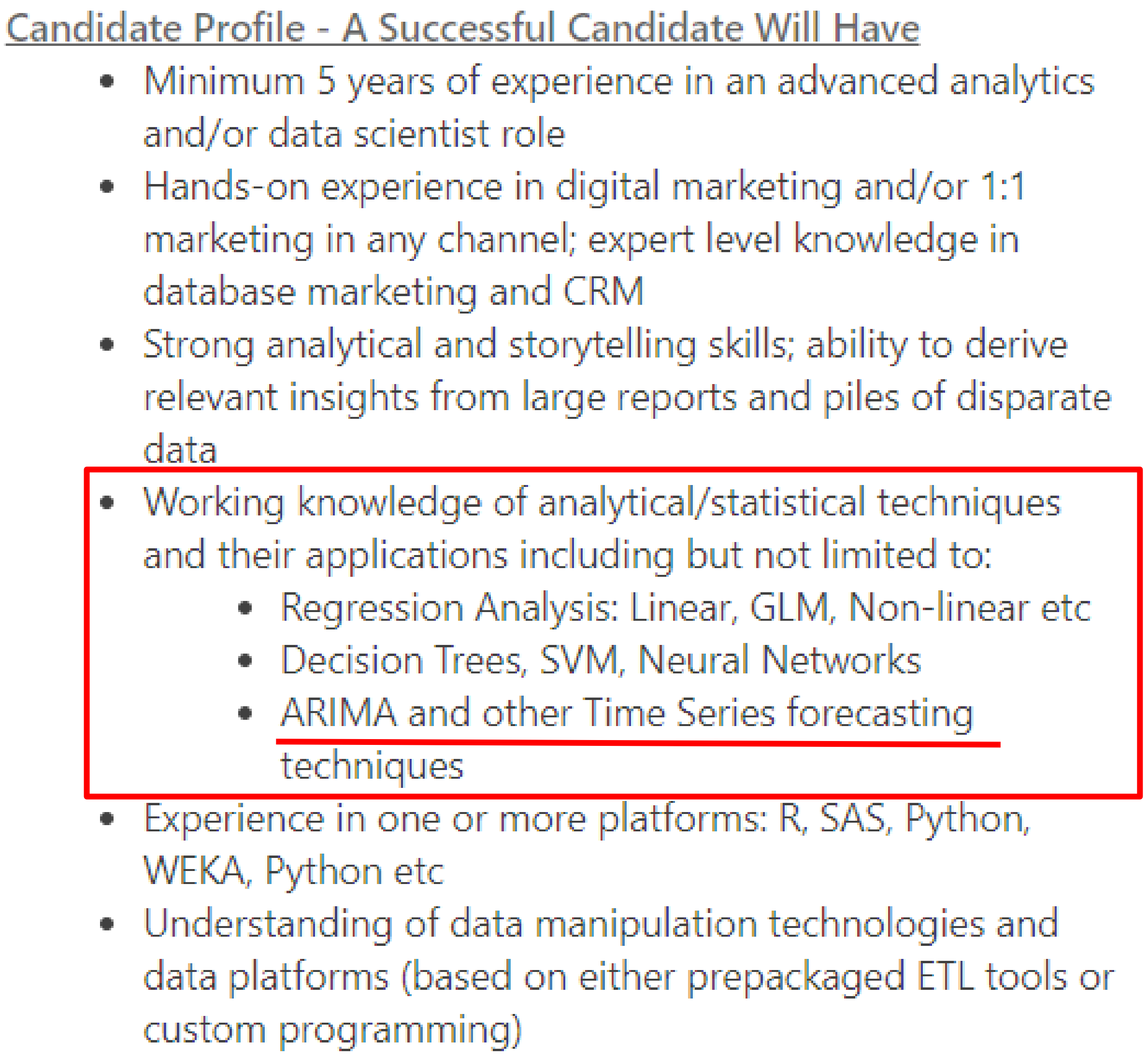
Tutorial 7
Convolutional Neural Networks (CNN)

FeedForward and Backpropagation

- Open-source References
  - Step-by-step Code:
    - https://machinelearningmastery.com/implement-backpropagation-algorithm-scratch-python/
    - May help you better understand “backpropagation” in a deep neural network

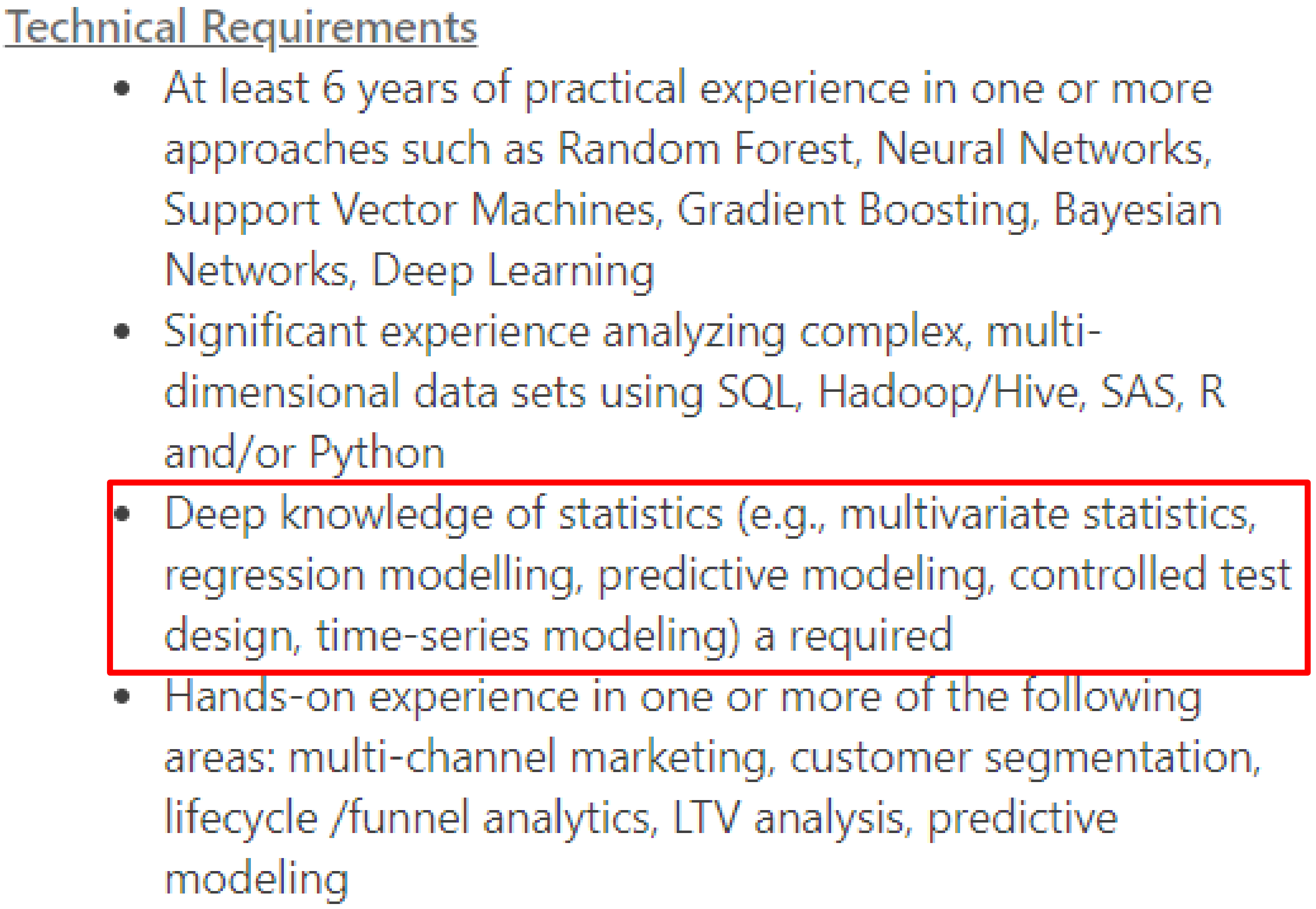
Time Series, Neural Networks and Deep Learning
Why Should We Learn Them?

