Welcome to COMP 590/790 Neural Rendering

Soumyadip Sengupta

Course Website:
Scan Me!

Today's Plan

- Introduction
- Motivation for the course
- Course Overview
- Course expectation (grading, policies, etc.)
- Introduction to Deep Neural Networks

Today’s Plan

- Introduction
- Motivation for the course
- Course Overview
- Course expectation (grading, policies, etc.)
- Introduction to Deep Neural Networks

Soumyadip (Roni) Sengupta

(he/him)

- PhD @ University of Maryland
- Postdoc @ University of Washington
- Asst. Prof @ UNC starting July 2022!

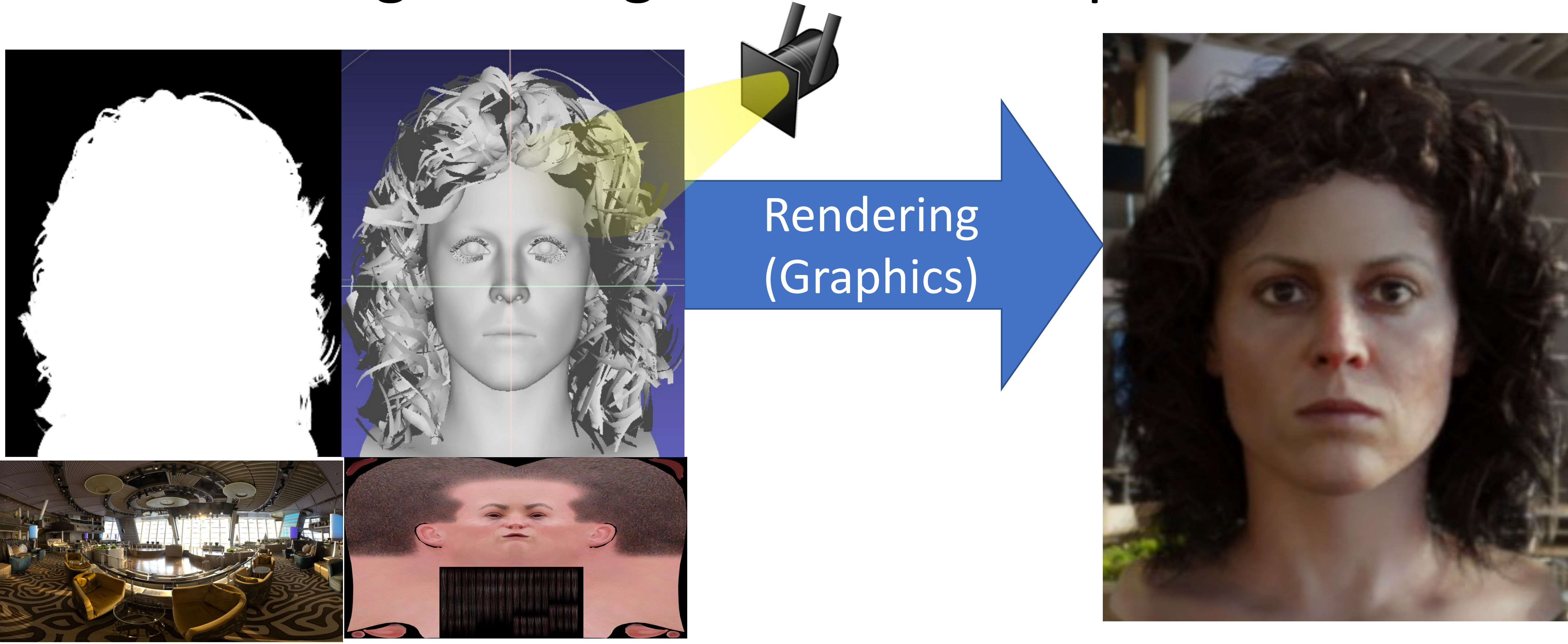
About You

- Name (feel free to let us know your pronouns)
- Program (BA/BS/MS/PhD which year?) & 590/790?
- (Optional) Your research/academic interest?

Today’s Plan

- Introduction
- Motivation for the course
- Course Overview
- Course expectation (grading, policies, etc.)
- Introduction to Deep Neural Networks

What is Rendering?
Creating an image from 3D components



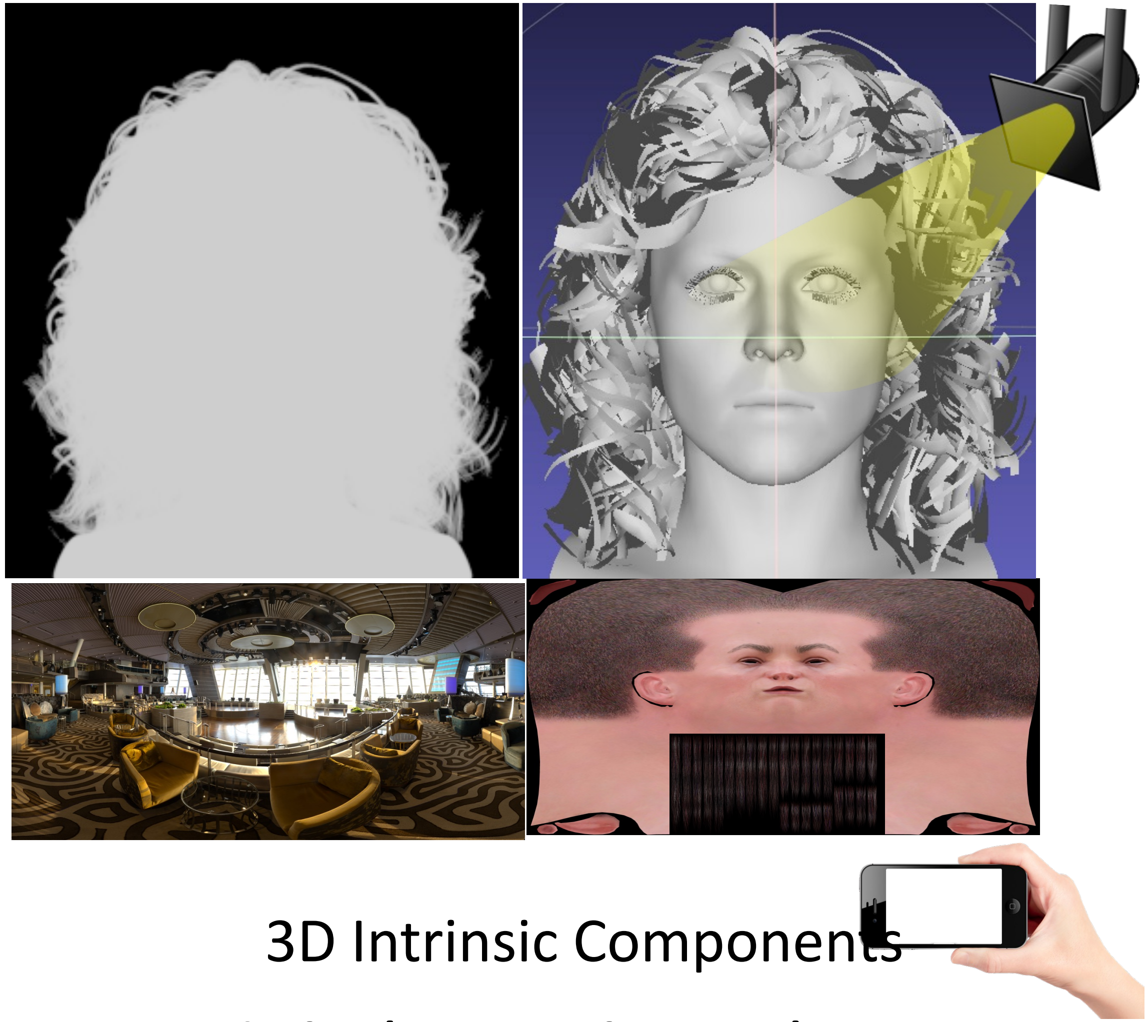
3D Intrinsic Components/ Assests

Computer Vision (Inverse Graphics):
Understanding the physical world in 3D from an image

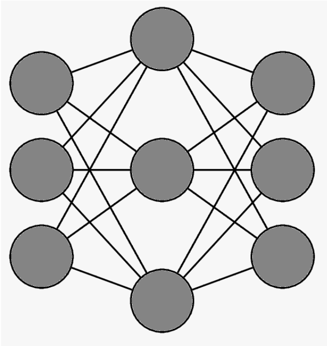
3D Intrinsic Components

Inverse Graphics with Machine Learning

ill-posed and under-constrained



3D Intrinsic Components



ML models (neural nets) learning priors from data

How does Computer Vision & Graphics work together?

Current Image

Vision

3D Intrinsic Components

Edit Components

Graphics

New Image under different conditions

Change:

- Viewpoint
- Lighting
- Reflectance
- Background
- Attributes
- Many others…

Virtual Lighting

Image

Computer Vision

Reflectance

Geometry

Illumination

Pose

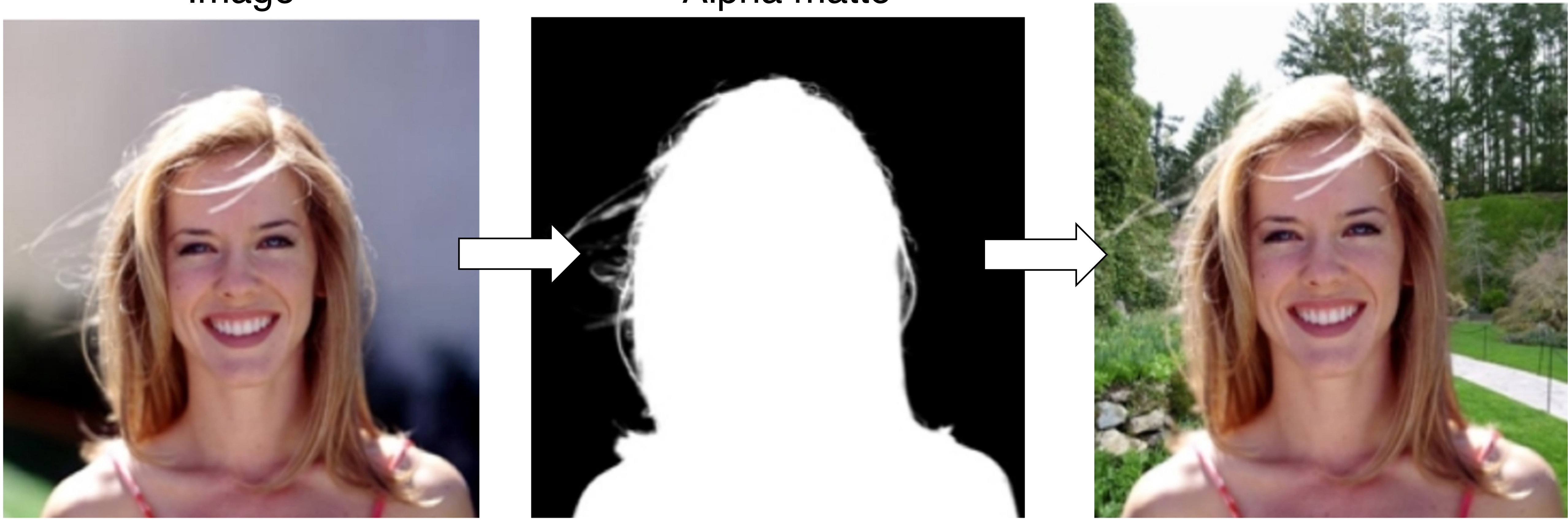
Edit: Lighting

"SfSNet", Sengupta et al, CVPR 2018.

