A Deep Understanding of CNN

Joseph Robinson

Advisor: Yun (Raymond) Fu

ECE7399
Spring 2018

So What-Who Cares???

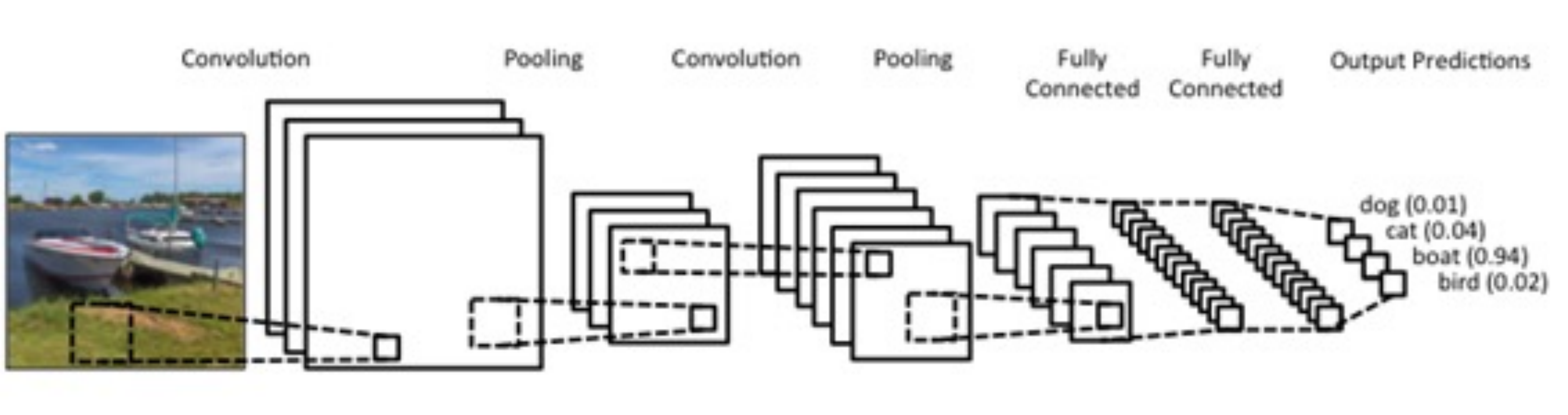
- Pre-Deep learning: saturation on existing datasets
  - Needed larger, more complex datasets to properly represent real-world data, in part, triggered Big Data Era
  - Need systems for Big Data, i.e., sample count & complexity
- Feature extraction, representation, and modeling were hand-crafted by experts, i.e., better if end-to-end learning
- Today's HW was not used to full capacity (i.e., GPGPU)
- Data-driven modern day demands *off-the-shelf* capability for a broader audience (i.e., accessible to those without PhD)
- Advances in 2012 brought deep learning center stage
- Toolboxes for deep learning provide variety of interfaces, level of implementation, and capability

Towards a Deeper Understanding of NN

Problem Definition

- Capture full complexity of big data without overfitting
- Maximize full capacity of modern day HW
- End-to-end training, opposed to professional with expert knowledge for data, hand-crafted features, and modeling