Data Scientist Job Requirements on Linkedin



Candidate Profile - A Successful Candidate Will Have

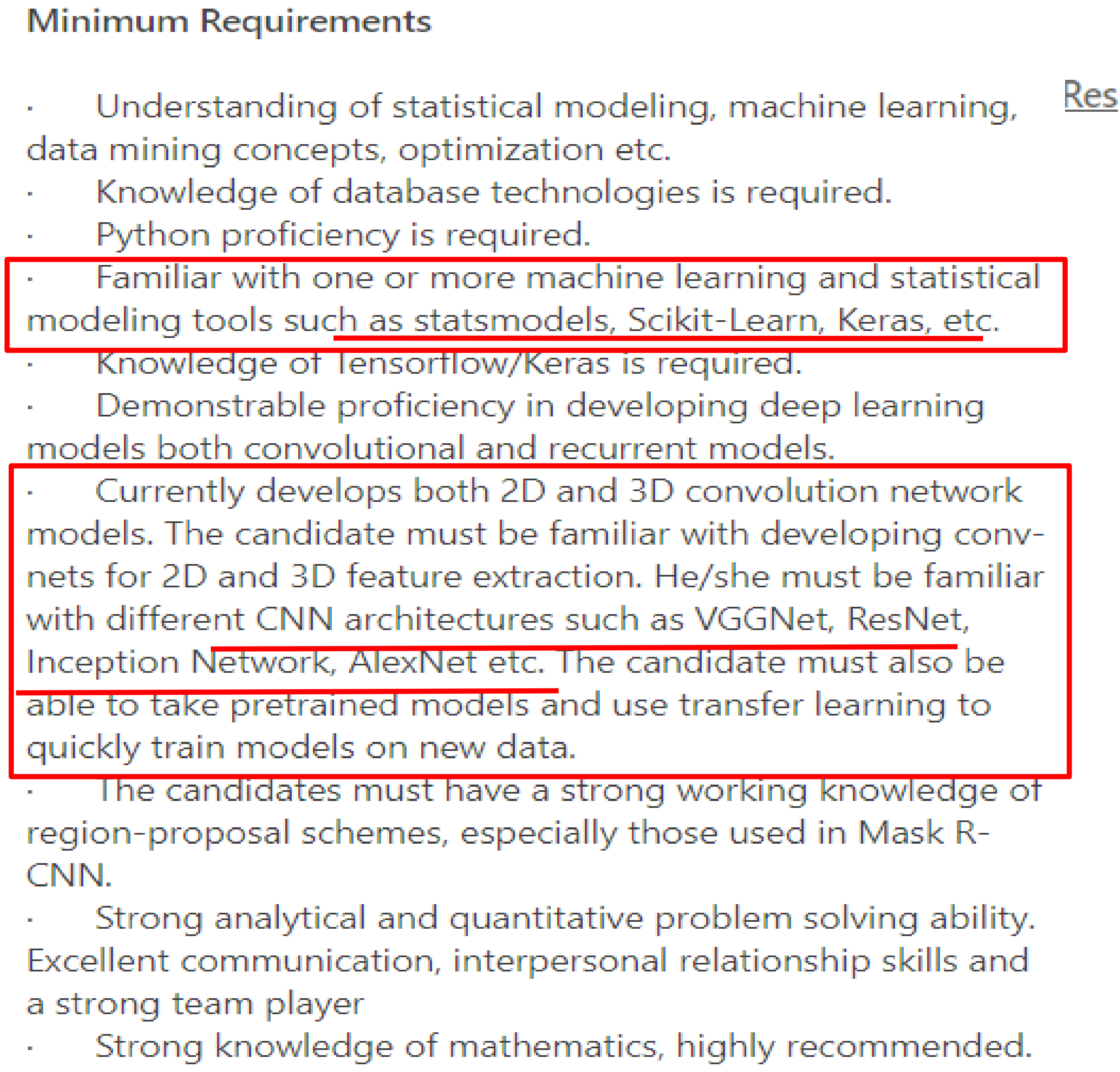
- Minimum 5 years of experience in an advanced analytics and/or data scientist role
- Hands-on experience in digital marketing and/or 1:1 marketing in any channel; expert level knowledge in database marketing and CRM
- Strong analytical and storytelling skills; ability to derive relevant insights from large reports and piles of disparate data
- Working knowledge of analytical/statistical techniques and their applications including but not limited to:
  - Regression Analysis: Linear, GLM, Non-linear etc
  - Decision Trees, SVM, Neural Networks
  - ARIMA and other Time Series forecasting techniques
- Experience in one or more platforms: R, SAS, Python, WEKA, Python etc
- Understanding of data manipulation technologies and data platforms (based on either prepackaged ETL tools or custom programming)



Technical Requirements

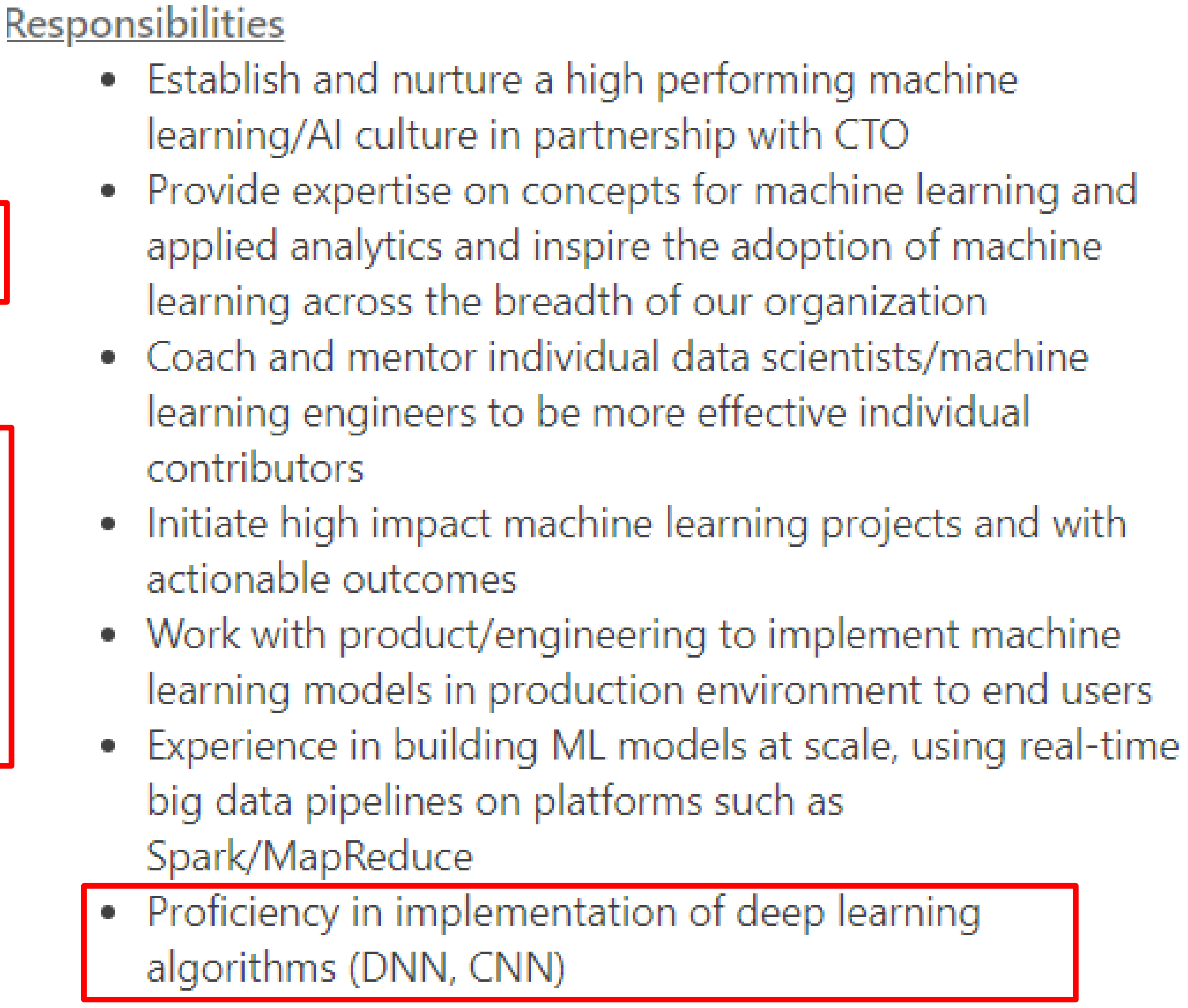
- At least 6 years of practical experience in one or more approaches such as Random Forest, Neural Networks, Support Vector Machines, Gradient Boosting, Bayesian Networks, Deep Learning
- Significant experience analyzing complex, multi-dimensional data sets using SQL, Hadoop/Hive, SAS, R and/or Python
- Deep knowledge of statistics (e.g., multivariate statistics, regression modelling, predictive modeling, controlled test design, time-series modeling) a required
- Hands-on experience in one or more of the following areas: multi-channel marketing, customer segmentation, lifecycle /funnel analytics, LTV analysis, predictive modeling

Data Scientist Job Requirements on Linkedin



Minimum Requirements

- Understanding of statistical modeling, machine learning, data mining concepts, optimization etc.
- Knowledge of database technologies is required.
- Python proficiency is required.
- Familiar with one or more machine learning and statistical modeling tools such as statsmodels, Scikit-Learn, Keras, etc.
- Knowledge of Tensorflow/Keras is required.
- Demonstrable proficiency in developing deep learning models both convolutional and recurrent models.
- Currently develops both 2D and 3D convolution network models. The candidate must be familiar with developing conv-nets for 2D and 3D feature extraction. He/she must be familiar with different CNN architectures such as VGGNet, ResNet, Inception Network, AlexNet etc. The candidate must also be able to take pretrained models and use transfer learning to quickly train models on new data.
- The candidates must have a strong working knowledge of region-proposal schemes, especially those used in Mask R-CNN.
- Strong analytical and quantitative problem solving ability. Excellent communication, interpersonal relationship skills and a strong team player
- Strong knowledge of mathematics, highly recommended.



Responsibilities

- Establish and nurture a high performing machine learning/AI culture in partnership with CTO
- Provide expertise on concepts for machine learning and applied analytics and inspire the adoption of machine learning across the breadth of our organization
- Coach and mentor individual data scientists/machine learning engineers to be more effective individual contributors
- Initiate high impact machine learning projects and with actionable outcomes
- Work with product/engineering to implement machine learning models in production environment to end users
- Experience in building ML models at scale, using real-time big data pipelines on platforms such as Spark/MapReduce
- Proficiency in implementation of deep learning algorithms (DNN, CNN)

Agenda

- Basic CNN framework
    - Filter/Kernel, Convolution Layer, Feature Map, Padding, Pooling, etc.
- Discussion about Programming Assignment 7
    - Build CNN model; Multi-class classification and prediction
- Python Implementation
    - Tensorflow and Keras
- Open-source References
    - Book: https://www.deeplearningbook.org/
    - Course (by Andrew Ng, Stanford U.):
        - http://cs231n.stanford.edu/
        - https://www.youtube.com/playlist?list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv

In the lecture:

Variants of Neural Networks:

Autoencoder

Restricted Boltzmann Machine (RBM)

Radial Basis Function Network (RBF)

Convolutional Neural Network (CNN): Computer vision, Natural language processing, etc.