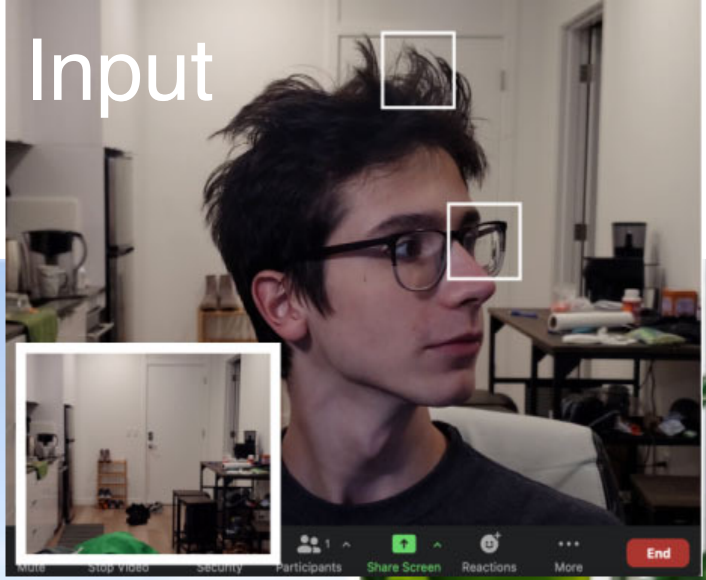
11

Virtual Background

Image Alpha matte Composed Image



“Background Matting”,
CVPR 2021, CVPR 2022

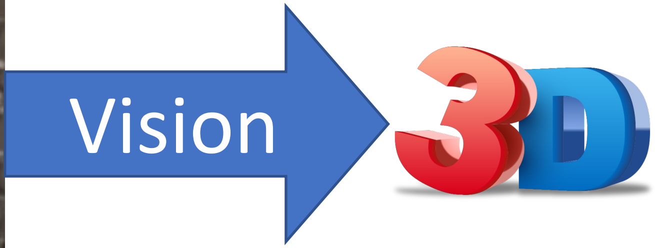


Zoom Virtual Green-Screen

Our Virtual Green-Screen

How do we define intrinsic 3D components?

Current Image



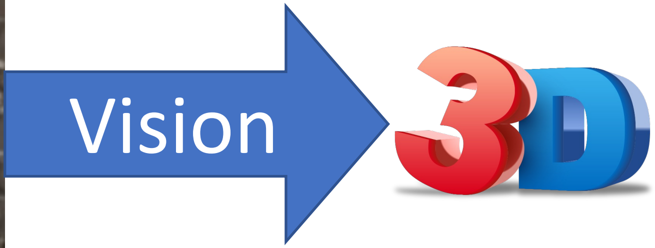
3D Intrinsic Components

Explicit (Intro to Computer Graphics):
- Reconstruct 3D geometry (mesh, voxel, point cloud)
- Estimate reflectance function (BRDF)
- Estimate lighting function (spherical harmonics)

Implicit (Generative Models):
- All intrinsic components are represented as some latent features in some subspace of a neural network.

How do we define intrinsic 3D components?

Current Image



3D Intrinsic Components

Why implicit?

- Estimating explicit representation is a harder problem
- Often one might get better result by using an implicit representation and directly solving for the end goal.

Explicit Decomposition

Image

Reflectance

Geometry

Illumination

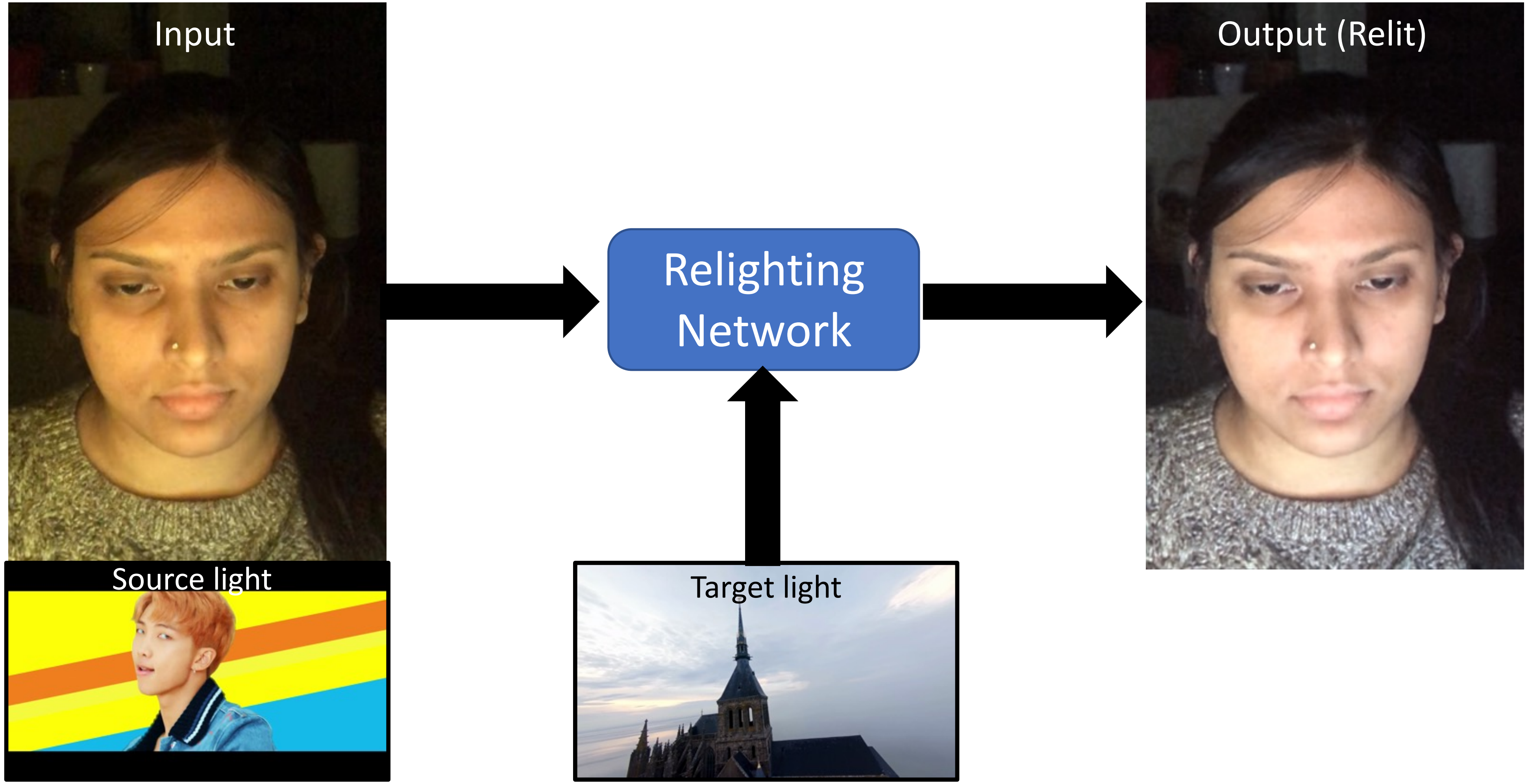
Relighting

× Explicit decomposition is hard

"SfSNet", S. Sengupta, A. Kanazawa, C. Castillo, D. Jacobs, CVPR 2018.

Implicit Decomposition

- No explicit decomposition.
- Directly learn a relighting function.



"A Light Stage on Every Desk", S. Sengupta, I. Kemelmacher, B. Curless, S. Seitz, ICCV 2021.

How does Computer Vision & Graphics work together?

Current Image

Vision → 3D → Edit Components → Graphics → New Image under different conditions

3D Intrinsic Components

Explicit: Reconstruct 3D
(Introduction to Graphics Lectures)

Implicit: Neural Representation
(Generative Models Lectures

Change:
- Viewpoint
- Lighting
- Reflectance
- Background
- Attributes
- Many others…

What is Neural Rendering?

Loosely defined as:

Use a neural network to generate new image/video/3D from an already captured image/video/3D.

Advances in Neural Rendering

SIGGRAPH 2021 Course

What is Neural Rendering?

Neural rendering is a new class of deep image and video generation approaches that enable **explicit or implicit control of scene properties** such as illumination, camera parameters, pose, geometry, appearance, and semantic structure. It combines **generative machine learning** techniques with **physical knowledge from computer graphics** to obtain controllable and photo-realistic outputs.

Today’s Plan

- Introduction
- Motivation for the course
- Course Overview
- Course expectation (grading, policies, etc.)
- Introduction to Deep Neural Networks

Part 1: 2D Neural Rendering

- Introduction to Computer Graphics (2 lectures)
  - How do we represent geometry, camera, reflectance, lighting and create an image from all these component? (Rendering)
- Introduction to Generative Models (2 lectures)
  - PixelCNN, VAE, GANs
- Applications of GAN (1 lecture + 8 papers)
  - Pix2Pix, CycleGAN, StyleGAN
- Denoising Diffusion Models (2 lectures + 6 papers)
  - DALL-E, Imagen

Part 2: 3D Neural Rendering

- Introduction to Neural Radiance Fields (NeRFs)
    - 3 lectures + 12 papers

Today's Plan

- Introduction
- Motivation for the course
- Course Overview
- Course expectation (grading, policies, etc.)
- Introduction to Deep Neural Networks

Course Website

- https://www.cs.unc.edu/~ronisen/teaching/fall_2022/comp_790_neural_rendering_fall_2022.html

- You can find it in my personal webpage:

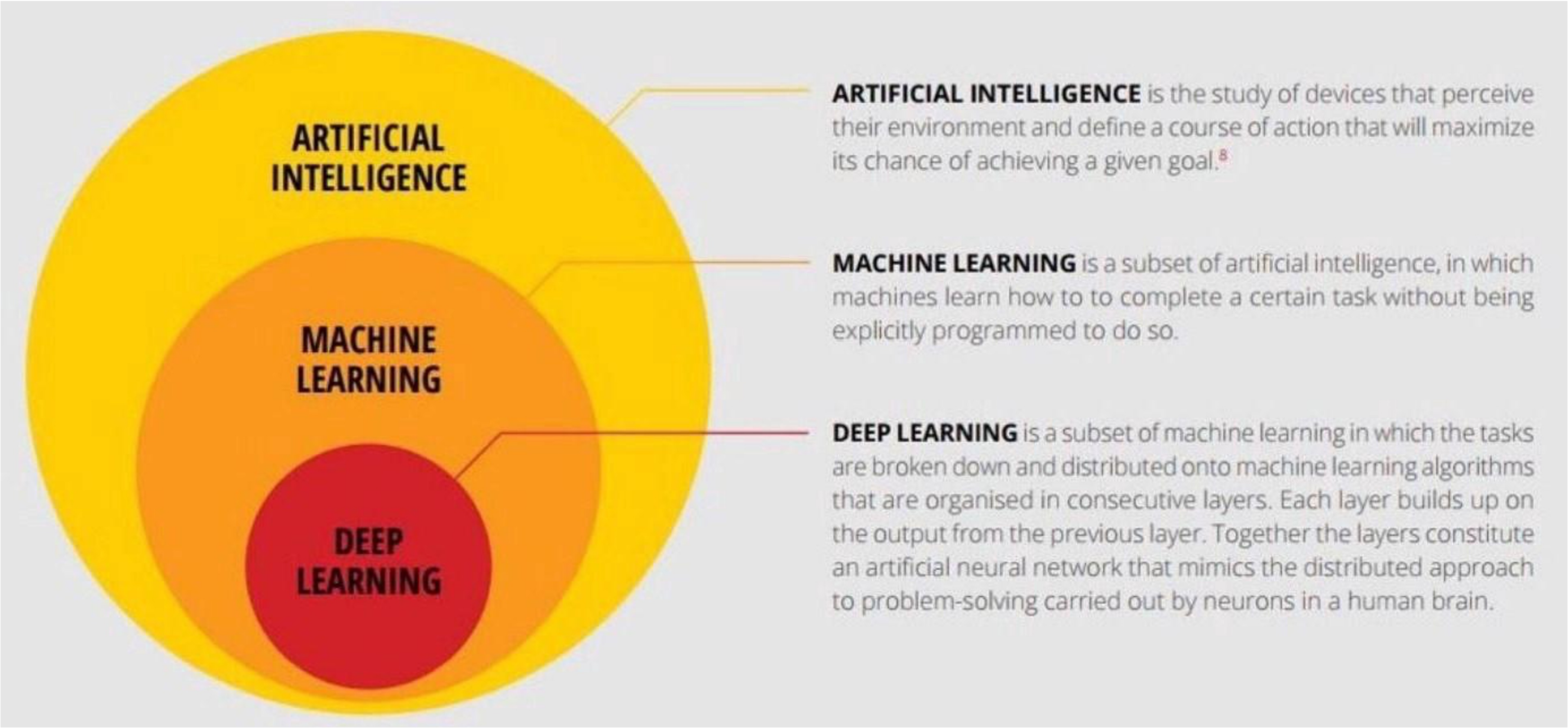
https://www.cs.unc.edu/~ronisen/ under the teaching tab!