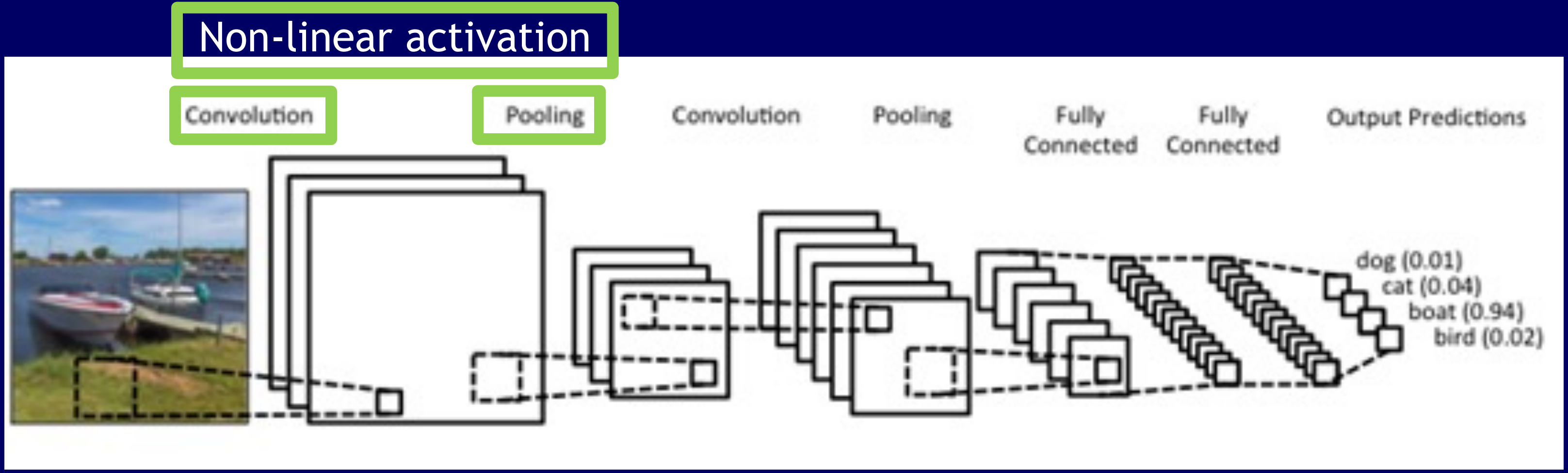
CNNs: Main Components



Neural Networks are constructed using variants of the same underlying components

1. Convolution Layer (Conv-Layer)
2. Non-Linear Activation
3. Pooling (i.e., sub-sampling)
4. Classification-Layer

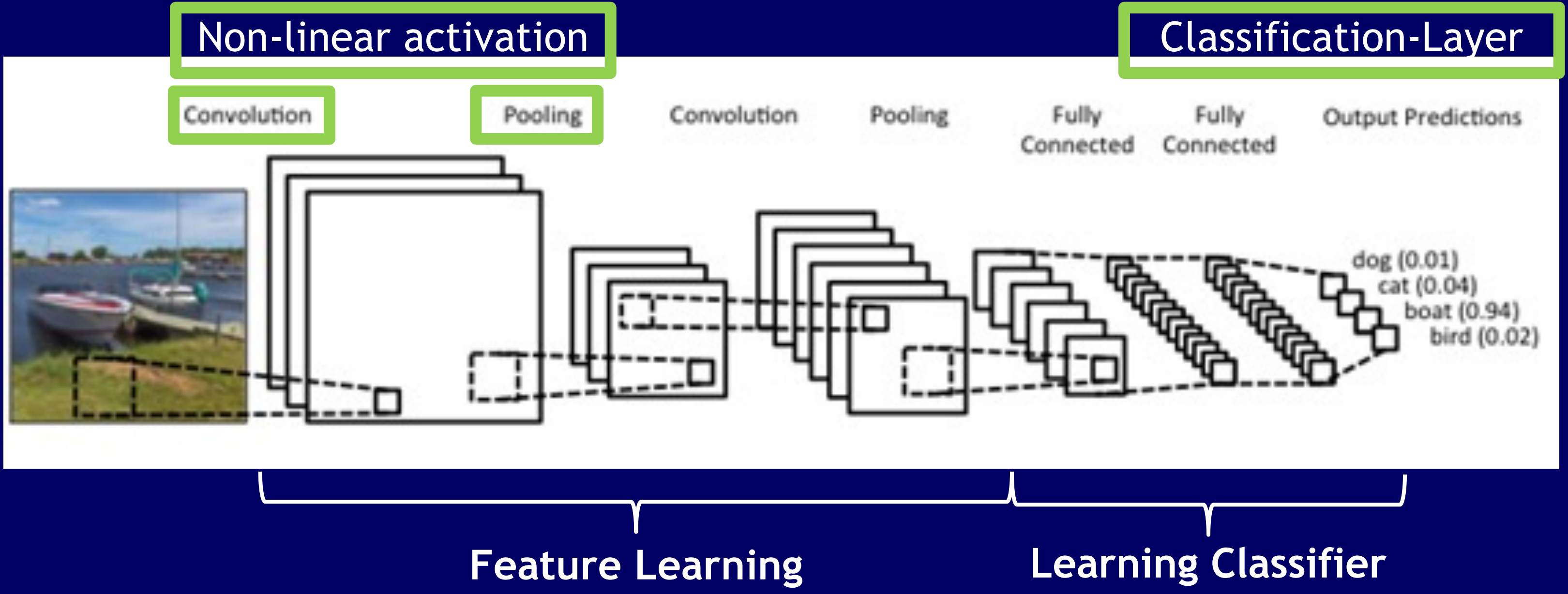
CNNs: Main Components



Neural Networks are constructed using variants of the same underlying components

1. Convolution Layer (Conv-Layer)
2. Non-Linear Activation
3. Pooling (i.e., sub-sampling)
4. Classification-Layer

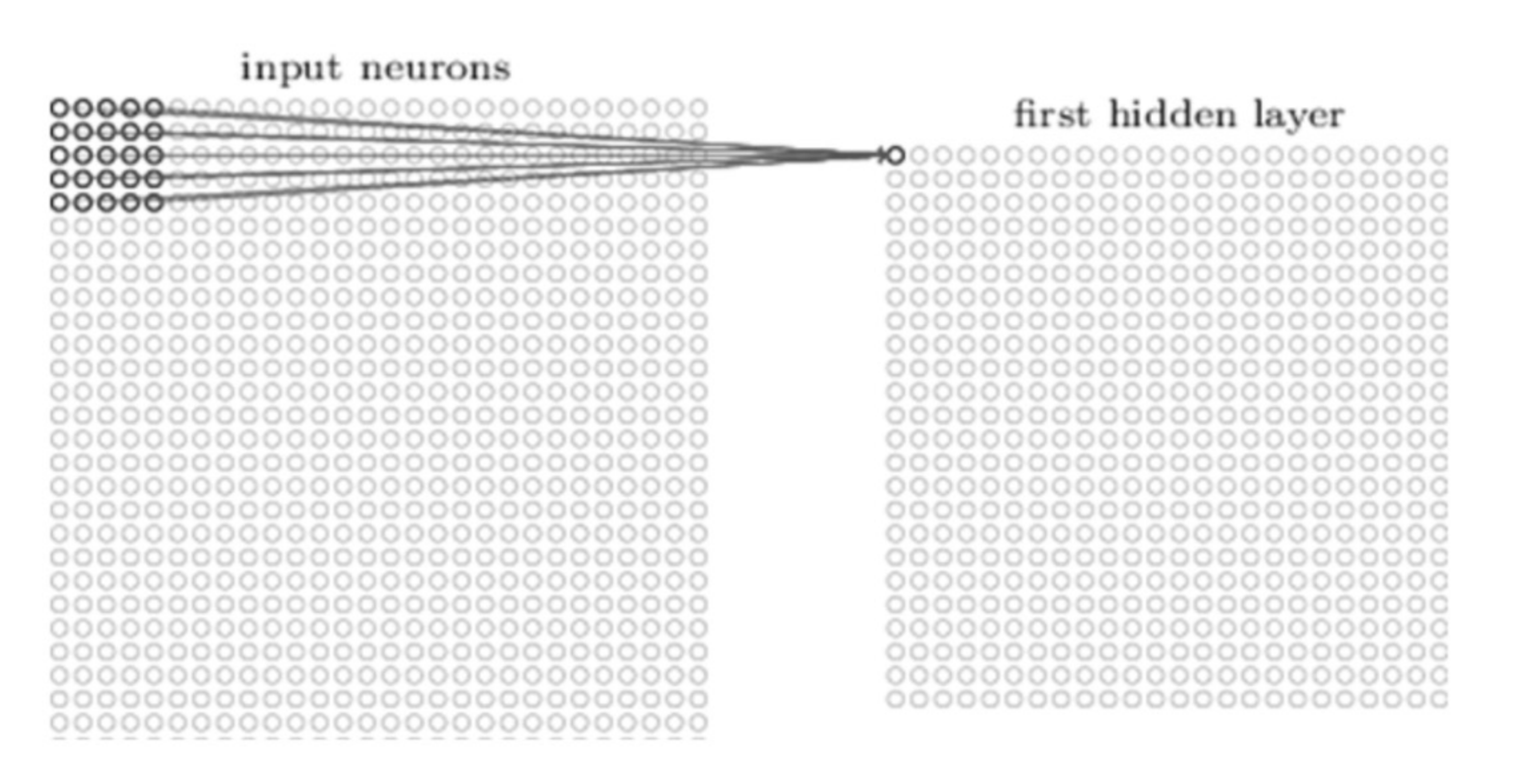
CNNs: Main Components