Recurrent Neural Network (RNN)
- Long Short-term Memory (LSTM)

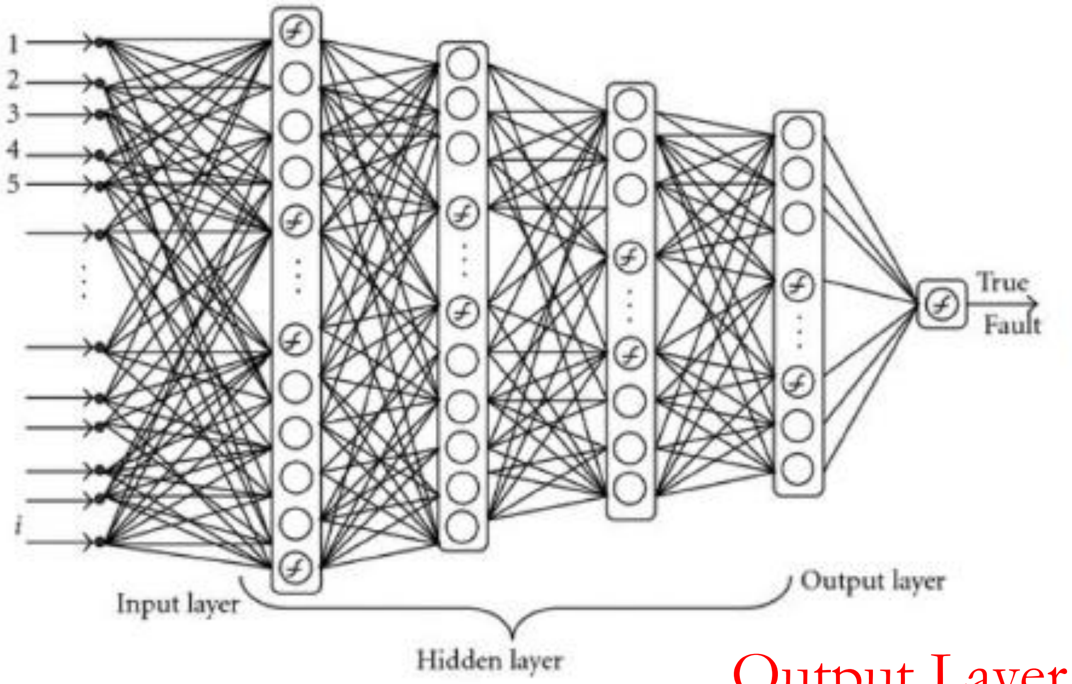
Deep Belief Network (DBN)

Neural Network and Convolutional Neural Network Similarities & Differences

A Normal Neural Network (Full Connected)

NN (perceptron) consists of three layers:

Input data →



→ Output data

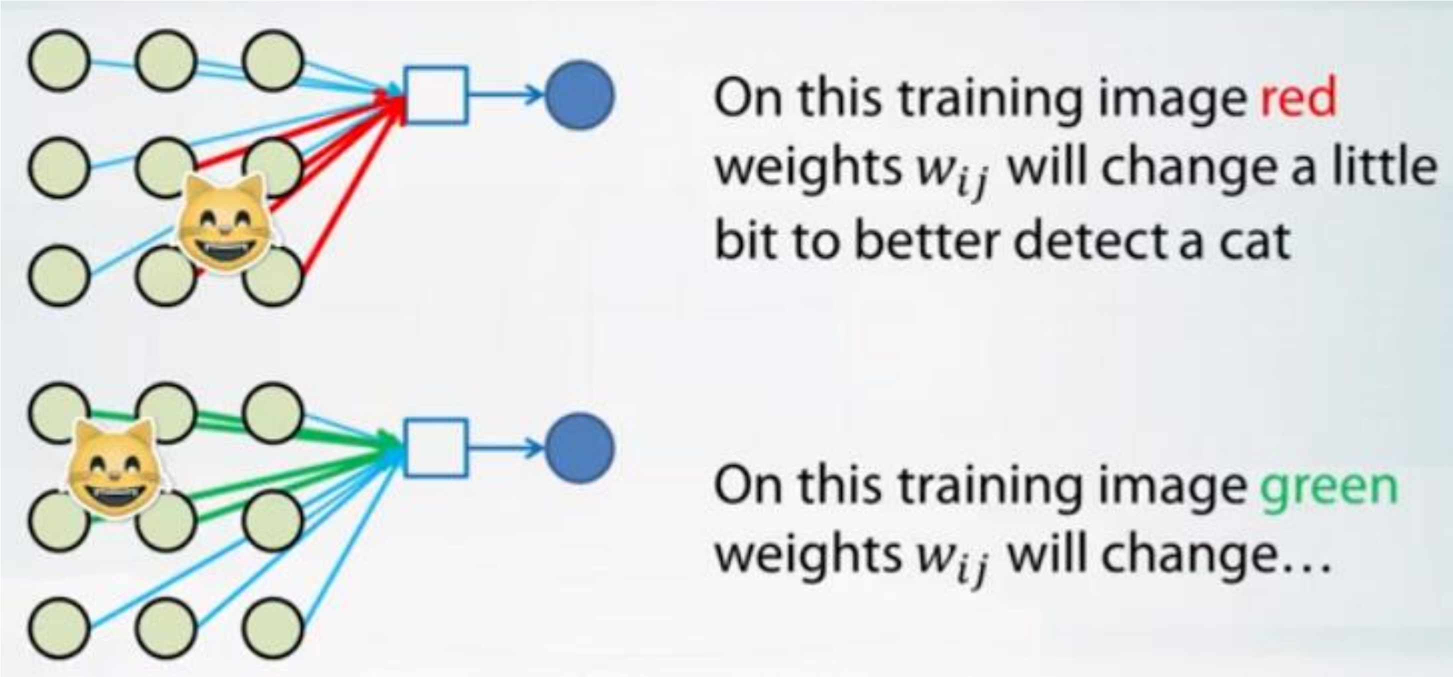
Input Layer

Hidden Layers

Output Layer

A Normal Neural Network (Full Connected)

- What if we use a normal full-connected neural network to do classification?
- We split the whole image into multiple pixels. Each pixel (a value to represent darkness or brightness of color: 0 to 255) is one feature.
- What is wrong with it?



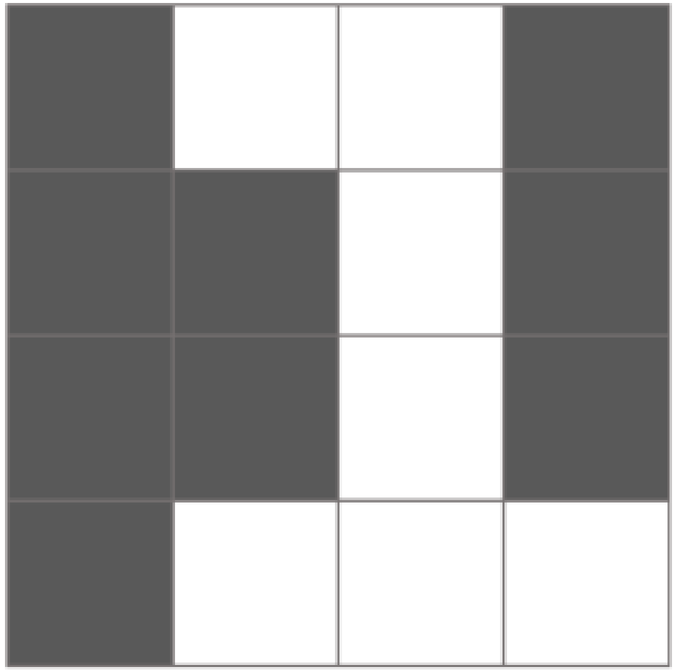
On this training image red weights w_{ij} will change a little bit to better detect a cat

On this training image green weights w_{ij} will change…

- We don’t fully utilize the training data
- What if in test data, the cat is in other areas (e.g., in the centre of the image)?

A Normal Neural Network (Full Connected)

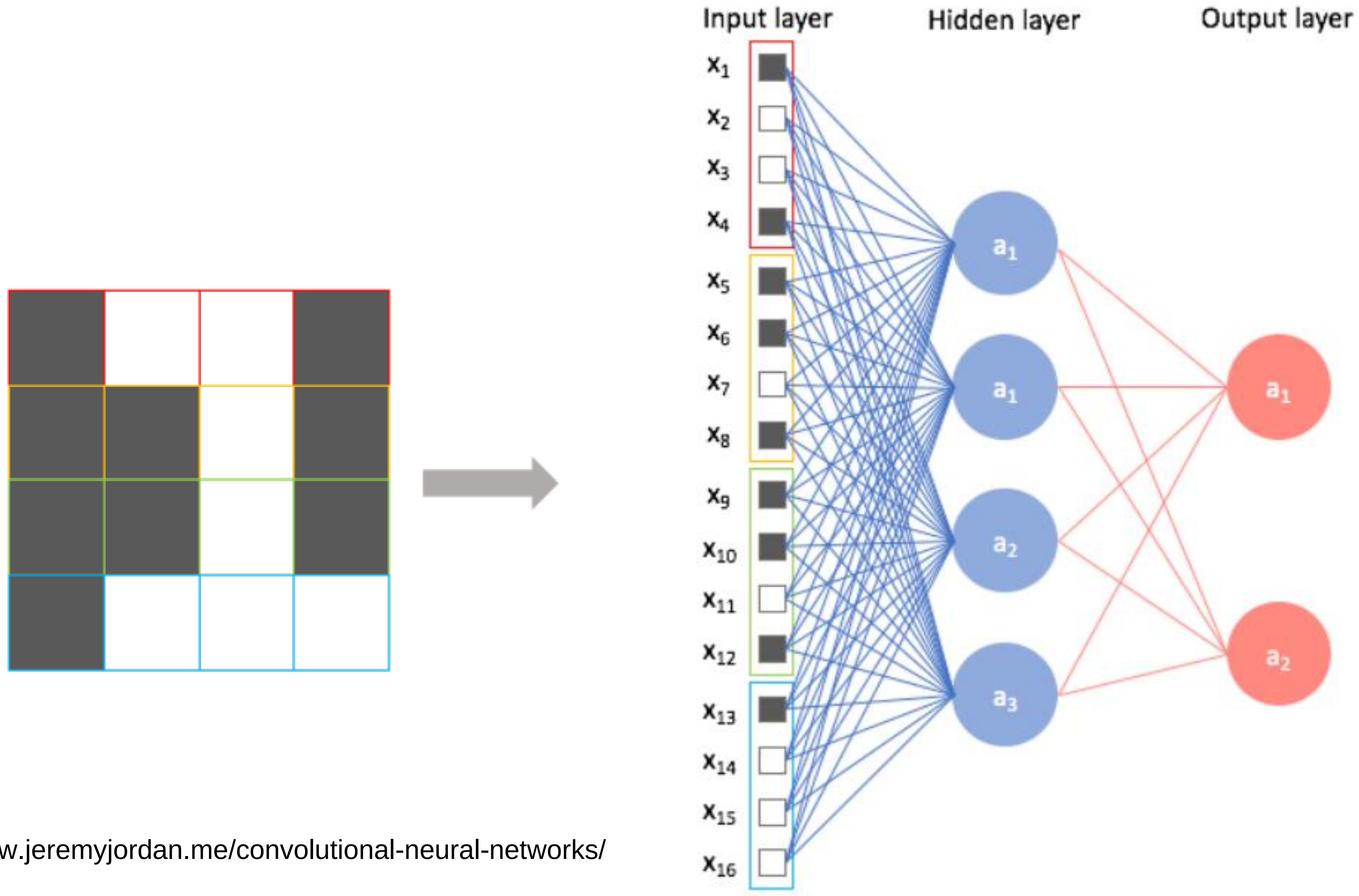
- What if we use a normal full-connected neural network to do classification?
- We split the whole image into multiple pixels. Each pixel (a value to represent darkness or brightness of color: 0 to 255) is one feature.
- What is wrong with it?