Any questions/concerns about the syllabus?

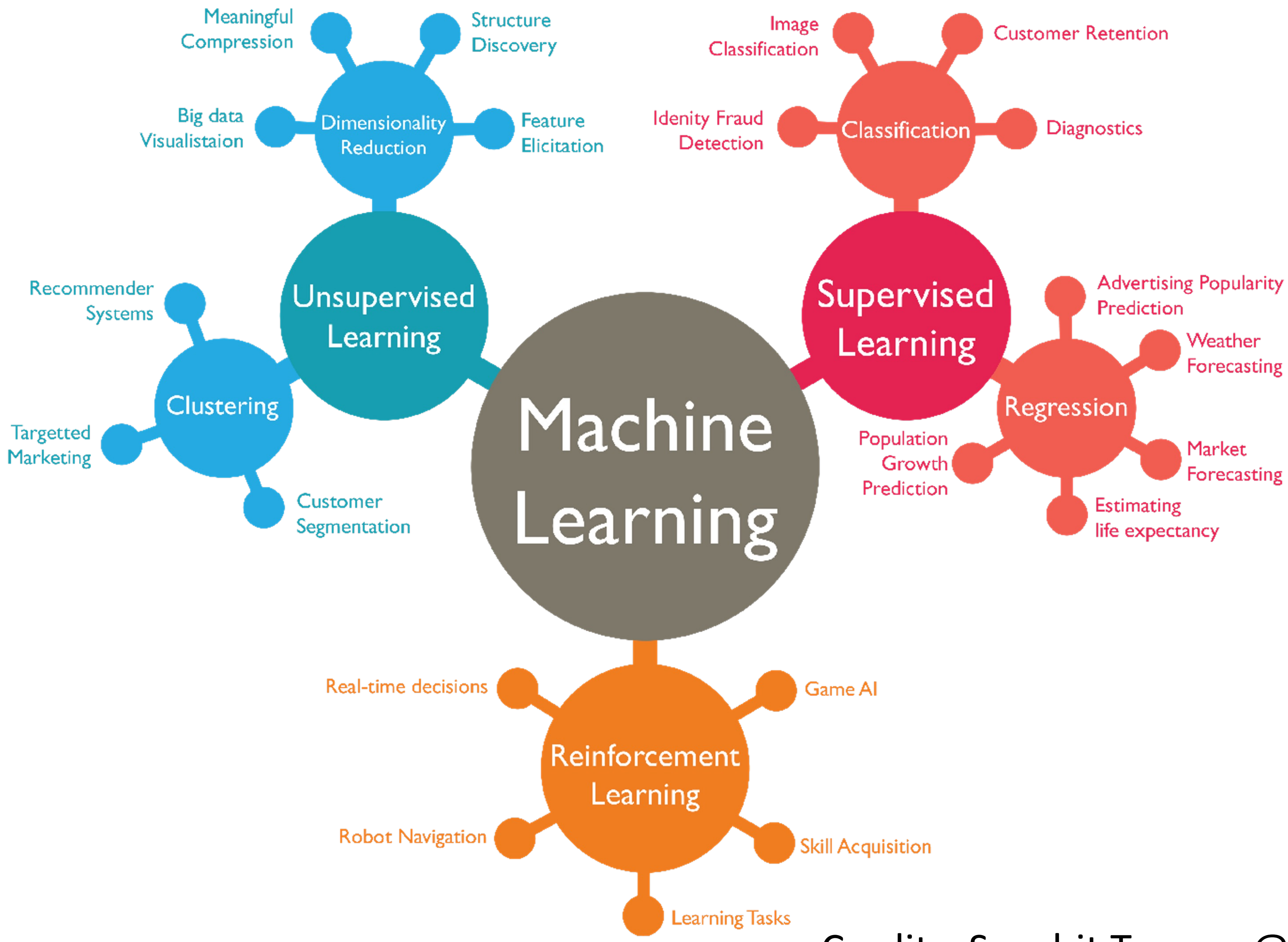
We will use PollEV.com (Polls Everywhere) for questions/comments throughout the class, if you are not comfortable sharing it out aloud!

Today’s Plan

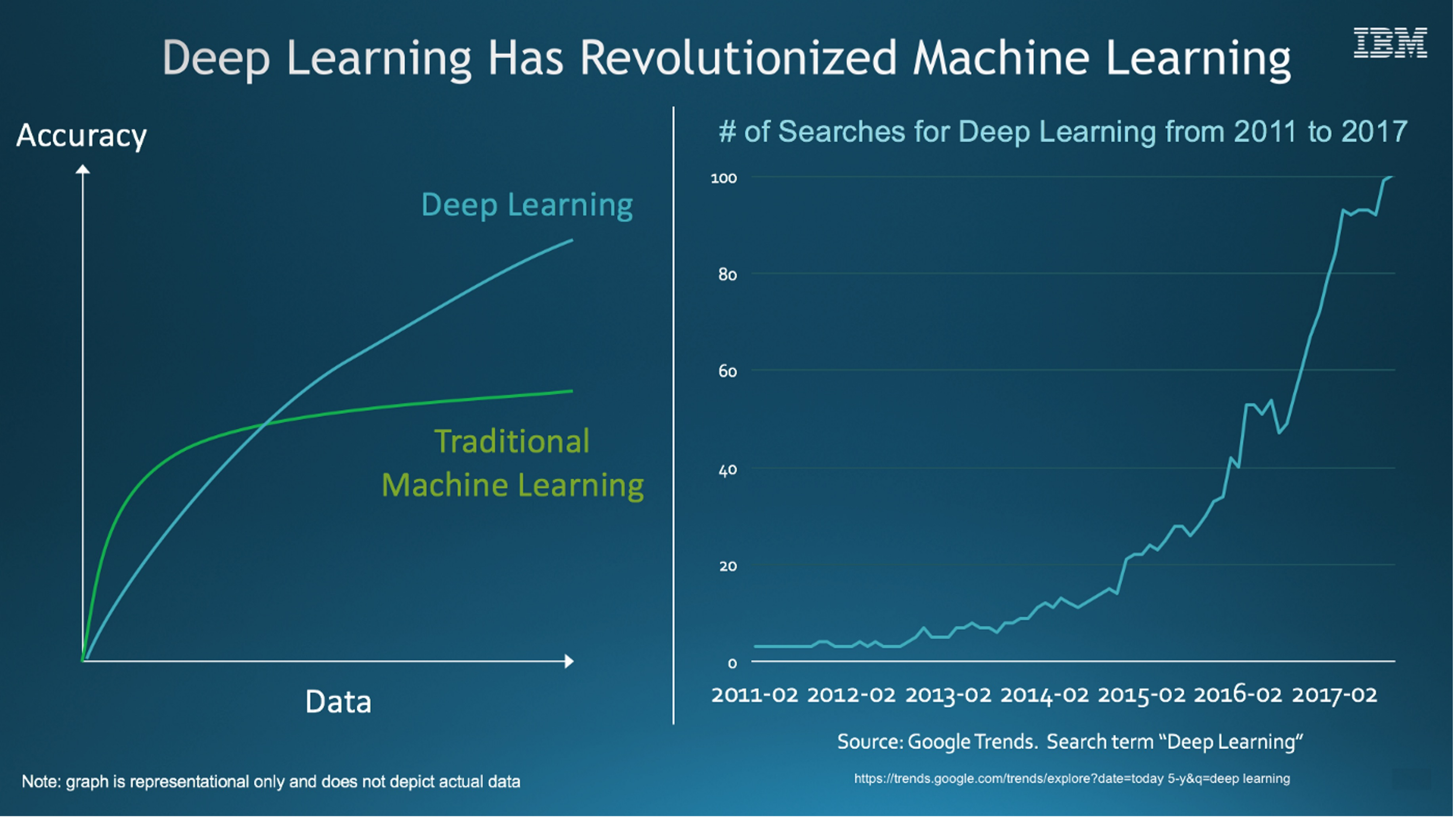
- Introduction
- Motivation for the course
- Course Overview
- Course expectation (grading, policies, etc.)
- Introduction to Deep Neural Networks



Credits: Sanchit Tanwar @ Medium



Credits: Sanchit Tanwar @ Medium



Credits: Sanchit Tanwar @ Medium

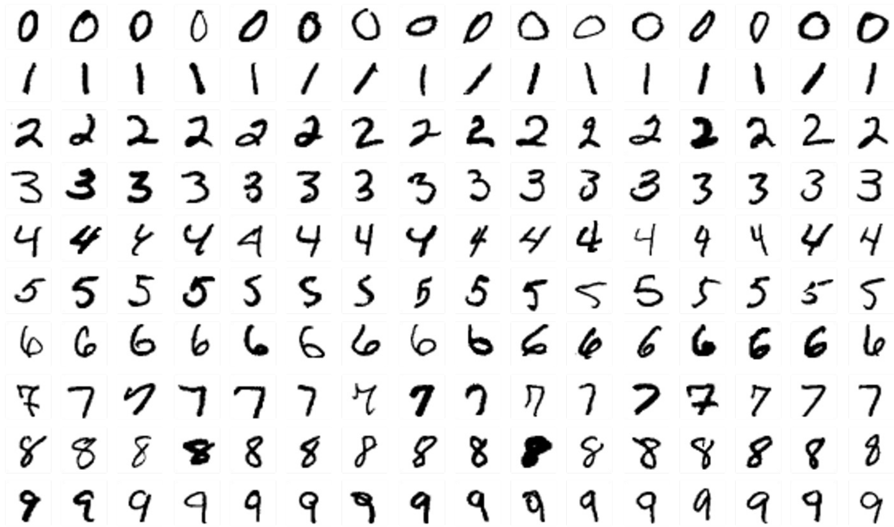
AI APPLICATIONS

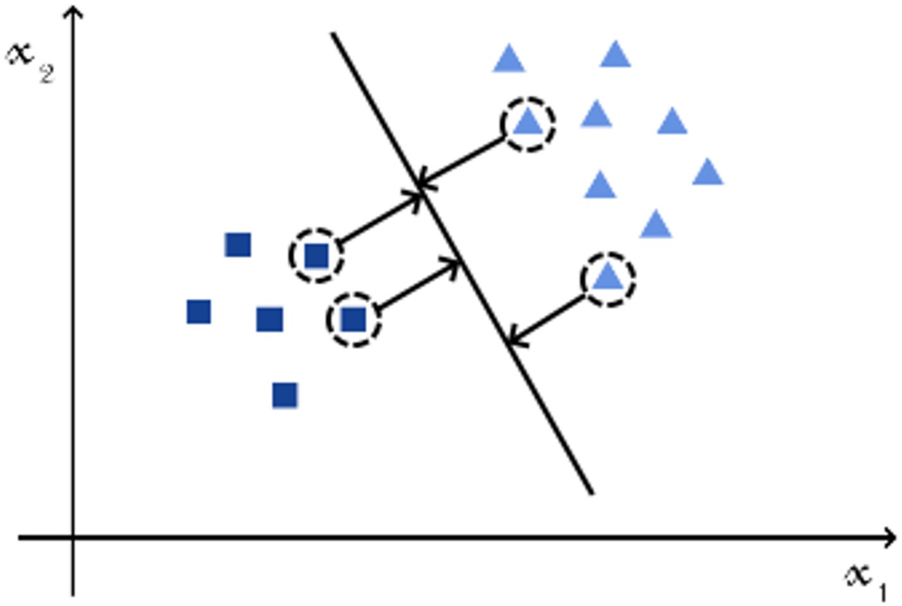
Credits: Sanchit Tanwar @ Medium

What is Classification?

- Calculate a function that can cluster features into groups of the same classification.