Neural Networks are constructed using variants of the same underlying components

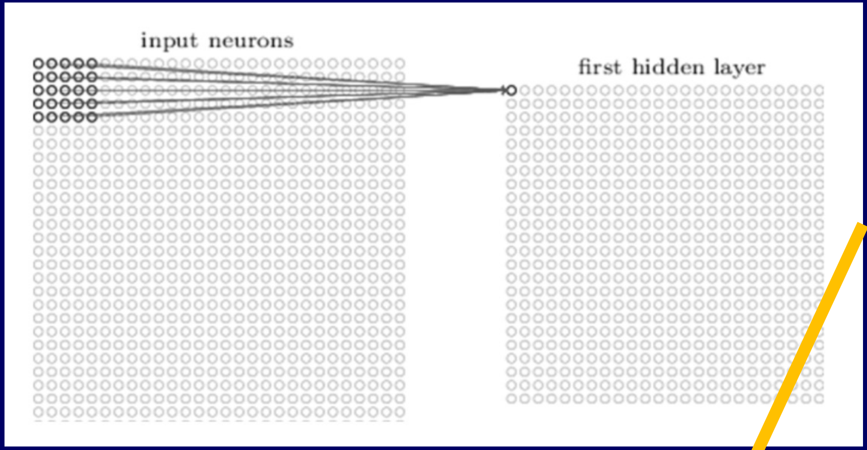
1. Convolution Layer (Conv-Layer)
2. Non-Linear Activation
3. Pooling (i.e., sub-sampling)
4. Classification-Layer

Conv-Net: Conv-Layers



- Element-wise multiplication applied to identical copies of each neuron (aka kernel aka filter)
- Each neurons learn weight for different spatial locations
- Essentially, just a measure of correlation

Conv-Net: Conv-Layers



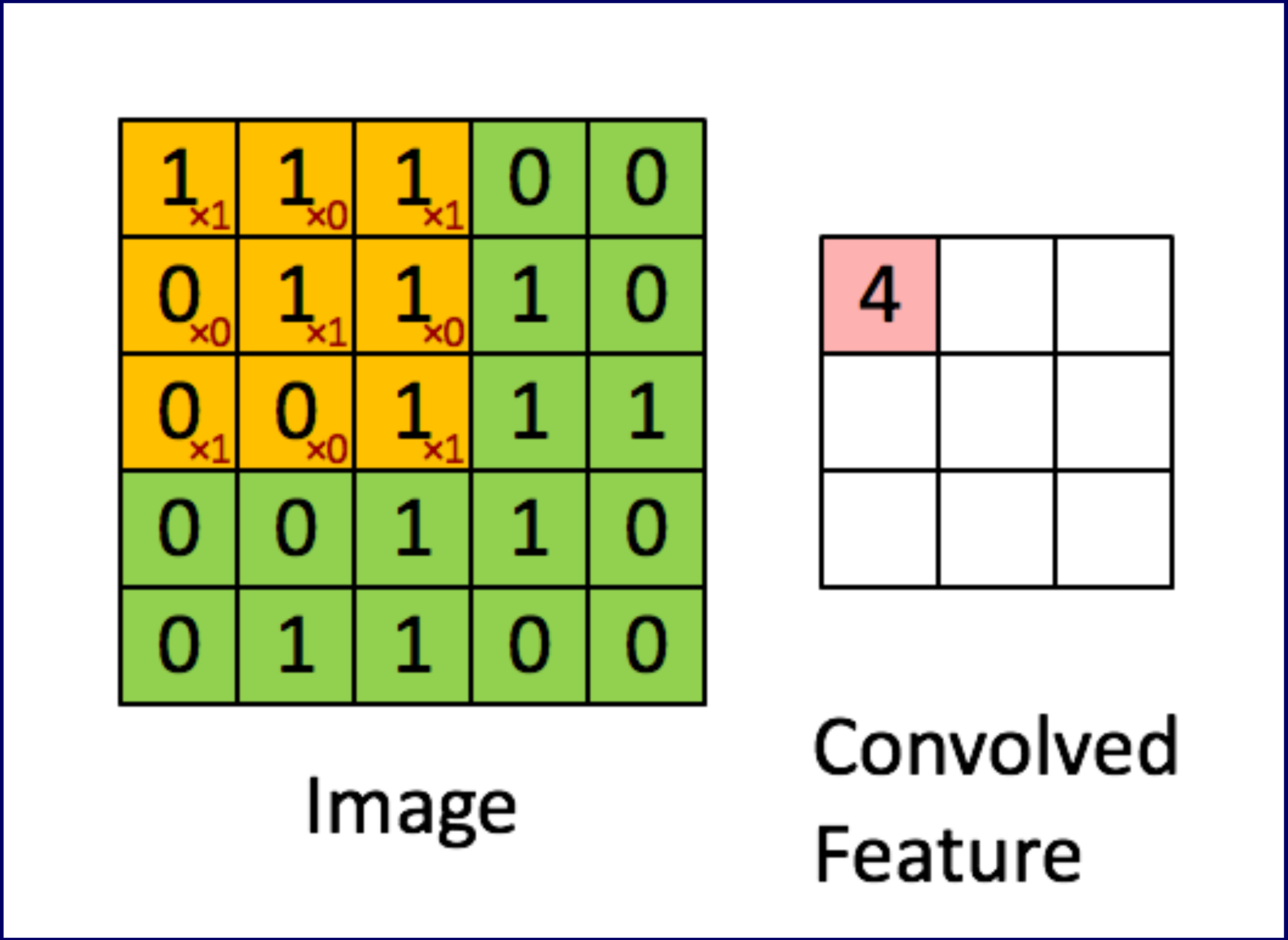
1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Input Signal
(image or activations of previous layer)