This is handwritten digit number “1”

https://www.jeremyjordan.me/convolutional-neural-networks/

A Normal Neural Network (Full Connected)

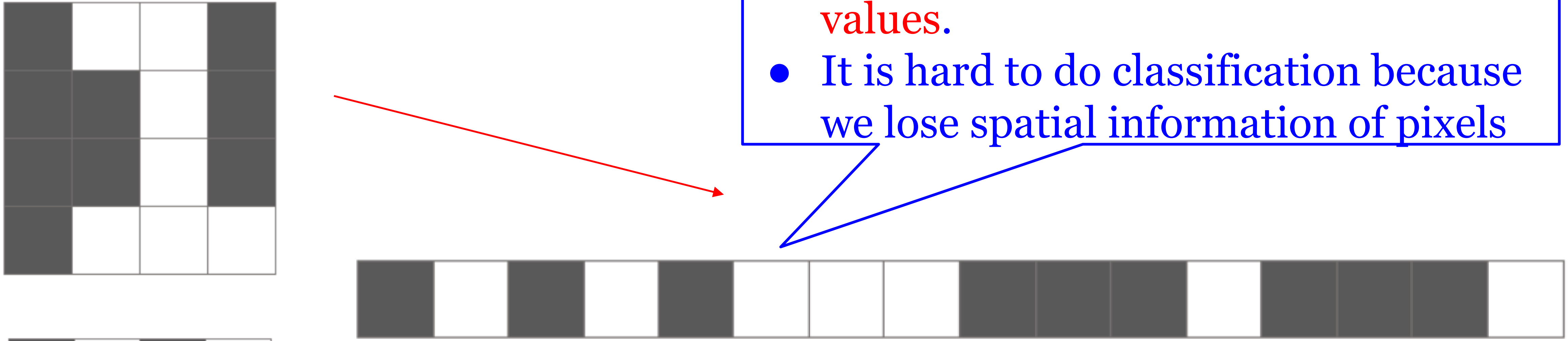


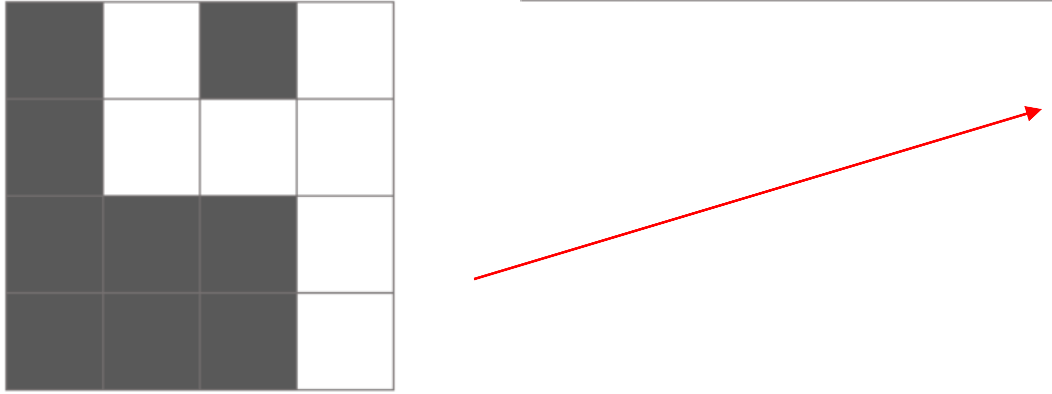
https://www.jeremyjordan.me/convolutional-neural-networks/

A Normal Neural Network

- What is wrong with it?

- If you split image number “1” and image number “4” into pixels, you may get the same feature row of values.
- It is hard to do classification because we lose spatial information of pixels



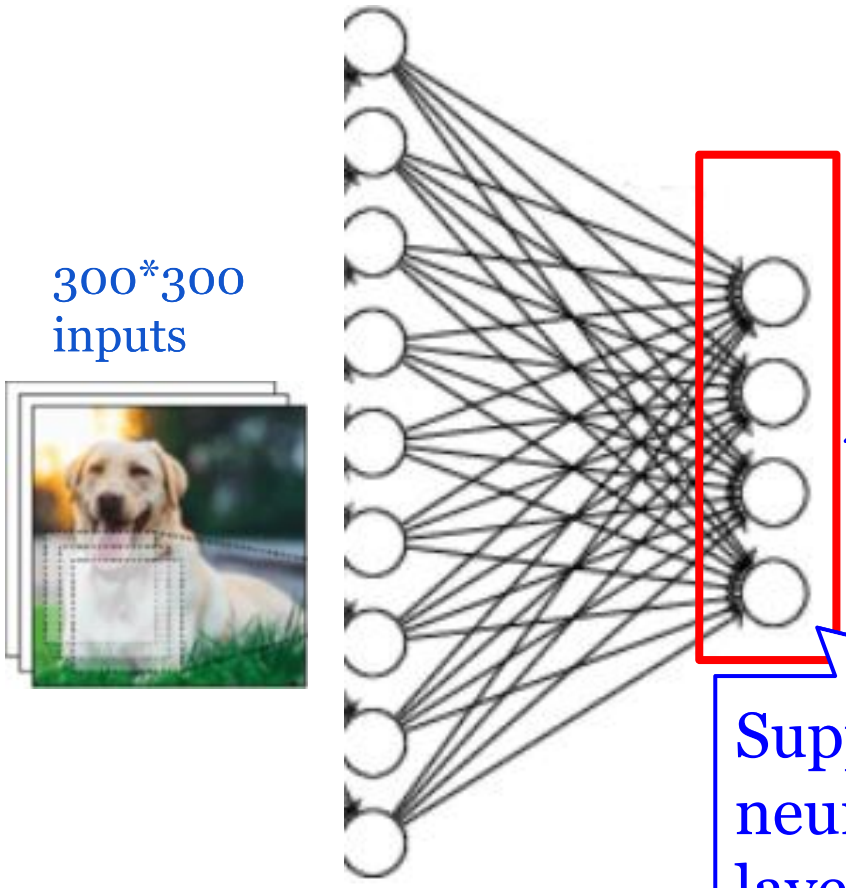


https://www.jeremyjordan.me/convolutional-neural-networks/

A Normal Neural Network (Full Connected)

- How many parameters are you training?

Fully Connected Normal NN



300*300
inputs

300*300*4+1
About 360,001
weights

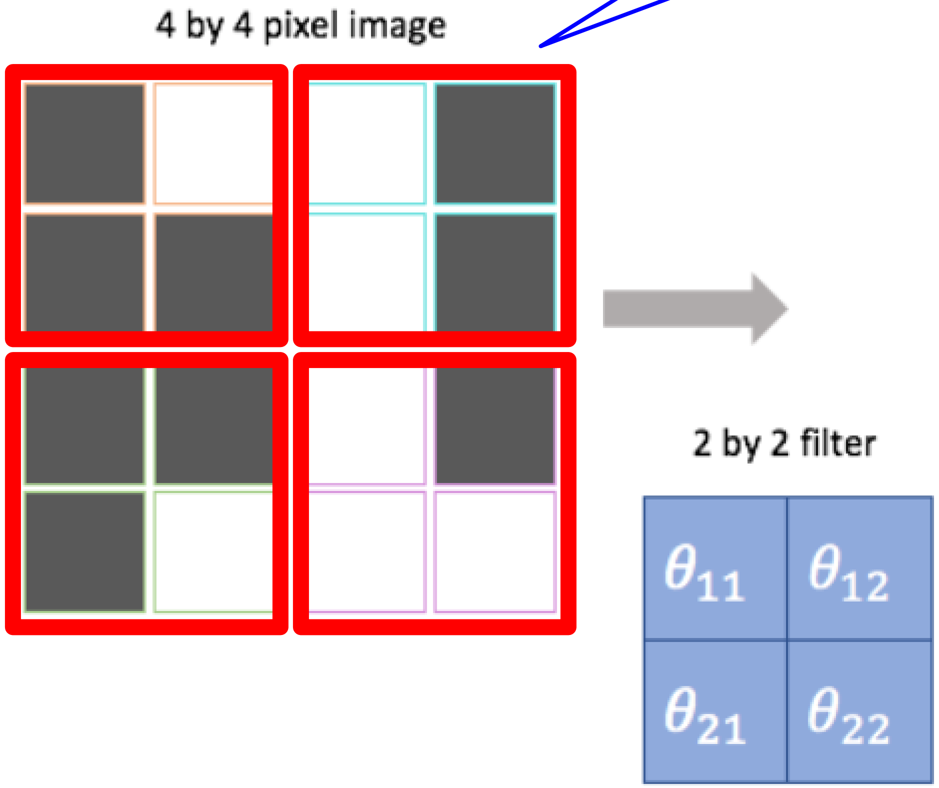
Suppose you have four neurons in the hidden layer

Fully-Connected Neural Network in Computer Vision:

- Slow
- Overfit

Convolutional Neural Network

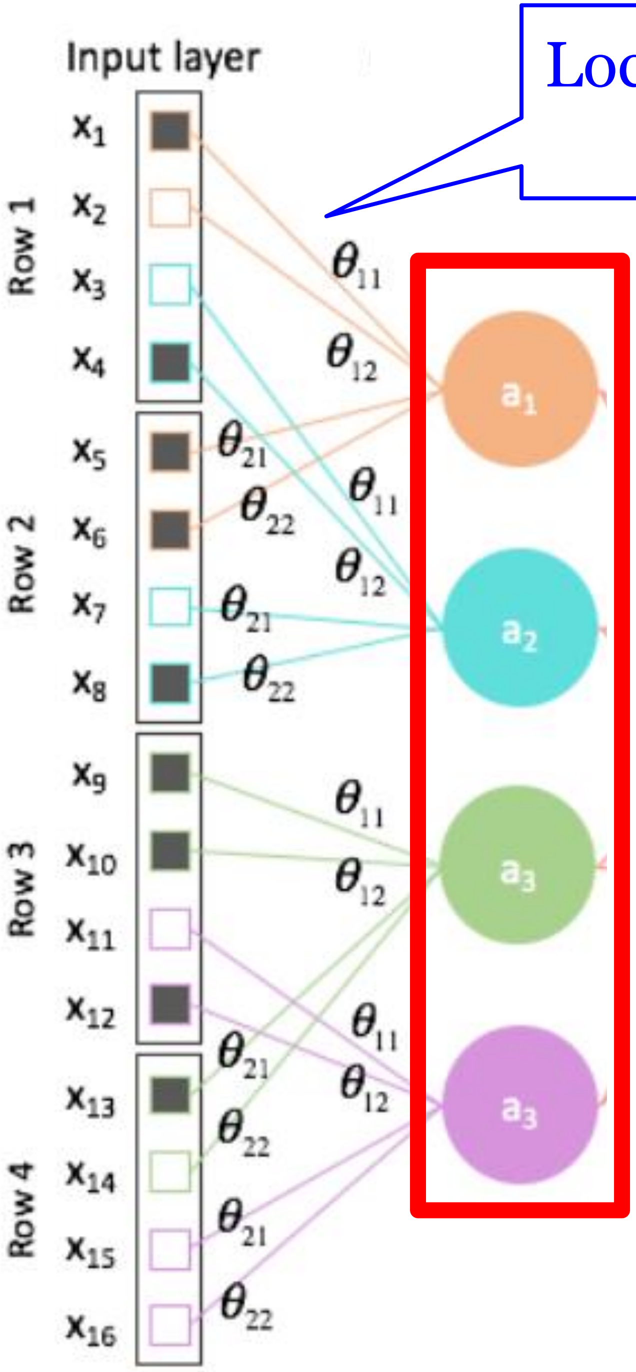
Spatially organized



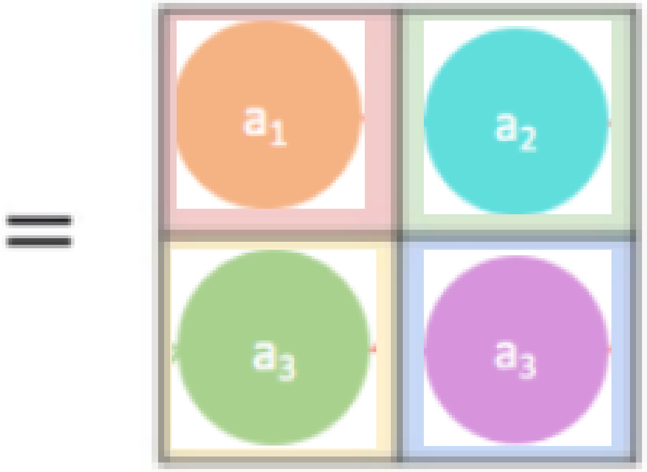
Convolution Layer (Filter/Kernel):

Suppose Slide Stride=2

Local/Sparse Connectivity: Sharing weights



Because interesting features (edges) can happen at anywhere in the image.



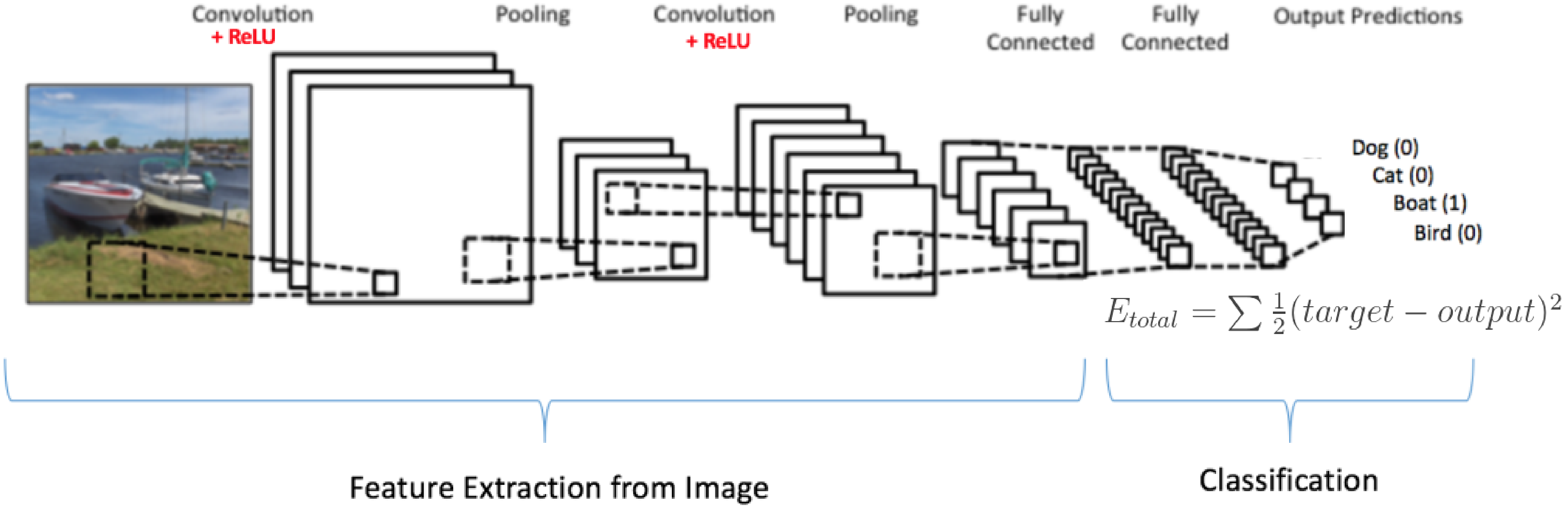
Feature Map:
with new “pixels”

https://www.jeremyjordan.me/convolutional-neural-networks/

Convolutional Neural Network (CNN)

CNN-Architecture

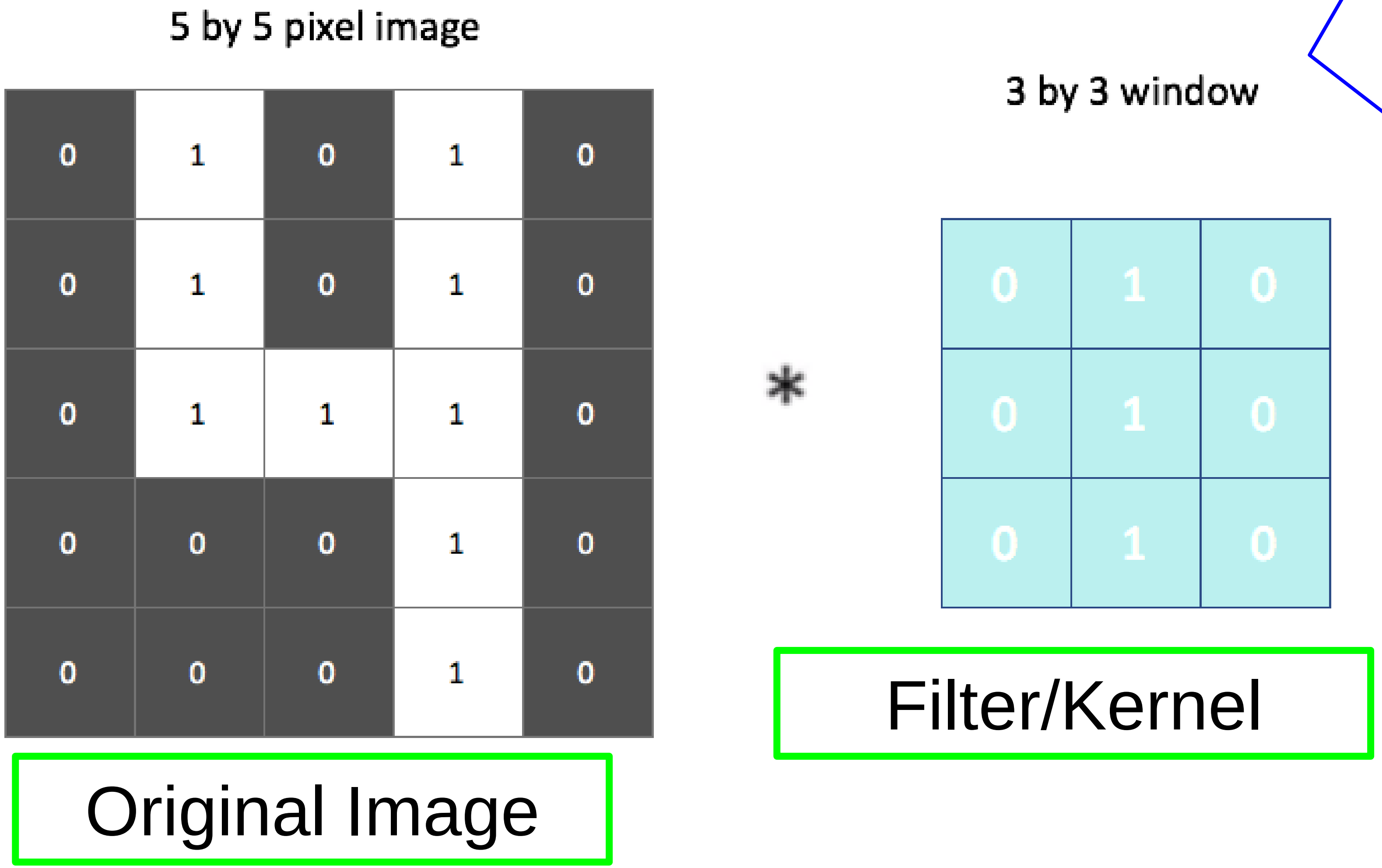
- A typical CNN architecture is composed of:
  Original image, Convolution layer (filters/kernels), Feature map, Activation function (ReLU), Pooling (subsampling), Fully-Connected (FC) layer