- Learn the function through linear regression by minimizing a loss function

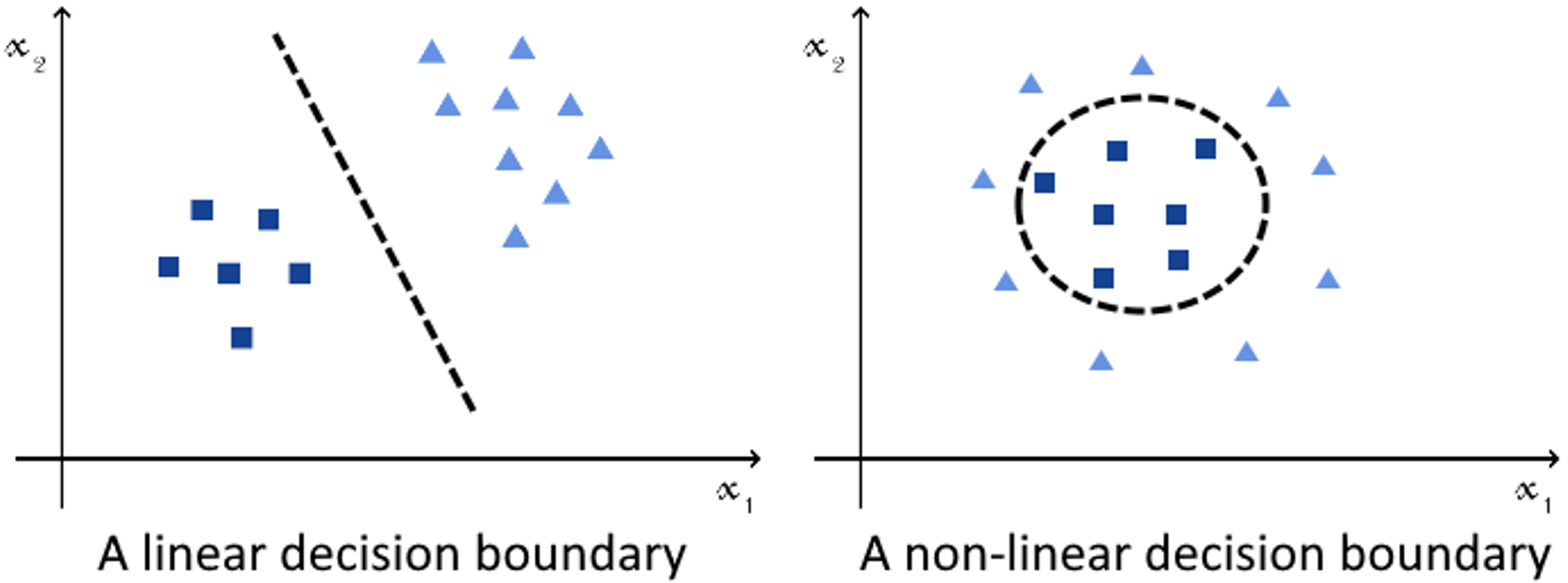




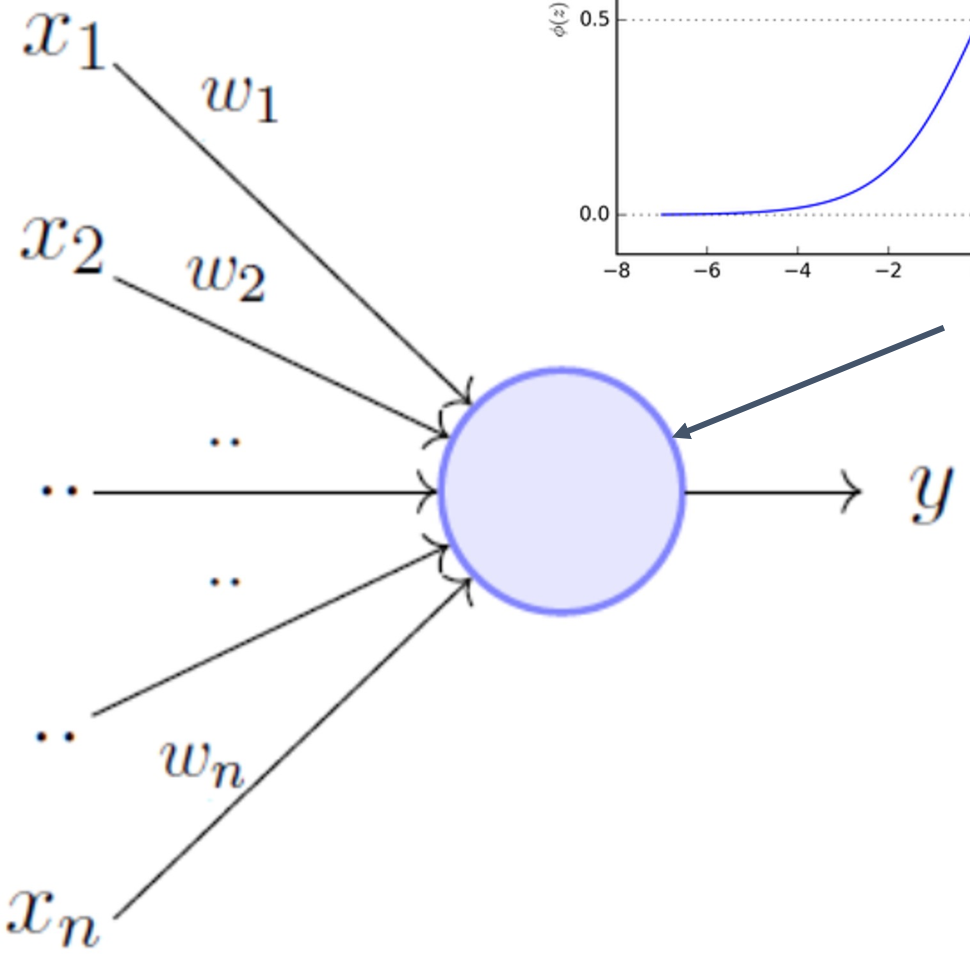
Identifying decision boundaries

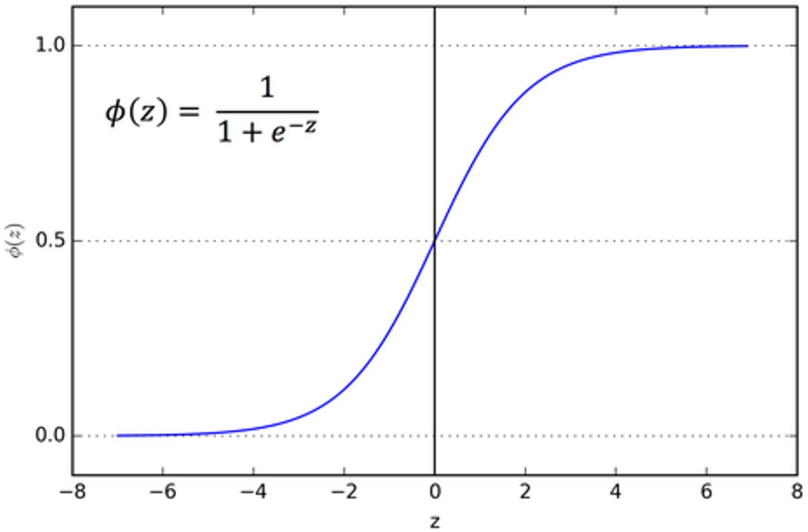
Credits: John Akers, Joe Redmond

Why Using Traditional Linear Models is a Challenge



- Linear models are only as effective as the quality of the features they are provided
- People must design good features for the model
- What if the computer could come up with its own features?





Add a sigmoid function (non linear)

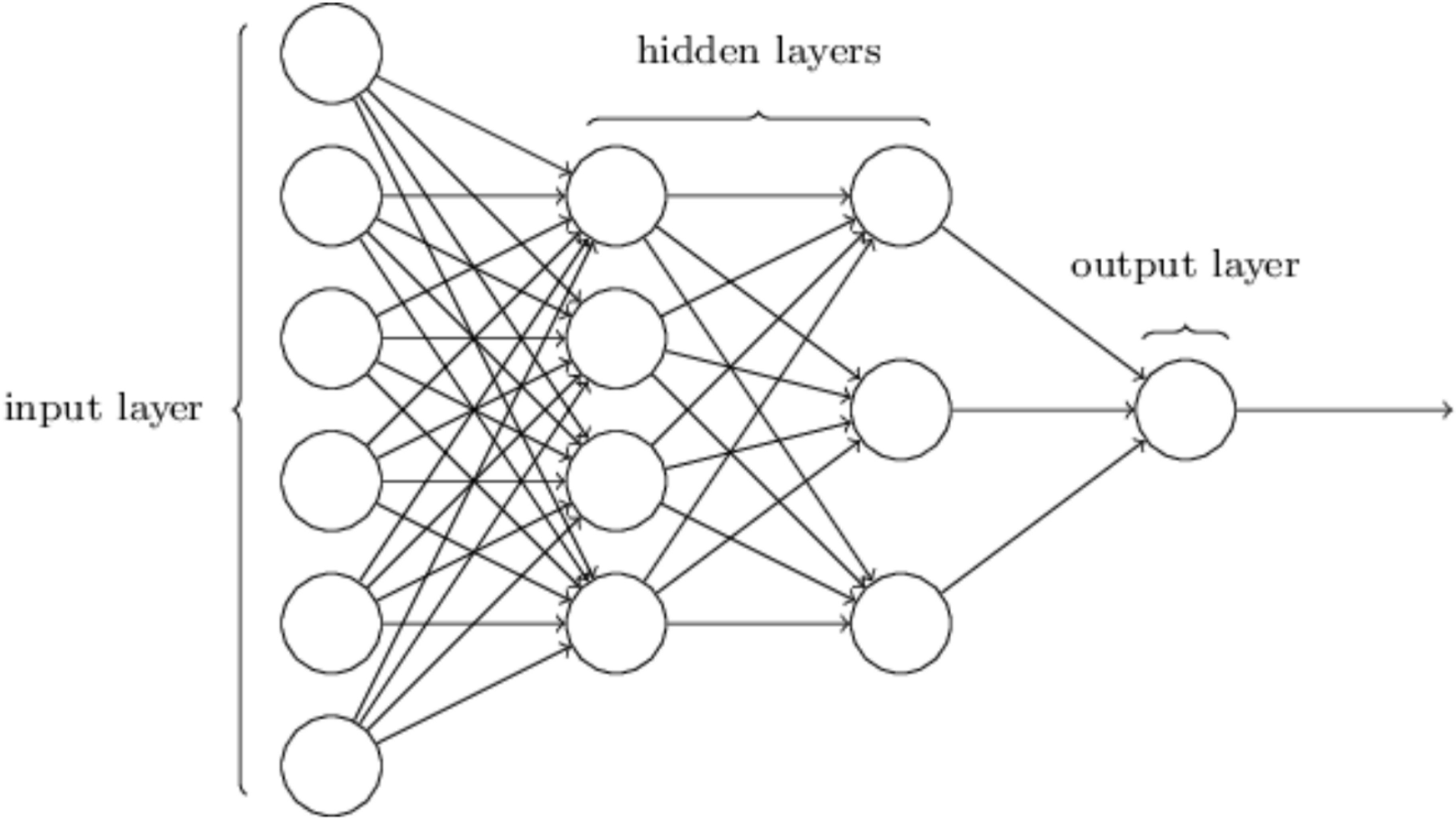
$$y = 1 \quad if \sum_{i=1}^{n} w_i * x_i \geq \theta$$