1	0	1
0	1	0
1	0	1

Filter (i.e., units in hidden layer)

Convolution Layer (Continued)



$1_{\times 1}$	$1_{\times 0}$	$1_{\times 1}$	0	0
$0_{\times 0}$	$1_{\times 1}$	$1_{\times 0}$	1	0
$0_{\times 1}$	$0_{\times 0}$	$1_{\times 1}$	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved Feature

Convolution operation– Output matrix is called Convolved Feature or Feature Map

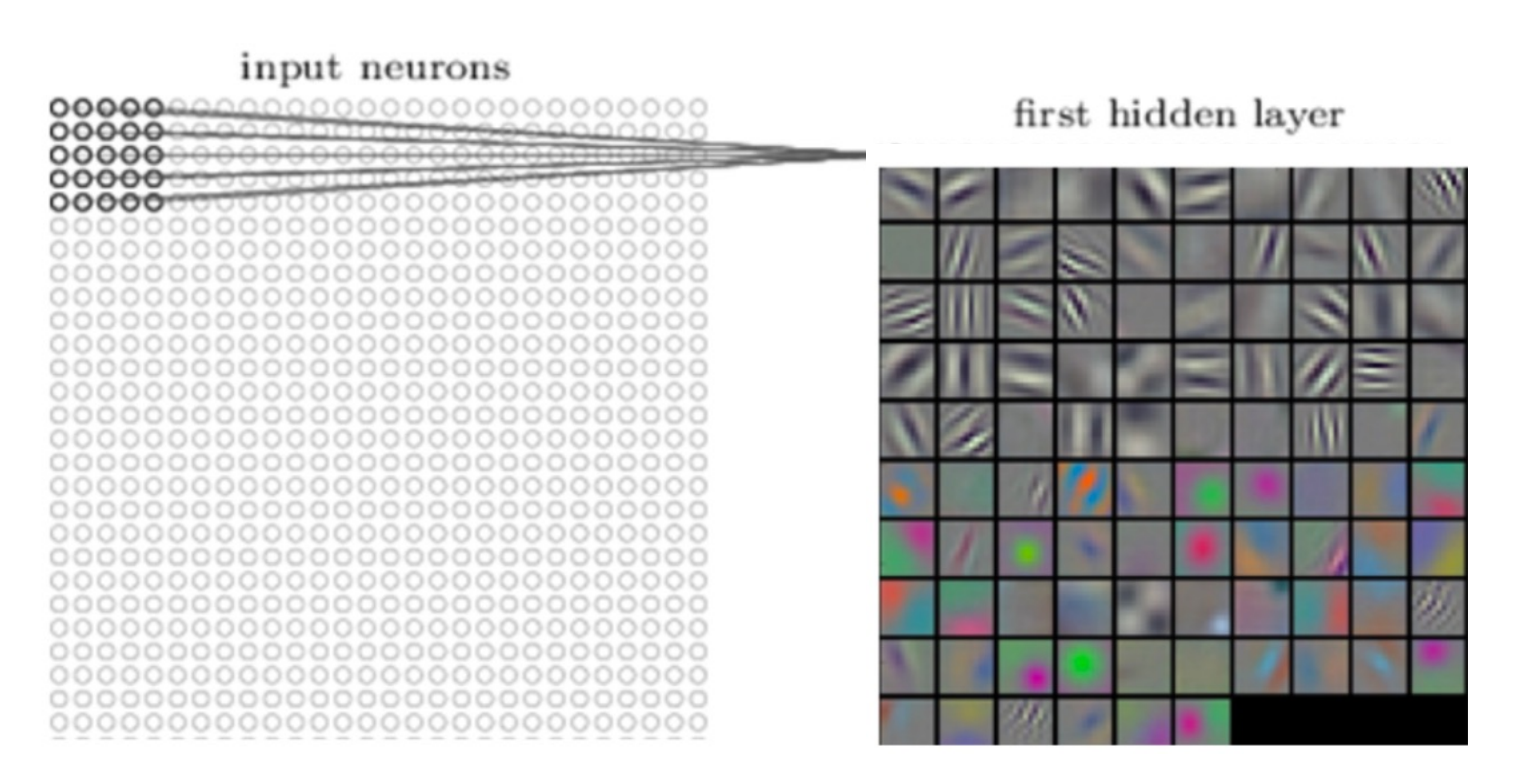
Convolution Layer (Continued)

Conv-Net: Conv-Layers

Convolution operation– Output matrix is called Convolved Feature or Feature Map

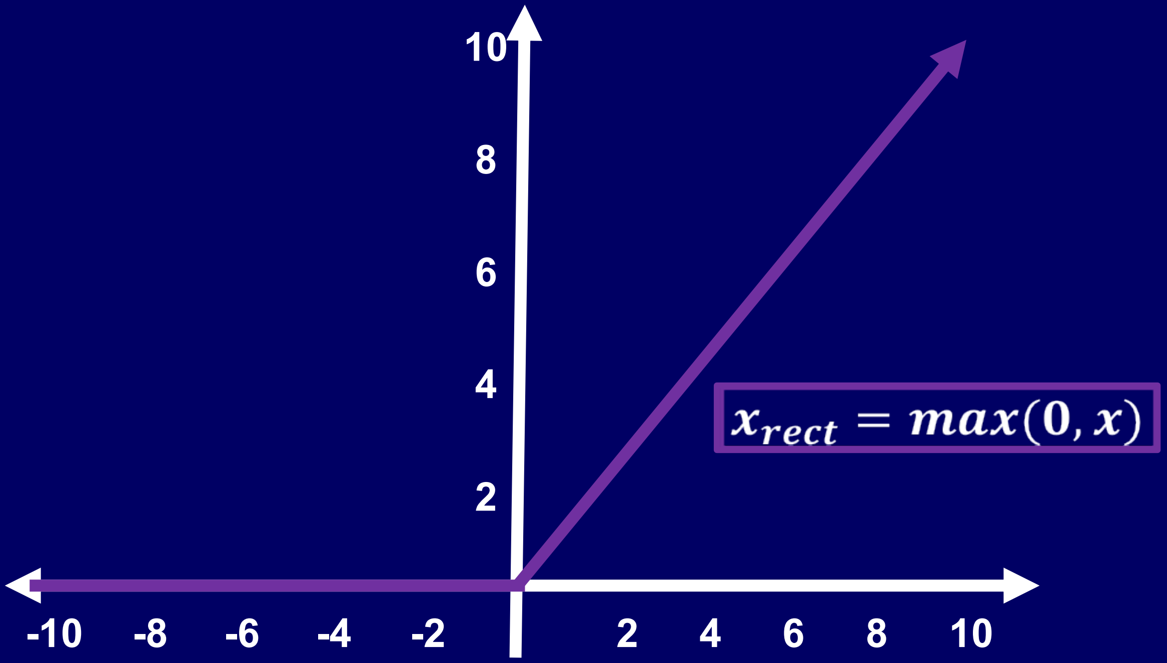
Convolution Layer (Continued)