CNN Building Blocks
Convolution Layer Filters/Kernels

CNN-Filter/Kernel

- Filter/Kernel: A sliding window



5 by 5 pixel image

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

Original Image

3 by 3 window

0	1	0
0	1	0
0	1	0

Filter/Kernel

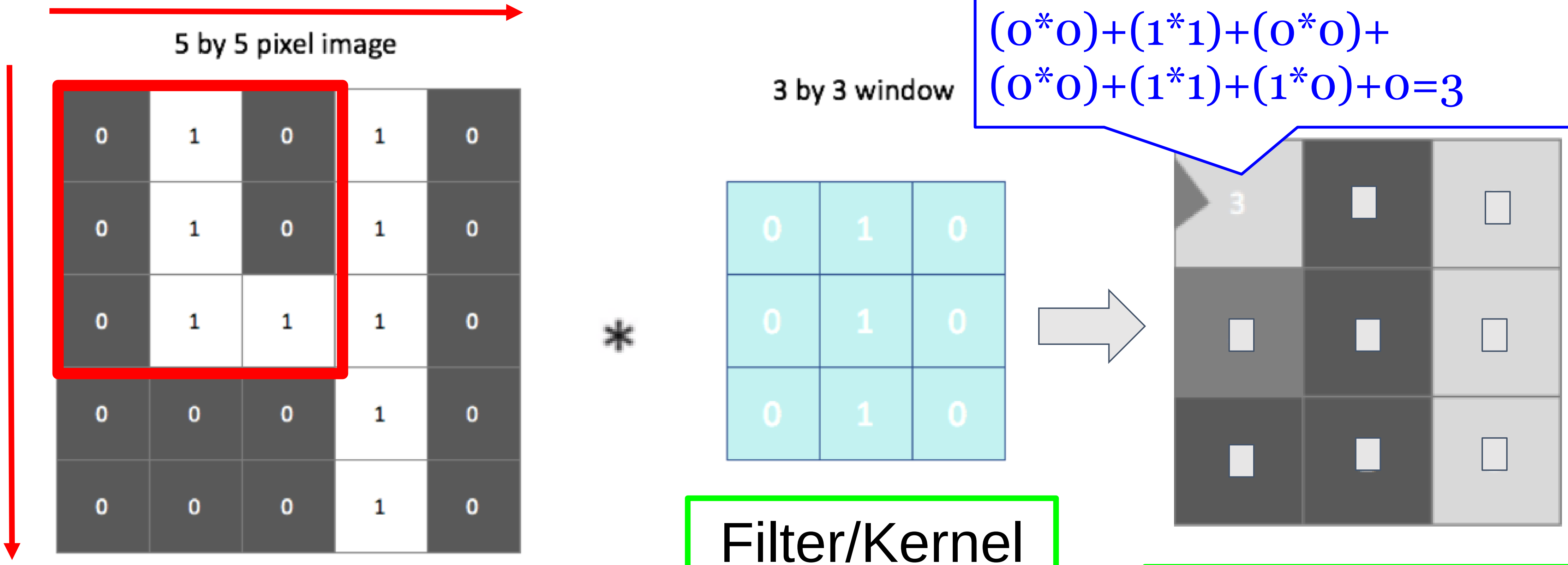
- Learn the image "patch" by "patch"
- Use a "window" to slide across the original image
- Dot product (i.e., element-wise multiplication and then sum up)
- Then we get an extracted feature (i.e., feature map)
- We can try different slide strides

CNN-Filter/Kernel

- Move the window by one step each time

Dot product (Suppose bias/intercept=0 and stride=1):

(0*0)+(1*1)+(0*0)+
(0*0)+(1*1)+(0*0)+
(0*0)+(1*1)+(1*0)+0=3



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

0	1	0
0	1	0
0	1	0

3	□	□
□	□	□
□	□	□

Original Image

Filter/Kernel

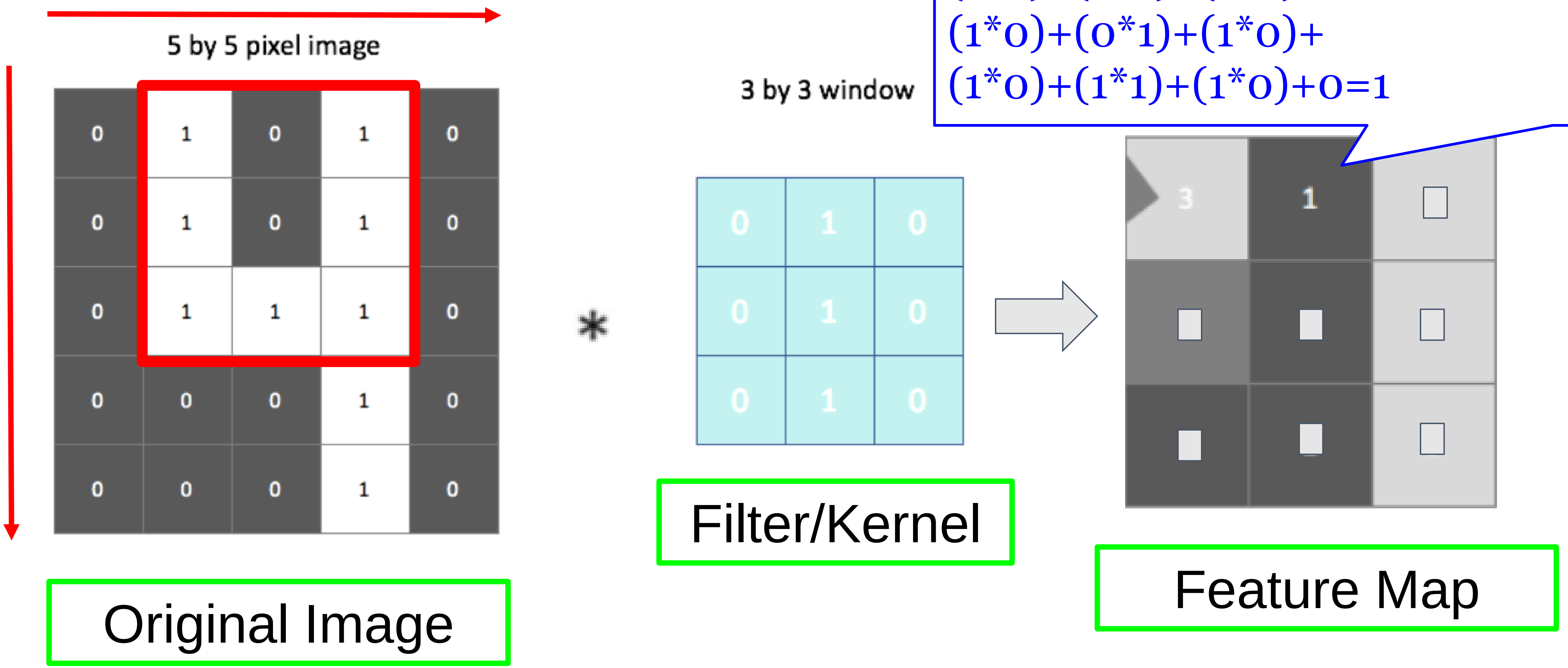
Feature Map

CNN-Filter/Kernel

- Move the window by one step each time

Dot product (Suppose bias/intercept=0 and stride=1):

(1*0)+(0*1)+(1*0)+
(1*0)+(0*1)+(1*0)+
(1*0)+(1*1)+(1*0)+0=1



5 by 5 pixel image

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

Original Image

*

3 by 3 window

0	1	0
0	1	0
0	1	0

Filter/Kernel

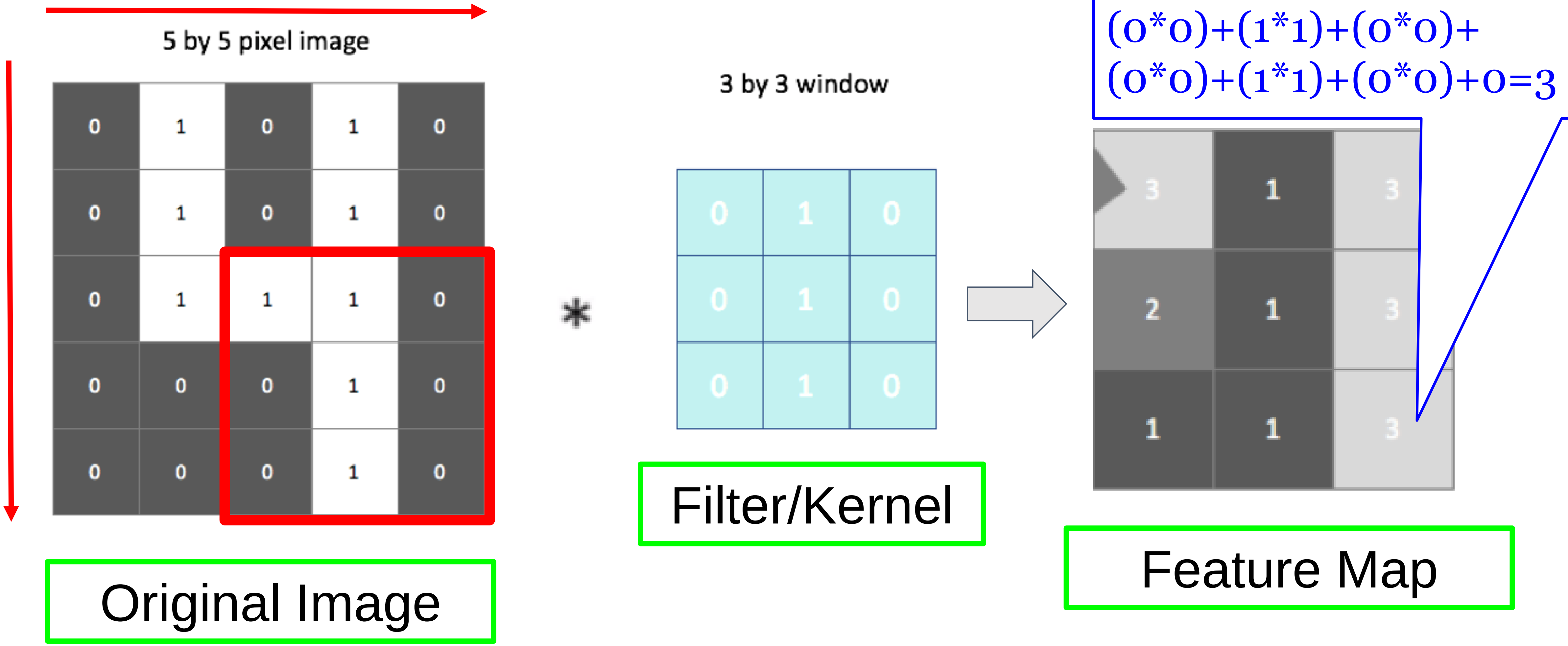
3	1	□
□	□	□
□	□	□

Feature Map

CNN-Filter/Kernel

- Finally:

Dot product (Suppose bias/intercept=0 and stride=1): (1*0)+(1*1)+(0*0)+ (0*0)+(1*1)+(0*0)+ (0*0)+(1*1)+(0*0)+0=3



5 by 5 pixel image

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

Original Image

*

3 by 3 window

0	1	0
0	1	0
0	1	0

Filter/Kernel

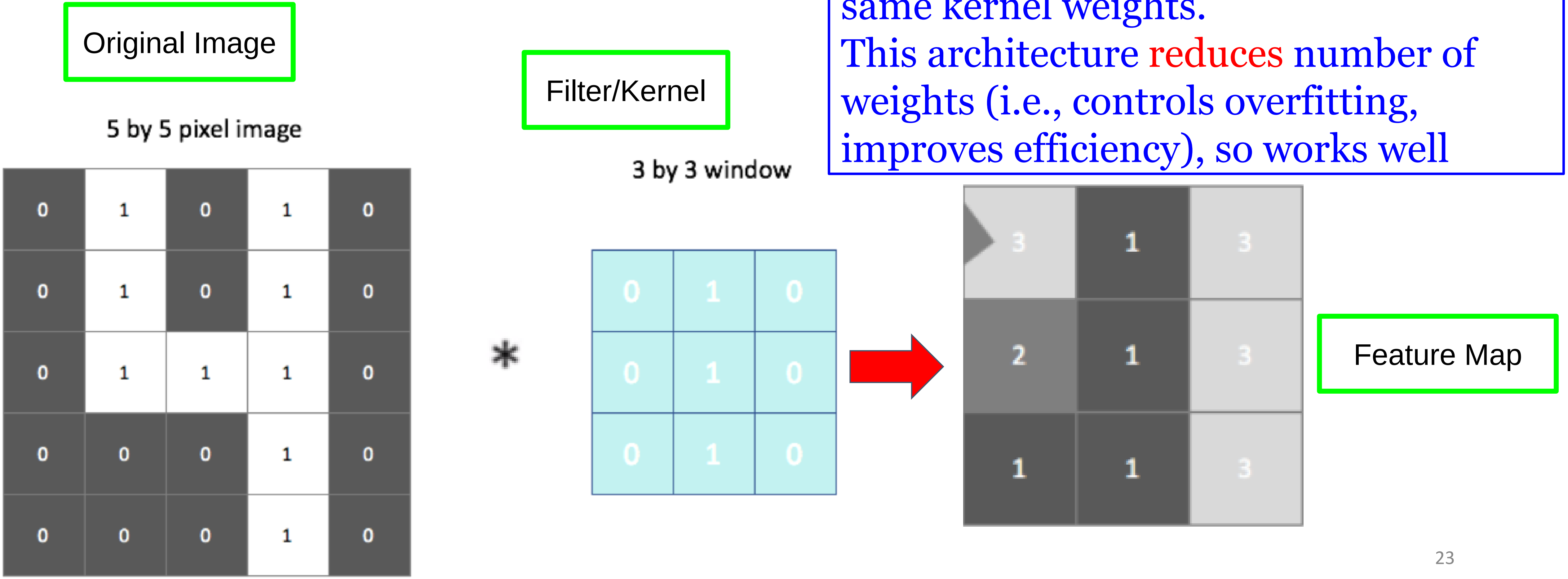
3	1	3
2	1	3
1	1	3

Feature Map

CNN-Filter/Kernel

- Finally:

Intuition:
Each cell (neuron) is connected only to a small chunk (subset) of the inputs from the original image.
These cells (neurons) use/share the same kernel weights.
This architecture reduces number of weights (i.e., controls overfitting, improves efficiency), so works well



Original Image

5 by 5 pixel image

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

Filter/Kernel

3 by 3 window

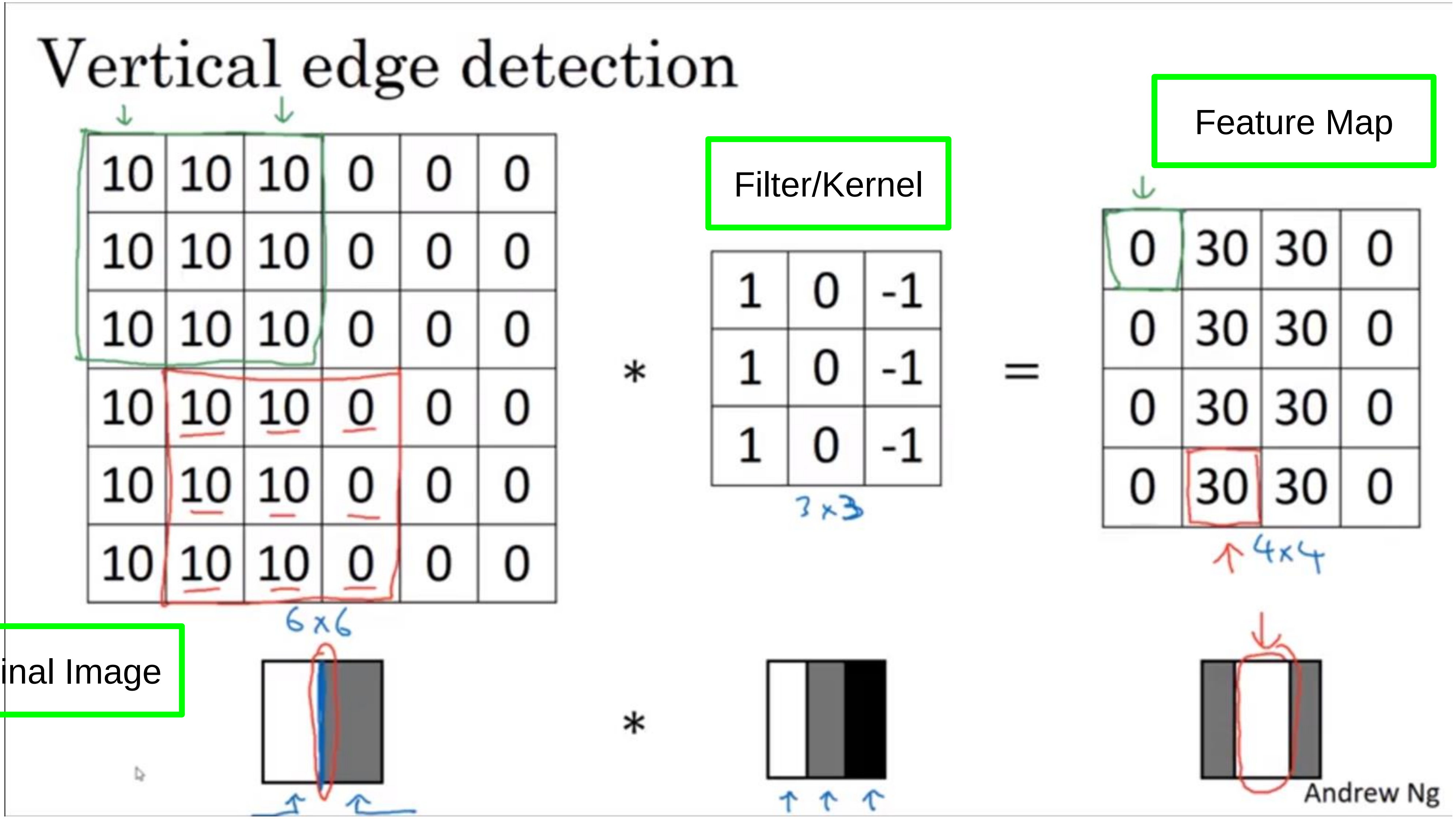
0	1	0
0	1	0
0	1	0

Feature Map

3	1	3
2	1	3
1	1	3

CNN-Filter/Kernel

- The results of using different filters/kernels (Suppose bias/intercept=0)



Vertical edge detection

$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix}_{6\times 6} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}_{3\times 3} = \begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}_{4\times 4}$$