$$= 0 \quad if \sum_{i=1}^{n} w_i * x_i < \theta$$

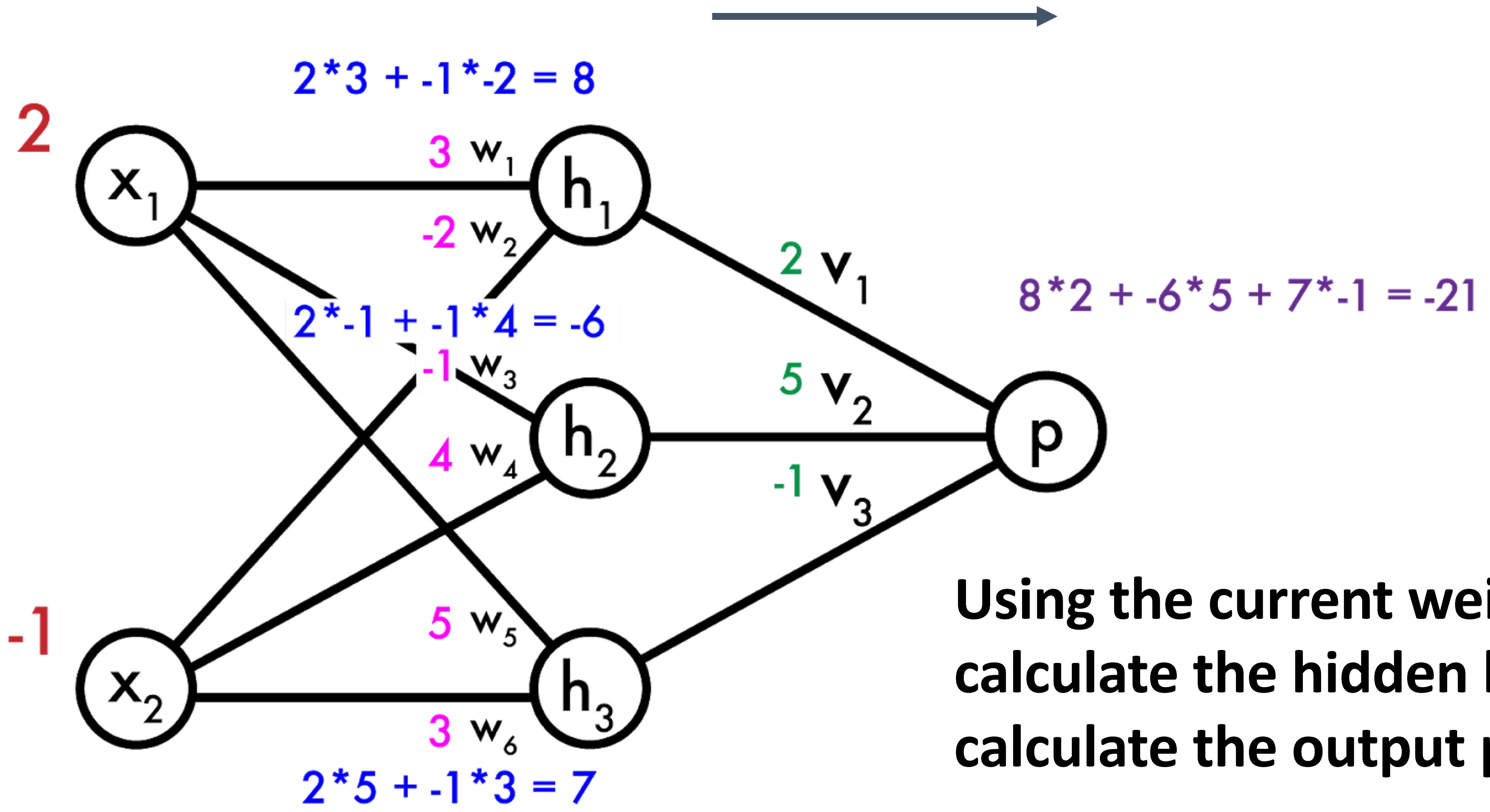
Rewriting the above,

$$y = 1 \quad if \sum_{i=1}^{n} w_i * x_i - \theta \geq 0$$

$$= 0 \quad if \sum_{i=1}^{n} w_i * x_i - \theta < 0$$

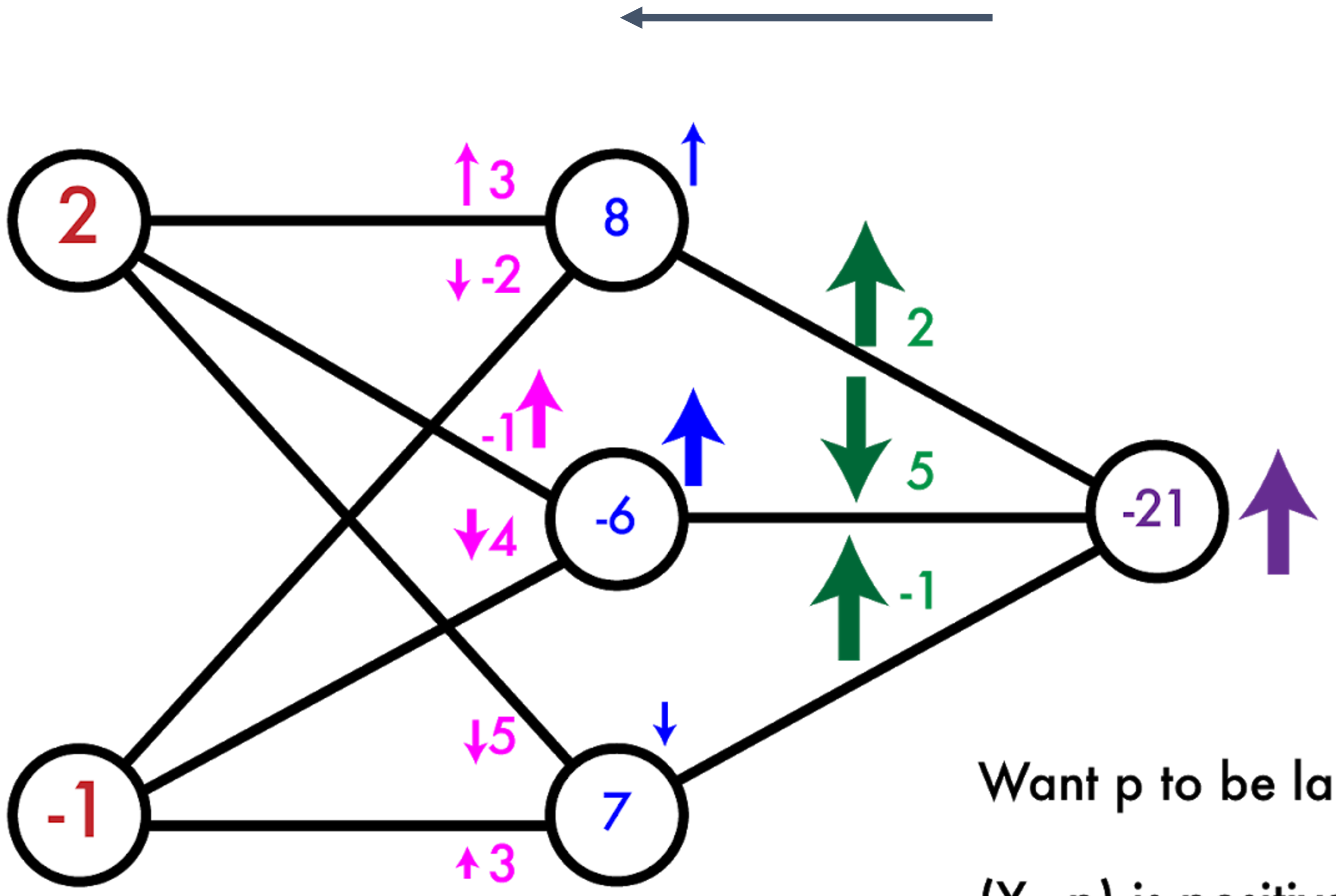

hidden layers
output layer
input layer

The Learning Cycle: Forward Propagation



Using the current weights of the model calculate the hidden layer neurons, then calculate the output p.

The Learning Cycle: Backward Propagation



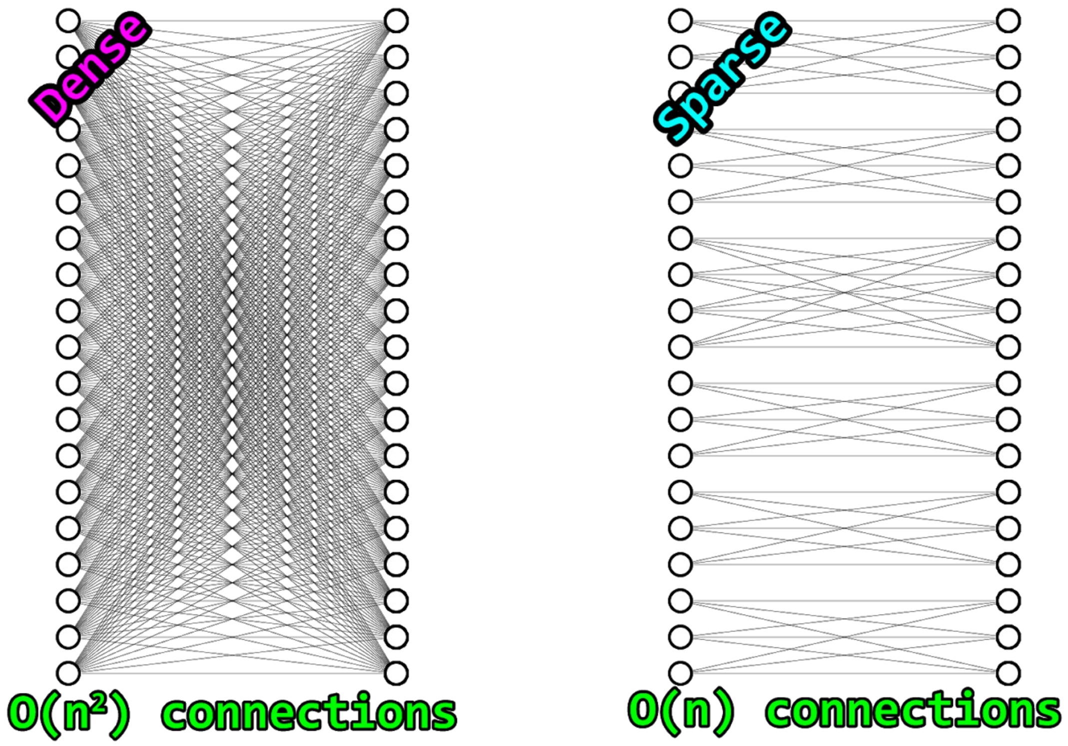
Want p to be larger...

(Y - p) is positive

How do we change our weights?

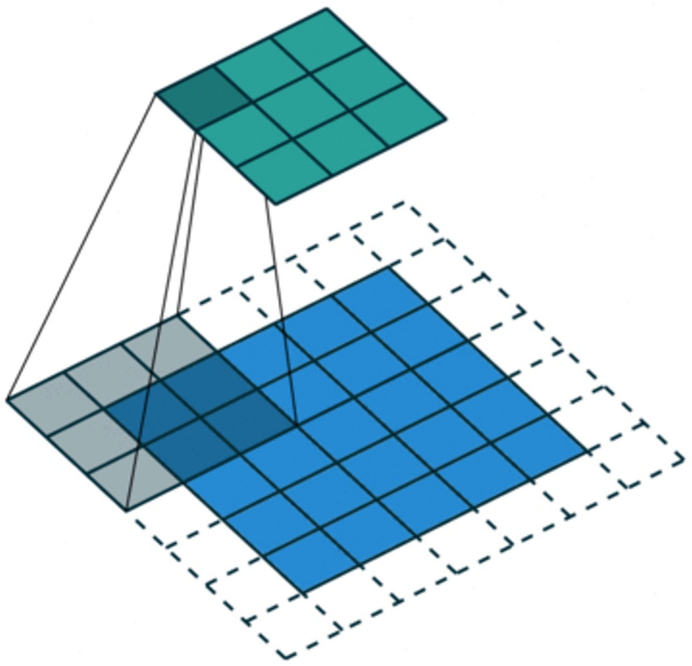
Size of Linear Layers

- In a linear layer every neuron is fully connected to every other neuron
- In the context of images where every channel of a pixel is a neuron the hidden layer size would be massive (billions of connections)



Convolutions can Reduce the Connections!