Conv-Net: Conv-Layers



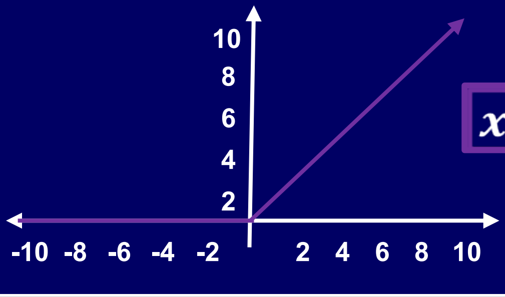
- Element-wise multiplication applied to identical copies of each neuron (aka kernel aka filter)
- Each neurons learn weight for different spatial locations
- Essentially, just a measure of correlation

Conv-Net: Activation Function

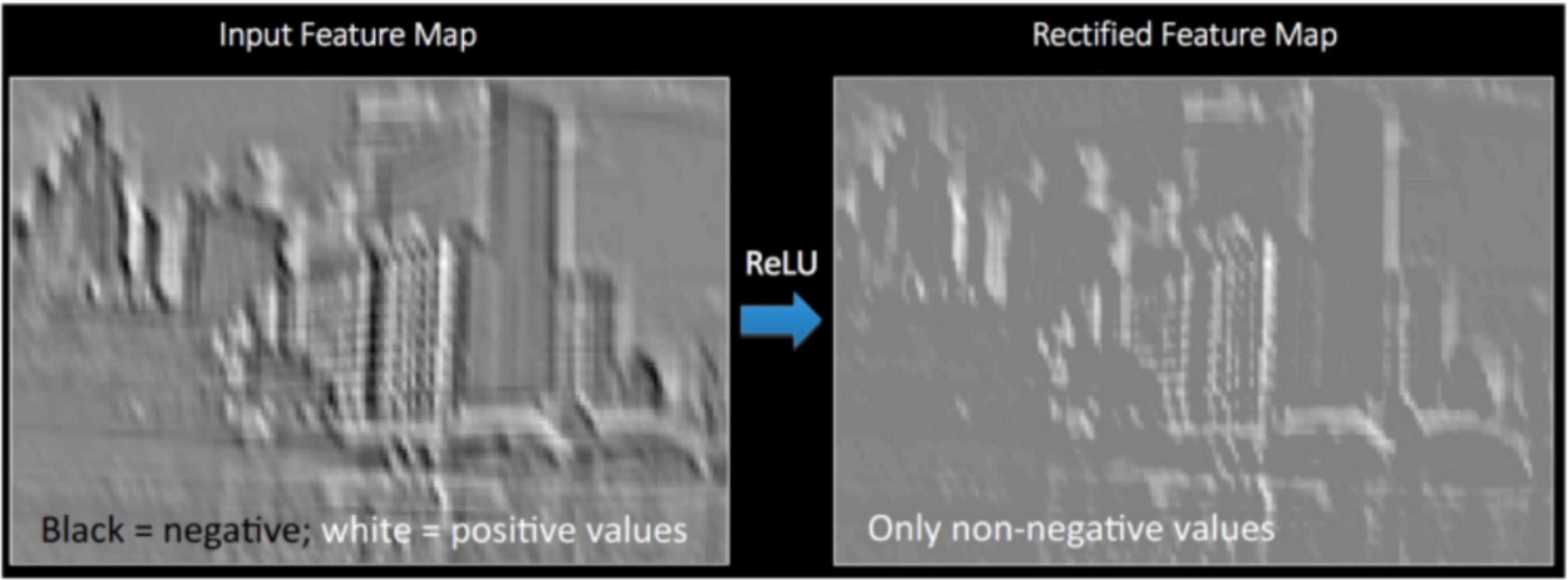


- Introduces non-linearity to representation (i.e., else, conv-layer is linear and unable to capture complex features)
- Just a threshold (e.g., ReLU sets negative values to zero)

