}_d}}_{\text{Regularization}}$$

$$v_1 = [\ 1\ ,\ 2\ ,\ 3\ , \ldots\ ]$$

$$v_2 = [\ 5\ ,\ 6\ ,\ 7\ , \ldots\ ]$$

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

Encourage Disentanglement: Loss Function

$$\begin{array}{ccccc} & z_1 & z_2 & & \\ v_1 = [& 1, & 2, & 3, ... &] \\ v_2 = [& 5, & 6, & 7, ... &] \end{array}$$

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

Original loss

Regularization

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

Encourage Disentanglement: Loss Function

$$\begin{array}{lcc} & z_1 & z_2 & & \\ v_1 = [& 1, & 2, & 3, \ldots &] \\ v_2 = [& 5, & 6, & 7, \ldots &] \end{array}$$

Output feature #1

Output feature #2

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

Original loss

Regularization

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

Summary

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