- Reduces the number of connections between layers
- Creates an assumption that nearby pixels are related and distant pixels are not
- Generates an output where each channel is a filter



Animation credit to:https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

2D Convolution

a 2D grid of values that we want to convolve (e.g. an image)

$f =$

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

convolution filter

$g =$

1	2	1
0	0	0
1	2	1

$h = g * f =$

0	0	0	0	0	0
0	0	0	0	0	0
0	0	1	2	3	4
0	0	5	6	7	8
0	0	9	10	11	12
0	0	13	14	15	16

$=$

Slide Credits: Gedas Bertasius

2D Convolution

a 2D grid of values that we want to convolve (e.g. an image)

$$f = \begin{array}{|c|c|c|c|} \hline 1 & 2 & 3 & 4 \\ \hline 5 & 6 & 7 & 8 \\ \hline 9 & 10 & 11 & 12 \\ \hline 13 & 14 & 15 & 16 \\ \hline \end{array}$$

convolution filter

$$g = \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

$$h = g * f = \begin{array}{|c|c|c|c|c|c|} \hline {}^{1}0 & {}^{2}0 & {}^{1}0 & 0 & 0 & 0 \\ \hline {}^{0}0 & {}^{0}0 & {}^{0}0 & 0 & 0 & 0 \\ \hline {}^{1}0 & {}^{2}0 & {}^{1}1 & 2 & 3 & 4 \\ \hline 0 & 0 & 5 & 6 & 7 & 8 \\ \hline 0 & 0 & 9 & 10 & 11 & 12 \\ \hline 0 & 0 & 13 & 14 & 15 & 16 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline 1 & & & \\ \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array}$$

Slide Credits: Gedas Bertasius

2D Convolution

a 2D grid of values that we want to convolve (e.g. an image)

$$f = \begin{array}{|c|c|c|c|} \hline 1 & 2 & 3 & 4 \\ \hline 5 & 6 & 7 & 8 \\ \hline 9 & 10 & 11 & 12 \\ \hline 13 & 14 & 15 & 16 \\ \hline \end{array}$$

convolution filter

$$g = \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

$$h = g * f = \begin{array}{|c|c|c|c|c|c|} \hline 0 & {}^{1}0 & {}^{2}0 & {}^{1}0 & 0 & 0 \\ \hline 0 & {}^{0}0 & {}^{0}0 & {}^{0}0 & 0 & 0 \\ \hline 0 & {}^{1}0 & {}^{2}1 & {}^{1}2 & 3 & 4 \\ \hline 0 & 0 & 5 & 6 & 7 & 8 \\ \hline 0 & 0 & 9 & 10 & 11 & 12 \\ \hline 0 & 0 & 13 & 14 & 15 & 16 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline 1 & 4 & & \\ \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array}$$

Slide Credits: Gedas Bertasius

2D Convolution

a 2D grid of values that we want to convolve (e.g. an image)

$$f = \begin{array}{|c|c|c|c|} \hline 1 & 2 & 3 & 4 \\ \hline 5 & 6 & 7 & 8 \\ \hline 9 & 10 & 11 & 12 \\ \hline 13 & 14 & 15 & 16 \\ \hline \end{array}$$

convolution filter

$$g = \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

$$h = g * f = \begin{array}{|c|c|c|c|c|c|} \hline 0 & 0 & {}^{1}0 & {}^{2}0 & {}^{1}0 & 0 \\ \hline 0 & 0 & {}^{0}0 & {}^{0}0 & {}^{0}0 & 0 \\ \hline 0 & 0 & {}^{1}1 & {}^{2}2 & {}^{1}3 & 4 \\ \hline 0 & 0 & 5 & 6 & 7 & 8 \\ \hline 0 & 0 & 9 & 10 & 11 & 12 \\ \hline 0 & 0 & 13 & 14 & 15 & 16 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline 1 & 4 & 8 & \\ \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array}$$

Slide Credits: Gedas Bertasius

2D Convolution

a 2D grid of values that we want to convolve (e.g. an image)

$$f = \begin{array}{|c|c|c|c|} \hline 1 & 2 & 3 & 4 \\ \hline 5 & 6 & 7 & 8 \\ \hline 9 & 10 & 11 & 12 \\ \hline 13 & 14 & 15 & 16 \\ \hline \end{array}$$

convolution filter

$$g = \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

$$h = g * f = \begin{array}{|c|c|c|c|c|c|} \hline 0 & 0 & 0 & {}^{1}0 & {}^{2}0 & {}^{1}0 \\ \hline 0 & 0 & 0 & {}^{0}0 & {}^{0}0 & {}^{0}0 \\ \hline 0 & 0 & 1 & {}^{1}2 & {}^{2}3 & {}^{1}4 \\ \hline 0 & 0 & 5 & 6 & 7 & 8 \\ \hline 0 & 0 & 9 & 10 & 11 & 12 \\ \hline 0 & 0 & 13 & 14 & 15 & 16 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline 1 & 4 & 8 & 12 \\ \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array}$$

Slide Credits: Gedas Bertasius

2D Convolution

a 2D grid of values that we want to convolve (e.g. an image)

$f =$

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

convolution filter

$g =$

1	2	1
0	0	0
1	2	1

$h = g * f =$

0	0	0	0	0	0
${}^{1}0$	${}^{2}0$	${}^{1}0$	0	0	0
${}^{0}0$	${}^{0}0$	${}^{0}1$	2	3	4
${}^{1}0$	${}^{2}0$	${}^{1}5$	6	7	8
0	0	9	10	11	12
0	0	13	14	15	16

$=$

1	4	8	12
5			

Slide Credits: Gedas Bertasius

2D Convolution

a 2D grid of values that we want to convolve (e.g. an image)

$$f = \begin{array}{|c|c|c|c|}\hline 1 & 2 & 3 & 4 \\ \hline 5 & 6 & 7 & 8 \\ \hline 9 & 10 & 11 & 12 \\ \hline 13 & 14 & 15 & 16 \\ \hline \end{array}$$

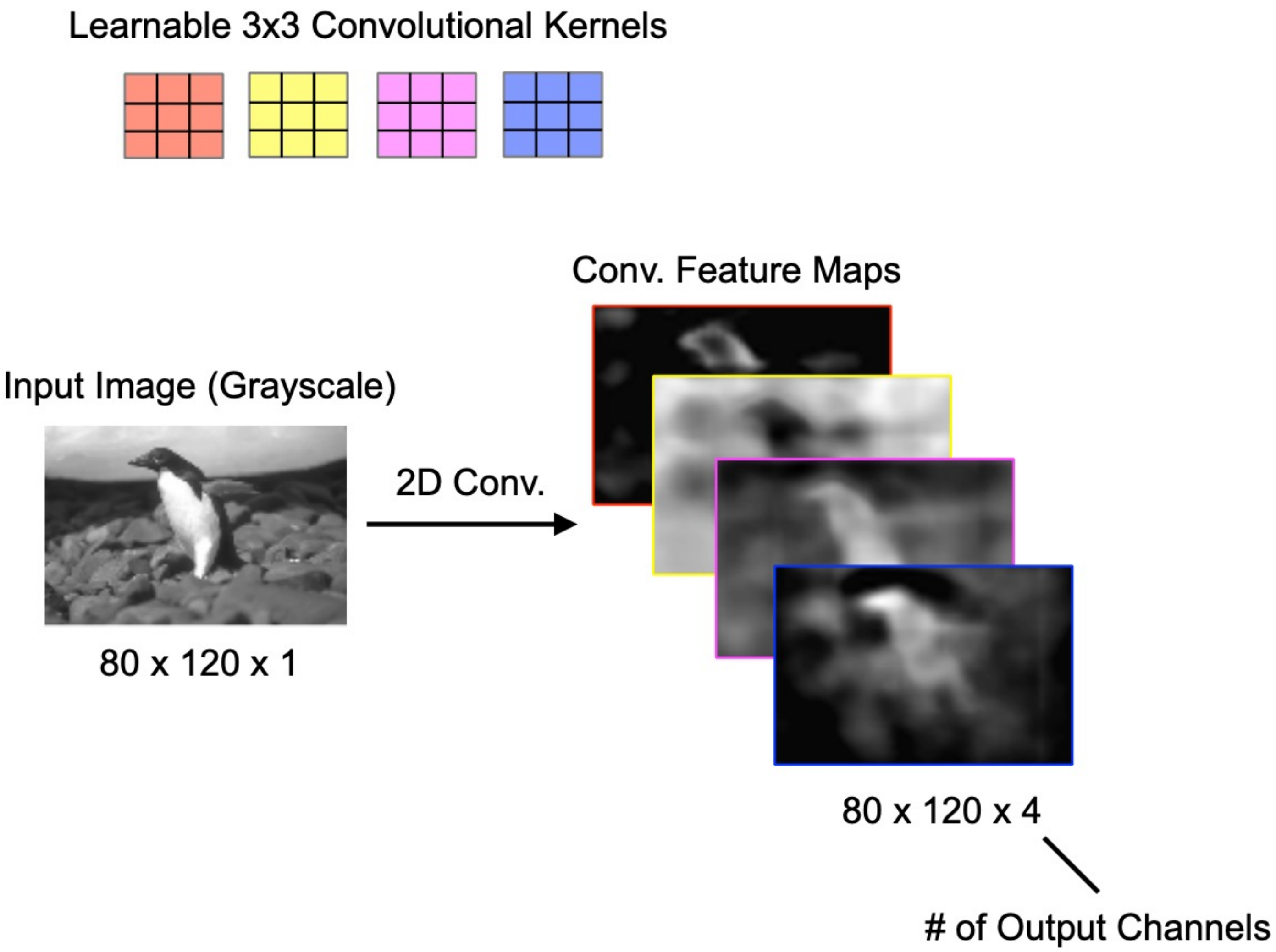
convolution filter

$$g = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

$$h = g * f = \begin{array}{|c|c|c|c|c|c|}\hline 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 0 & 5 & {}^{1}6 & {}^{2}7 & {}^{1}8 \\ \hline 0 & 0 & 9 & {}^{0}10 & {}^{0}11 & {}^{0}12 \\ \hline 0 & 0 & 13 & {}^{1}14 & {}^{2}15 & {}^{1}16 \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline 1 & 4 & 8 & 12 \\ \hline 5 & 16 & 24 & 28 \\ \hline 10 & 32 & 48 & 56 \\ \hline 18 & 56 & 80 & 88 \\ \hline \end{array}$$