Conv-Net: Activation Function



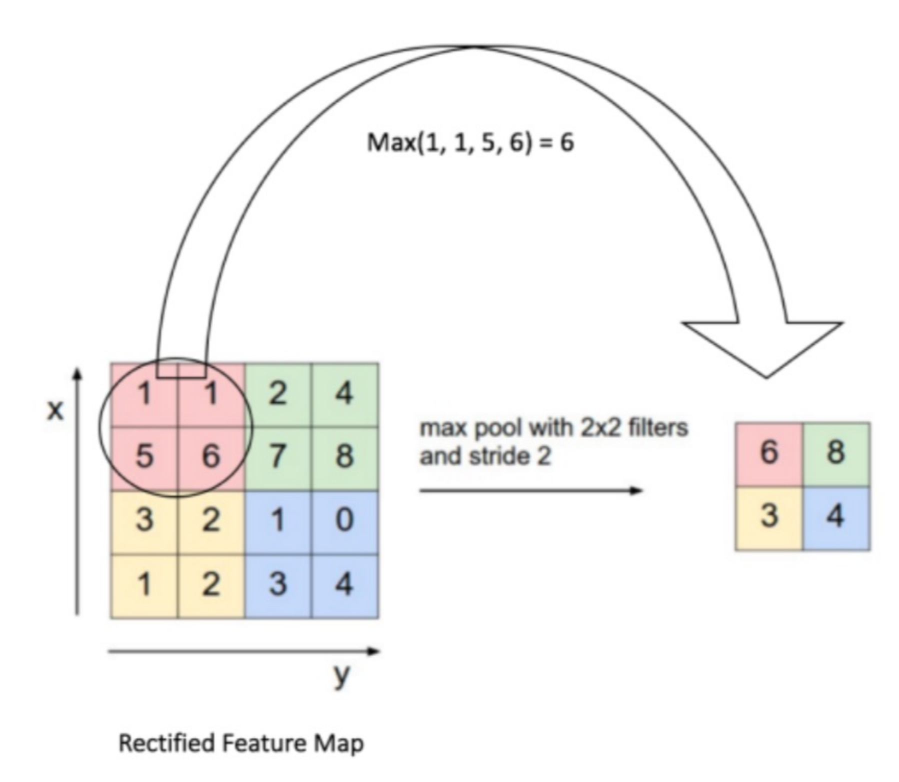
$$x_{rect} = max(0, x)$$



- Introduces non-linearity to representation (i.e., else, conv-layer is linear and unable to capture complex features)

- Just a threshold (e.g., ReLU sets negative values to zero)

Conv-Net: Subsampling (pooling or down-sampling)



1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

6	8
3	4

- Reduces dimension of feature map
- Introduces shift invariance

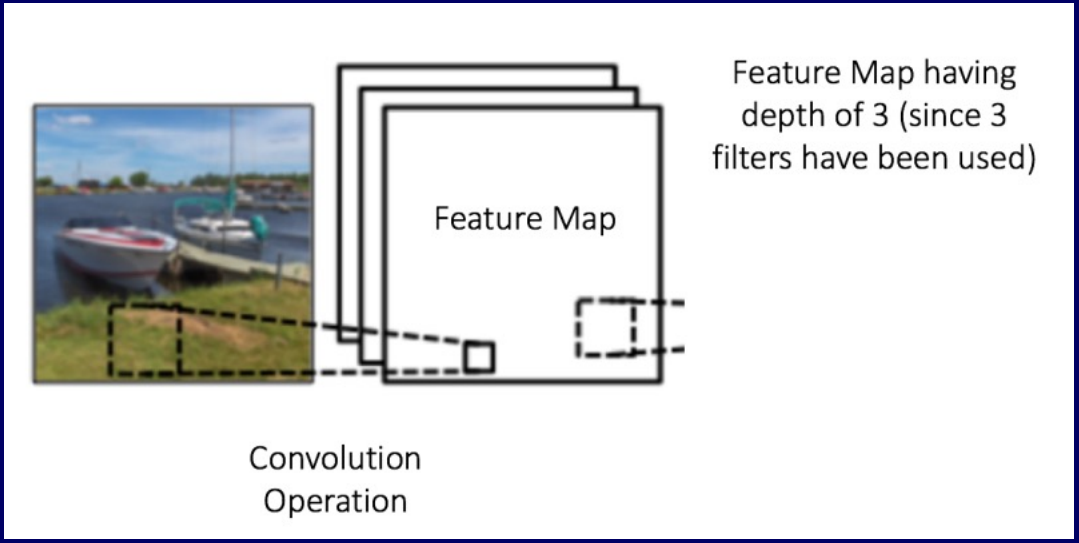
CNNs: Hyper-parameters

- Hyper-parameters: Parameters of parameters, manually set (i.e., typically optimized via cross validation, trial and error, or, less often, transfer expert knowledge from another visual task solved with deep learning)

CNNs: Hyper-parameters

- Hyper-parameters: Parameters of parameters, manually set (i.e., typically optimized via cross validation, trial and error, or, less often, transfer expert knowledge from another visual task solved with deep learning)

1. **Depth**. Number of filters used at a given layer.


Feature Map having depth of 3 (since 3 filters have been used)
Feature Map
Convolution Operation

CNNs: Hyper-parameters

- Hyper-parameters: Parameters of parameters, manually set (i.e., typically optimized via cross validation, trial and error, or, less often, transfer expert knowledge from another visual task solved with deep learning)

1. **Depth**. Number of filters used at a given layer.
2. **Stride**. Number of neurons to skip between filters. Double the stride results in ½ sized feature maps.