Slide Credits: Gedas Bertasius

Convolutional Networks

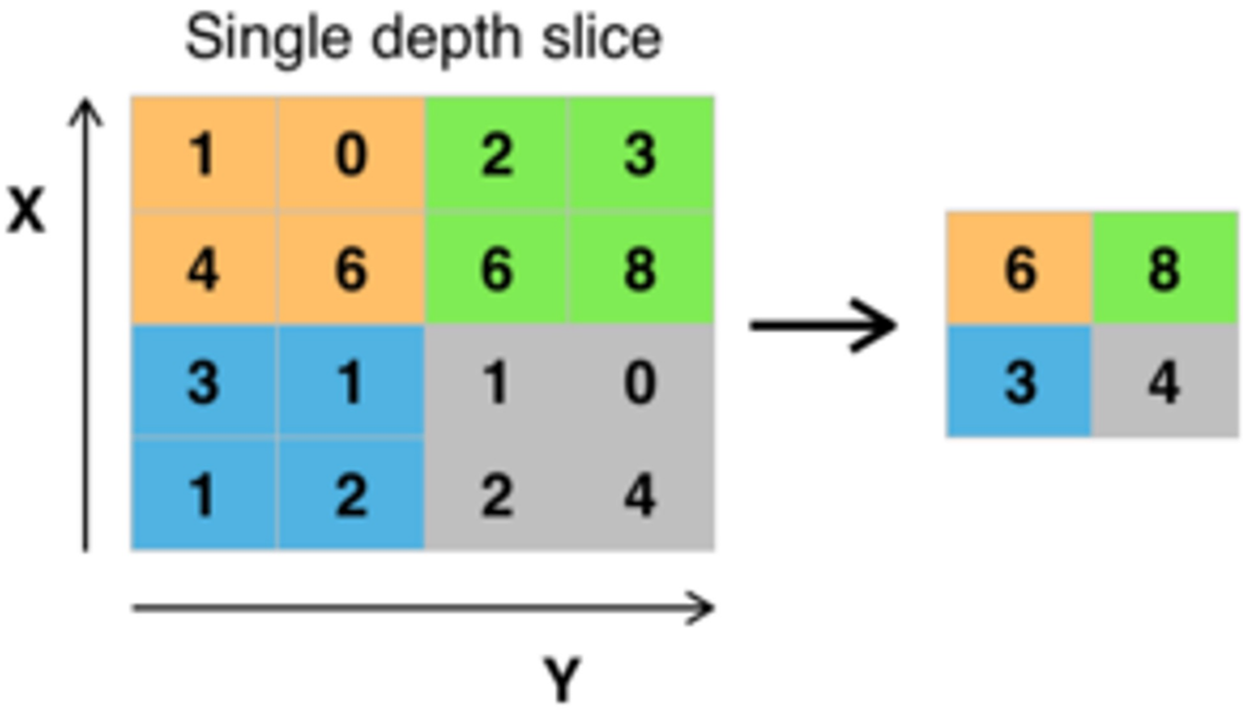


Slide Credits: Gedas Bertasius

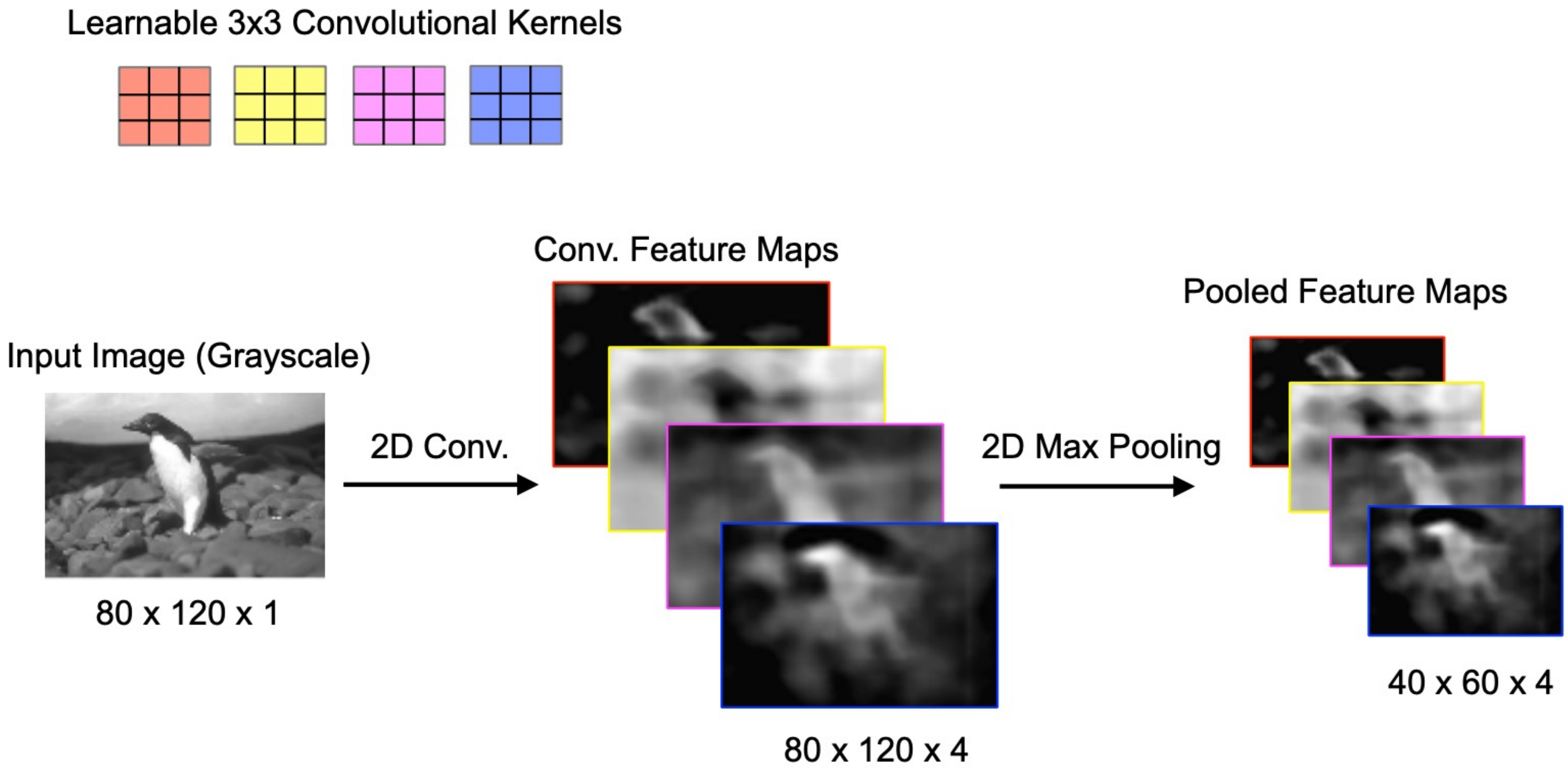
How Else to Shrink the Model Size?

Pooling Layer:

- Max Pooling
- Other pooling options like average pooling are also used



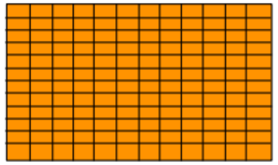
Convolutional Networks



Slide Credits: Gedas Bertasius

Convolutional Networks

Learnable FC Layer Weight Matrix

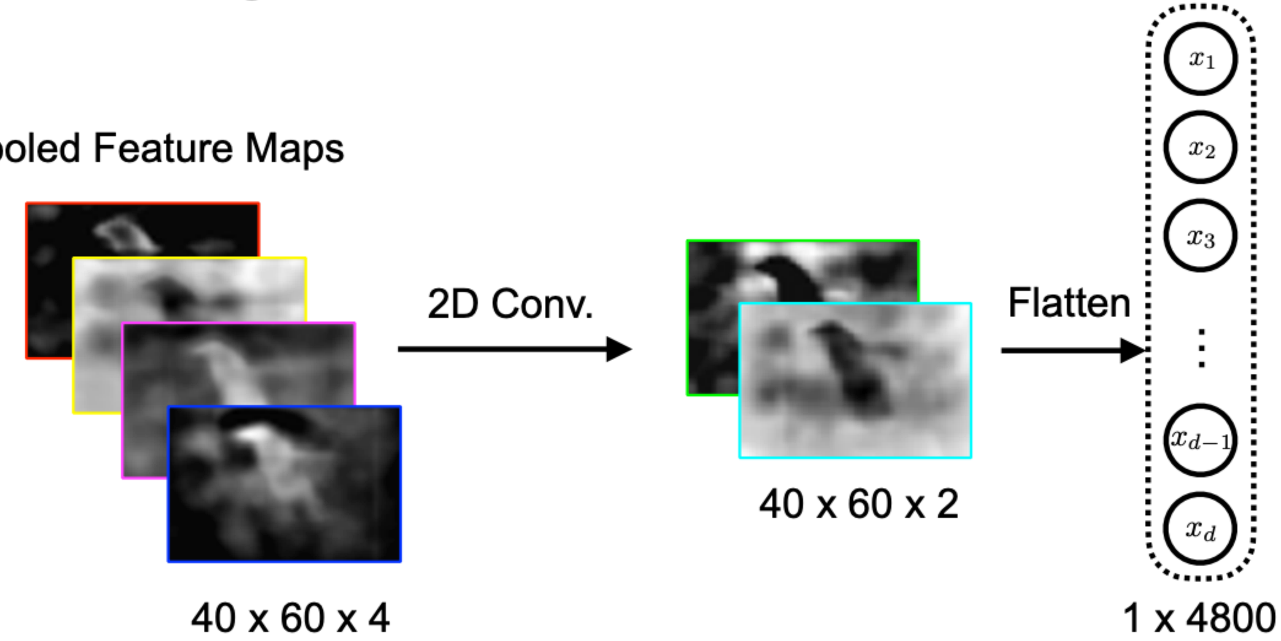


$$W \in \mathbb{R}^{d \times C}$$

d - feature dimensionality (4800 in this example)

C - number of classes

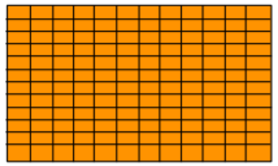
Pooled Feature Maps



Slide Credits: Gedas Bertasius

Convolutional Networks

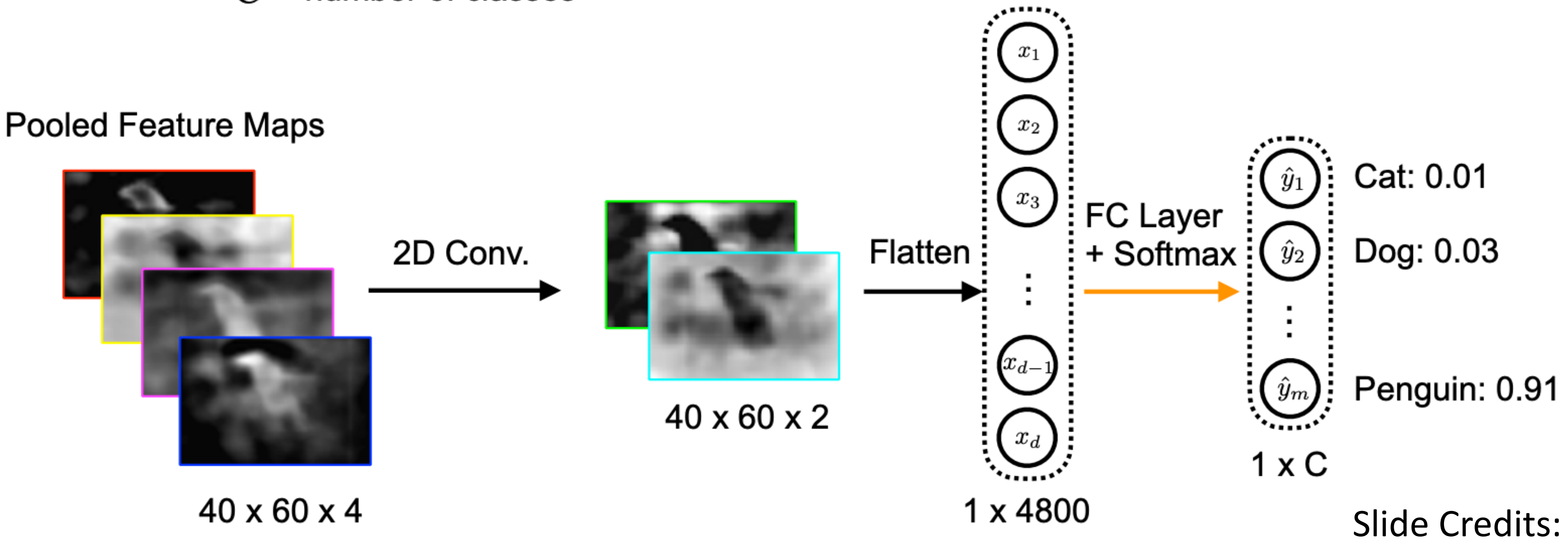
Learnable FC Layer Weight Matrix



$$W \in \mathbb{R}^{d \times C}$$

d - feature dimensionality (4800 in this example)

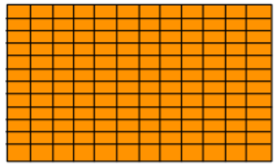
C - number of classes



Slide Credits: Gedas Bertasius

Convolutional Networks

Learnable FC Layer Weight Matrix



$$W \in \mathbb{R}^{d \times C}$$

d - feature dimensionality (4800 in this example)

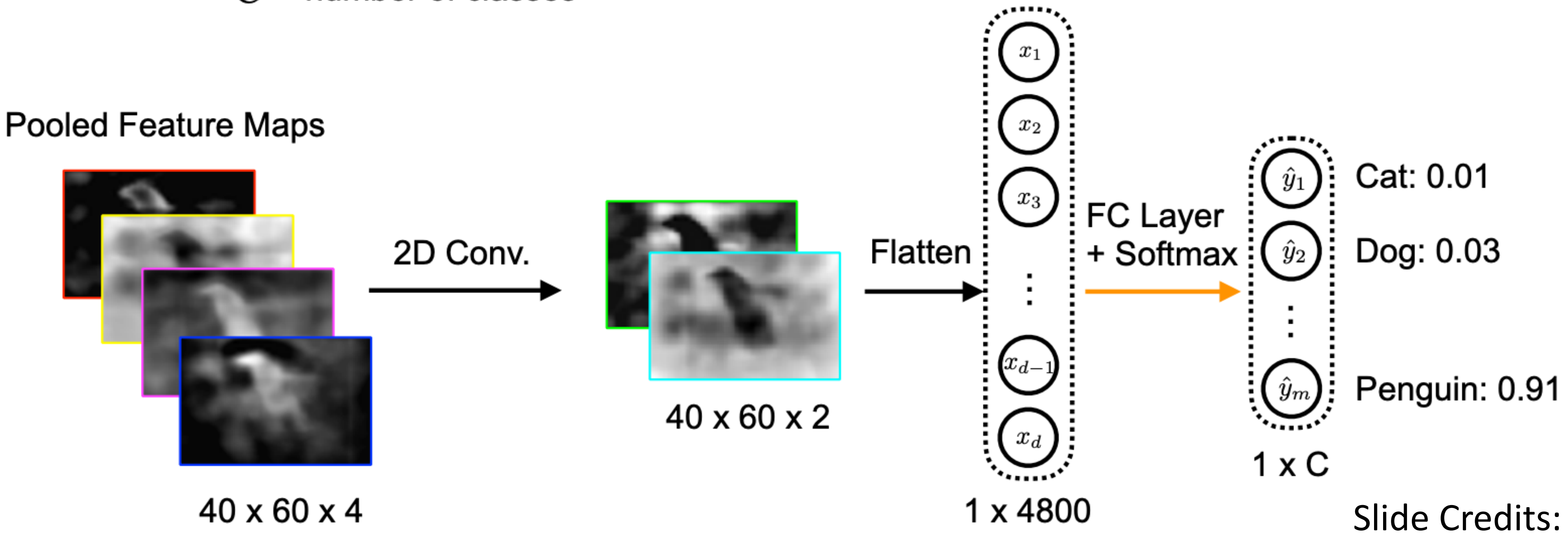
C - number of classes

Fully Connected Layers

$$z = xW$$
$$\hat{y} = softmax(z)$$

$x \in \mathbb{R}^{1 \times d}$ - flattened feature vector

$\hat{y} \in \mathbb{R}^{1 \times C}$ - predicted probabilities

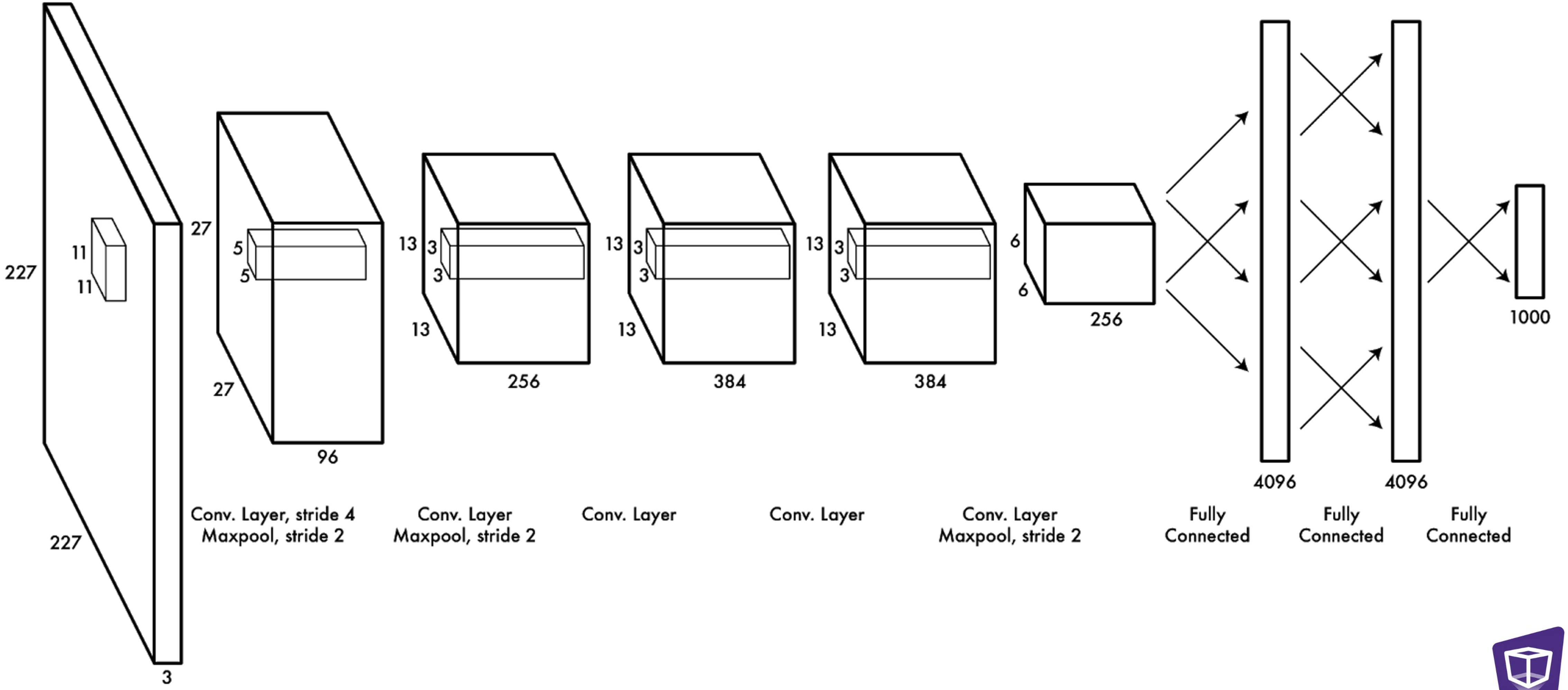


Slide Credits: Gedas Bertasius

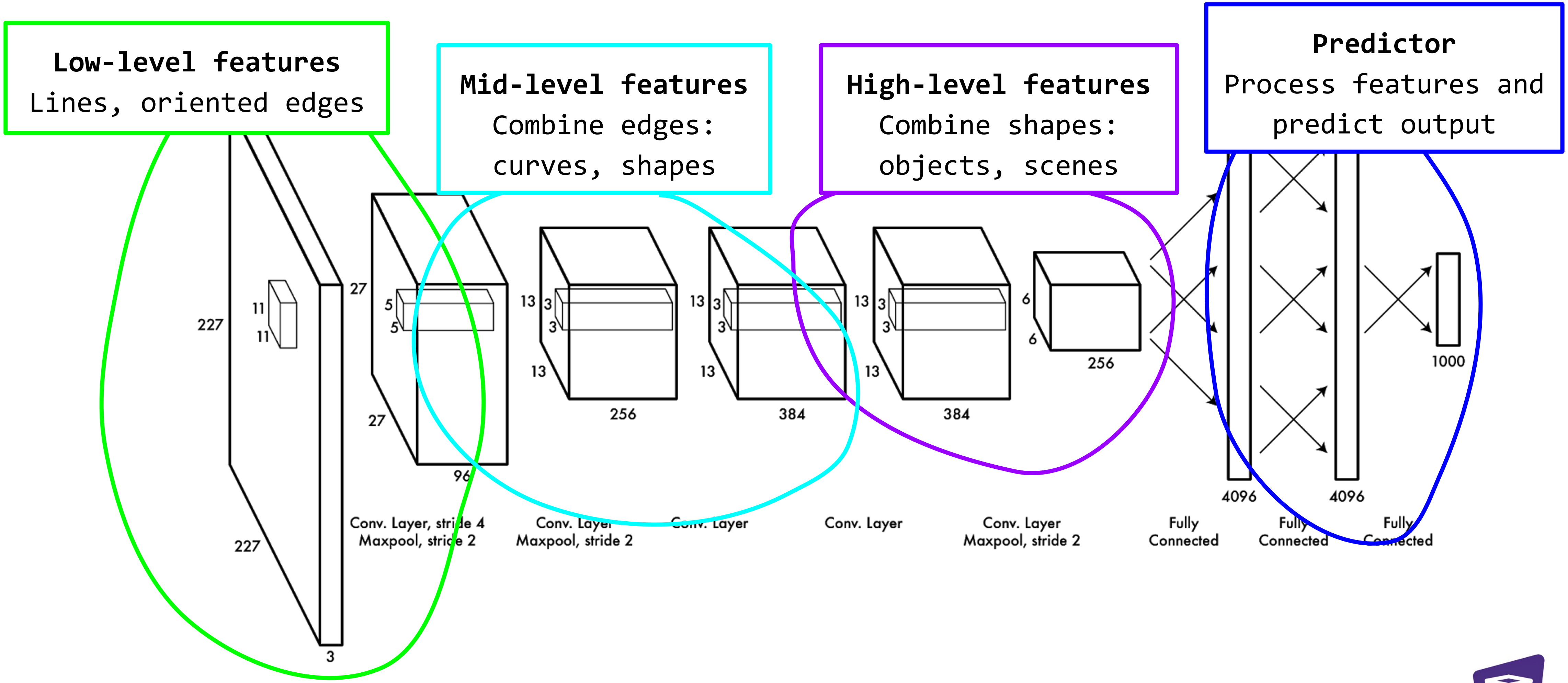
Convolutional Neural Networks Summary

- Convolutional Layers
  - Used to extract features
- Pooling Layers
  - Used to downsample feature maps to increase efficiency
- Connected Layers
  - Used at the end of a model to map image features to predictions
- Activation Function
  - Used to calculate the output of every neuron on a layer
  - Sometimes considered a layer but does not change the shape of the model

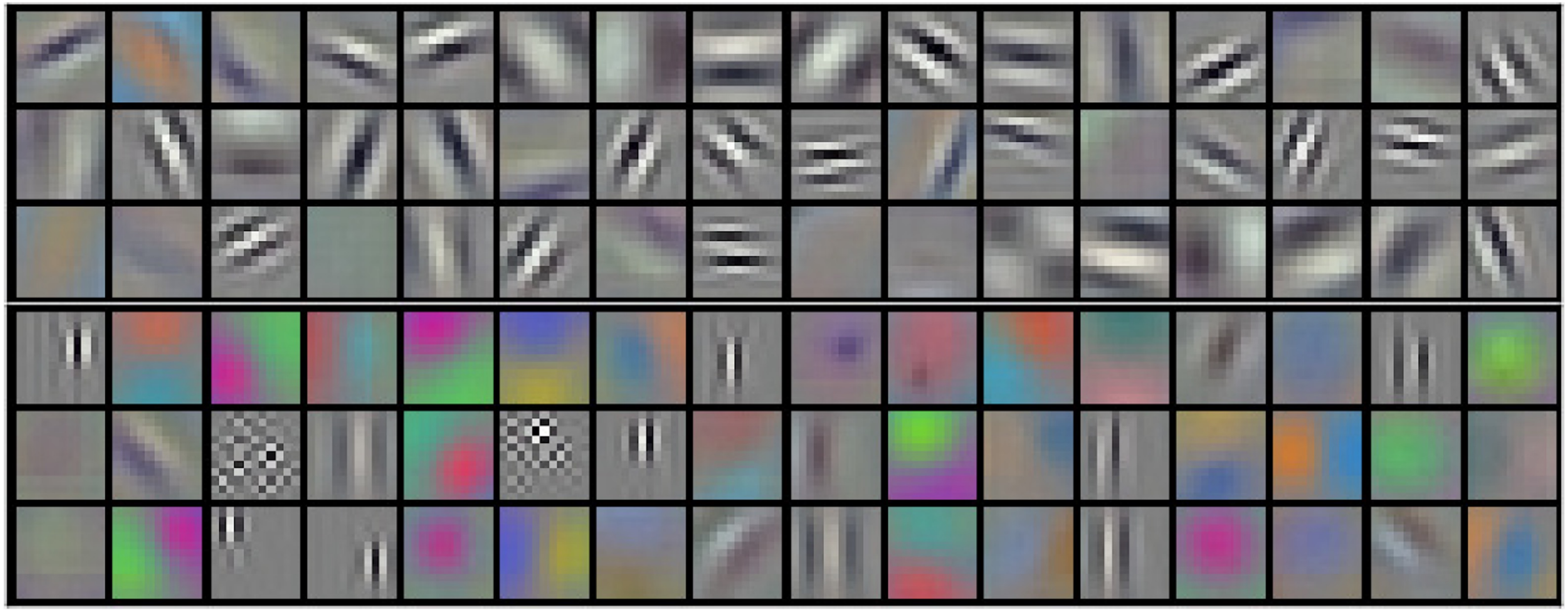
AlexNet: An Early Example



Where Models Learn Features of an Image



What are Models Actually Looking For?



Where to Look for More Information

- Explore existing computer vision and machine learning frameworks
  - https://pytorch.org/
  - https://www.tensorflow.org/
  - https://keras.io/
  - https://opencv.org/
- Watch more in-depth lecture series
  - The Ancient Secrets of Computer Vision - Joseph Redmon
  - Deep Learning Specialization - Andrew Ng
- Checkout other online courses and guides
  - https://ai.google/education/
  - https://www.udacity.com/course/deep-learning-pytorch--ud188

590 Assignment: Train a neural network for classification on CIFAR10 dataset in google colab

[Google Colab Page]