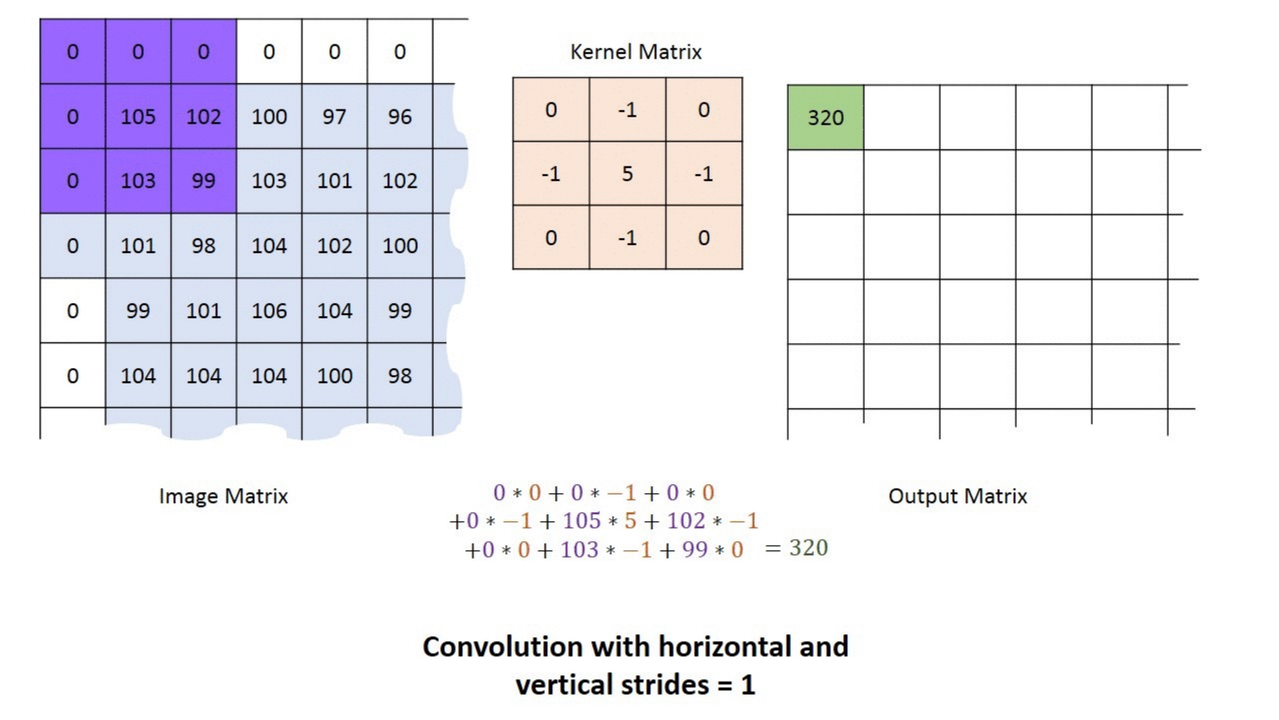


Image Matrix

0	0	0	0	0	0
0	105	102	100	97	96
0	103	99	103	101	102
0	101	98	104	102	100
0	99	101	106	104	99
0	104	104	104	100	98

Kernel Matrix

0	-1	0
-1	5	-1
0	-1	0

Output Matrix

320				

$$0*0+0*-1+0*0 \\ +0*-1+105*5+102*-1 \\ +0*0+103*-1+99*0 \quad = 320$$

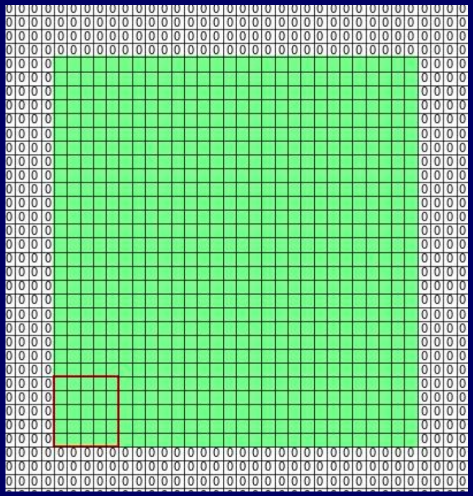
Convolution with horizontal and vertical strides = 1

CNNs: Hyper-parameters

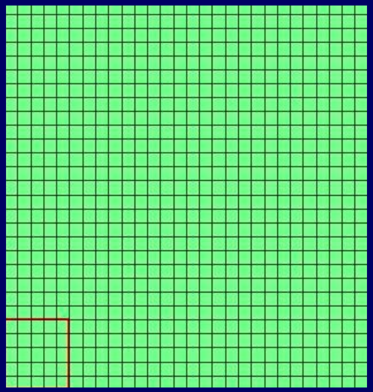
- Hyper-parameters: Parameters of parameters, manually set (i.e., typically optimized via cross validation, trial and error, or, less often, transfer expert knowledge from another visual task solved with deep learning)

1. **Depth**. Number of filters used at a given layer.
2. **Stride**. Number of neurons to skip between filters. Double the stride results in ½ sized feature maps.
3. **Zero padding.** Padding borders with zeros is a means of tailoring edge values. Moreover, it provides a way of controlling size of output.

Zero-Padded

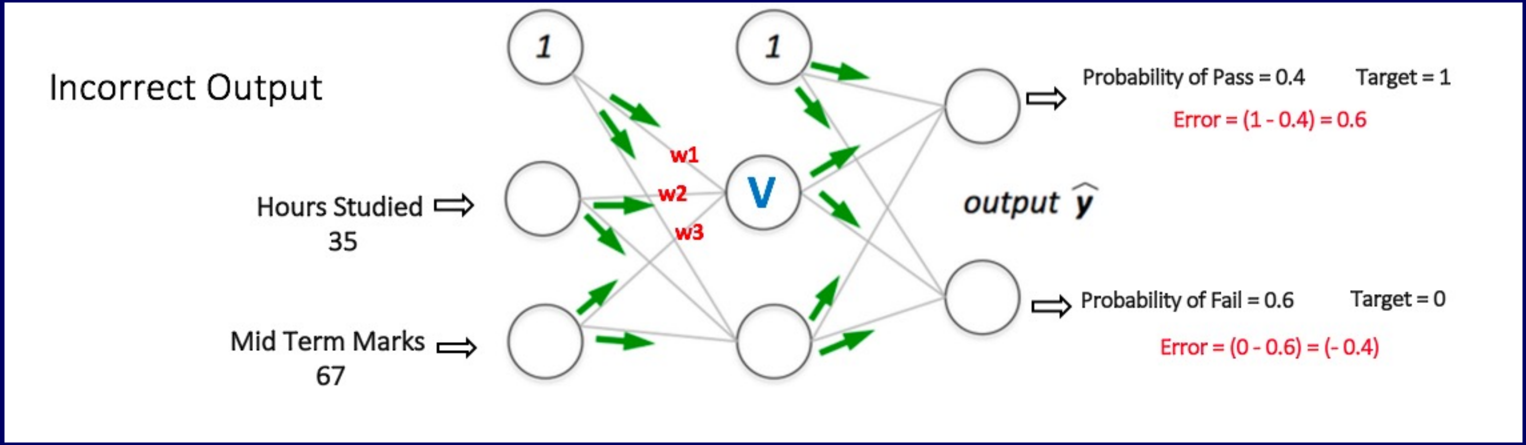


No Zero-Pad



CNNs: Feed Forward

Feed-Forward



Incorrect Output

Hours Studied 35

Mid Term Marks 67

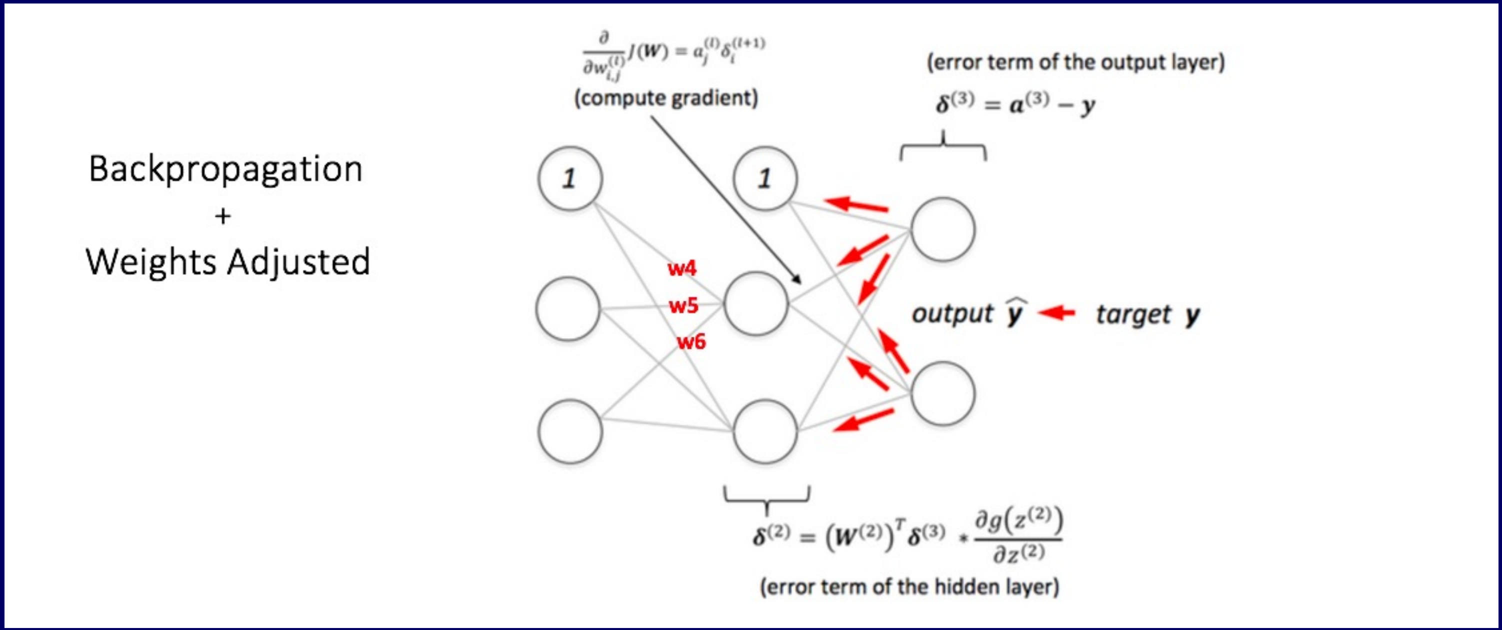
Probability of Pass = 0.4 Target = 1

Error = (1 - 0.4) = 0.6

Probability of Fail = 0.6 Target = 0

Error = (0 - 0.6) = (- 0.4)

Back Propagation



Backpropagation
+
Weights Adjusted

$$\frac{\partial}{\partial w_{i,j}^{(l)}} J(W) = a_j^{(l)} \delta_i^{(l+1)}$$

(compute gradient)

(error term of the output layer)

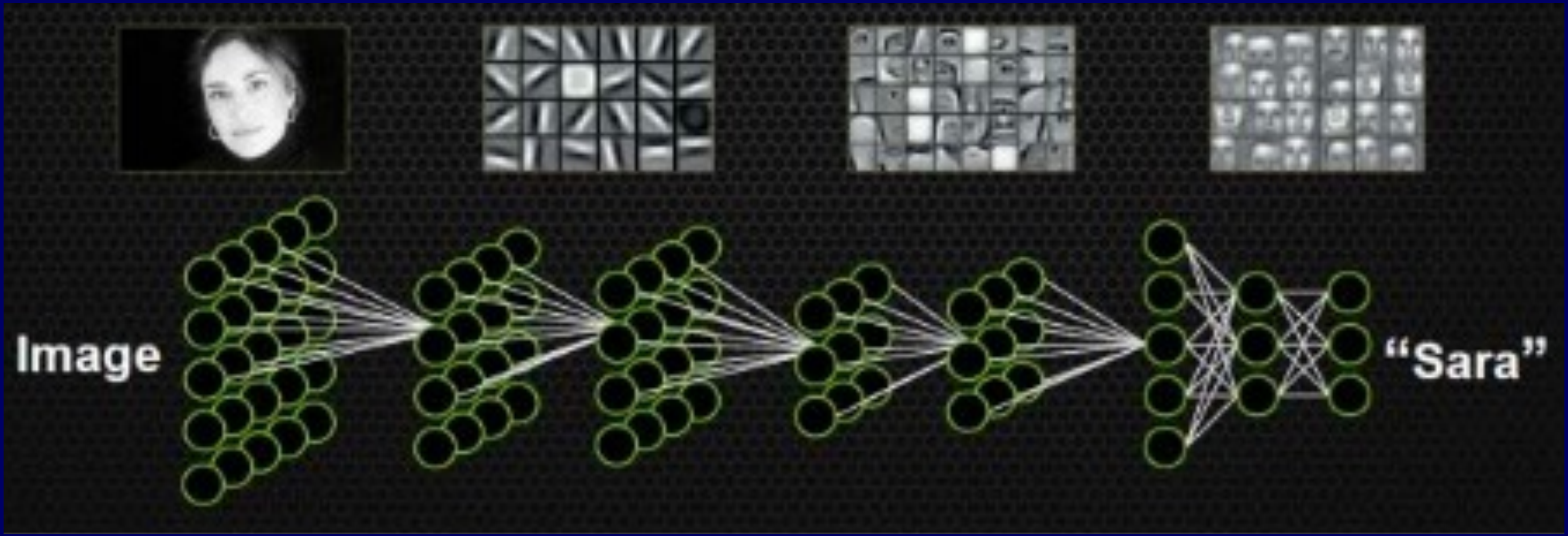
$$\boldsymbol{\delta}^{(3)} = \boldsymbol{a}^{(3)} - \boldsymbol{y}$$

$$\boldsymbol{\delta}^{(2)} = \left(\boldsymbol{W}^{(2)}\right)^T \boldsymbol{\delta}^{(3)} * \frac{\partial g\left(z^{(2)}\right)}{\partial z^{(2)}}$$

(error term of the hidden layer)

What is learned at Different Levels

Conv-Net: Learned Features



- Higher-level abstraction captured at higher layers
- Lower level features are combined at proceeding layers