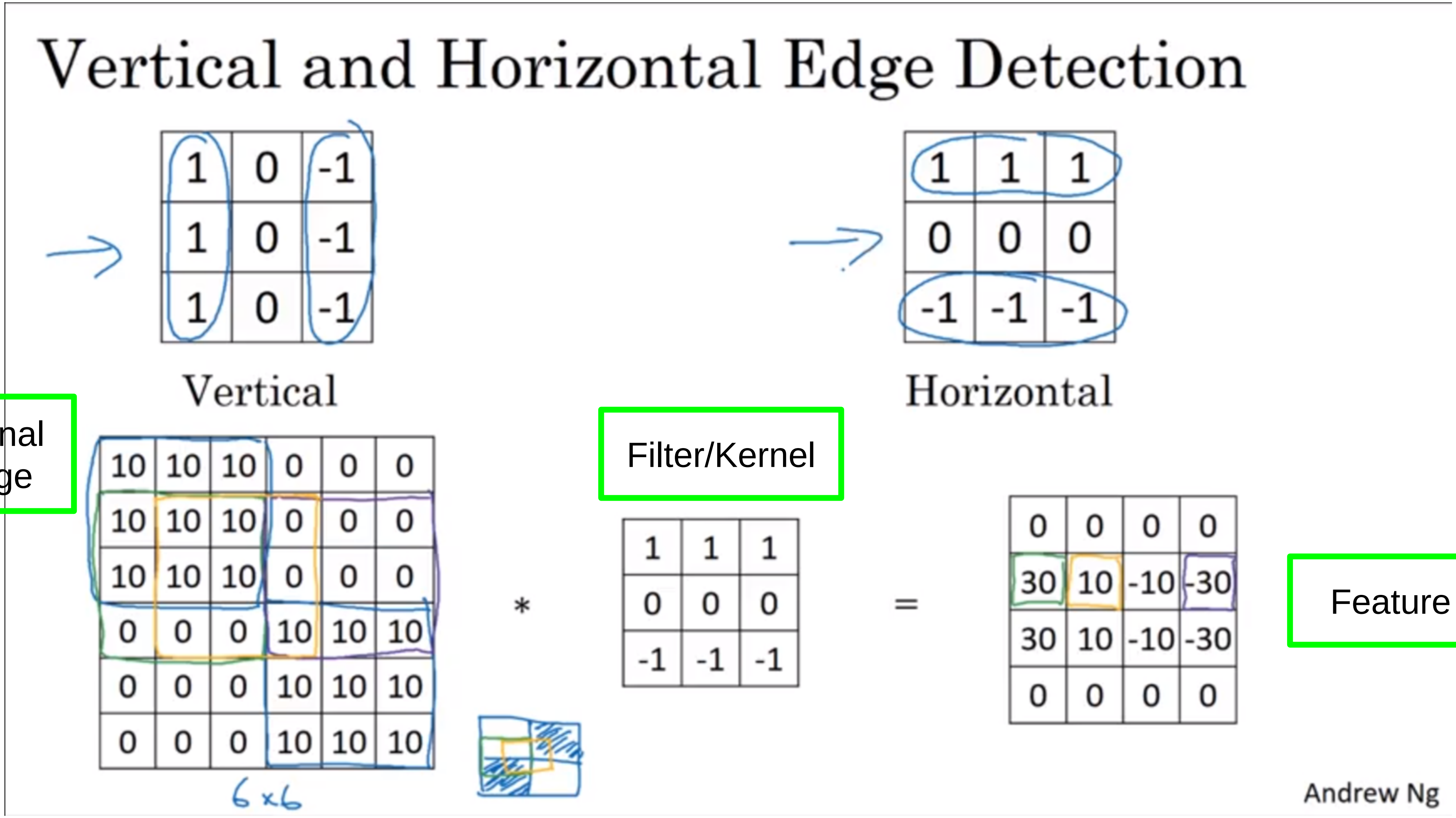
Original Image

Filter/Kernel

Feature Map

CNN-Filter/Kernel

- The results of using different filters/kernels (Suppose bias/intercept=0)



Vertical and Horizontal Edge Detection

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

Vertical

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

Horizontal

Original Image

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

6 x6

*

Filter/Kernel

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

=

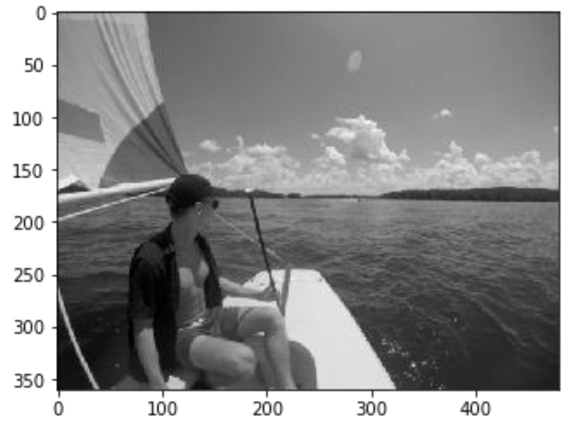
0	0	0	0
30	10	-10	-30
30	10	-10	-30
0	0	0	0

Feature Map

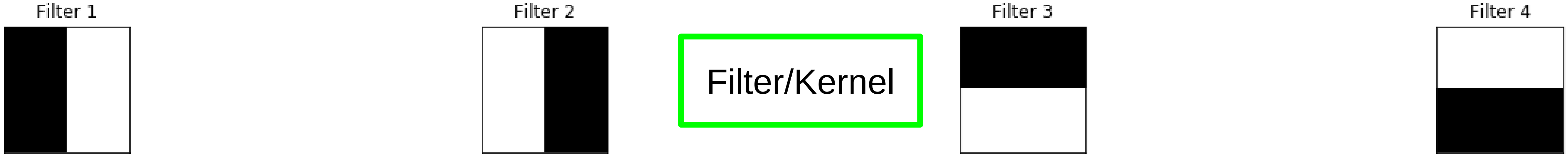
Andrew Ng

CNN-Filter/Kernel

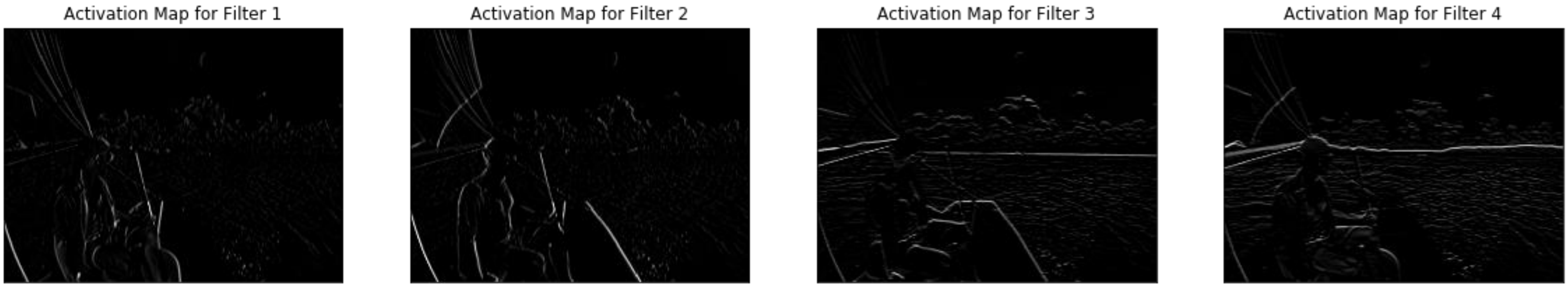
- The results of using different filters/kernels



Original Image



Filter/Kernel

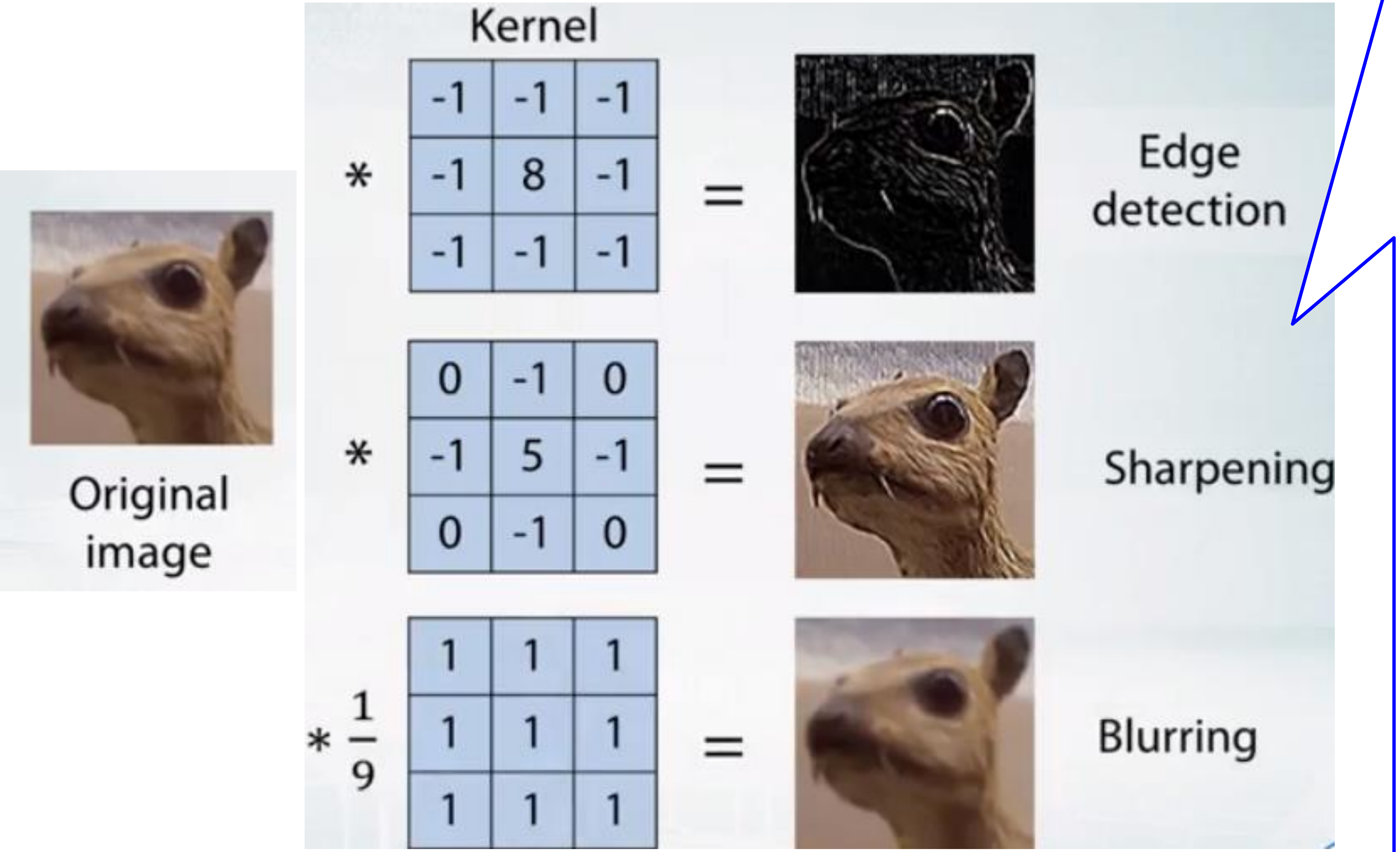

Activation Map for Filter 1
Activation Map for Filter 2
Activation Map for Filter 3
Activation Map for Filter 4

Feature/Activation Map

https://www.jeremyjordan.me/convolutional-neural-networks/

CNN-Filter/Kernel

- The results of using different filters/kernels



Use filters/kernels to extract features (e.g., edges, curves, shape, items, etc.) from the original image.

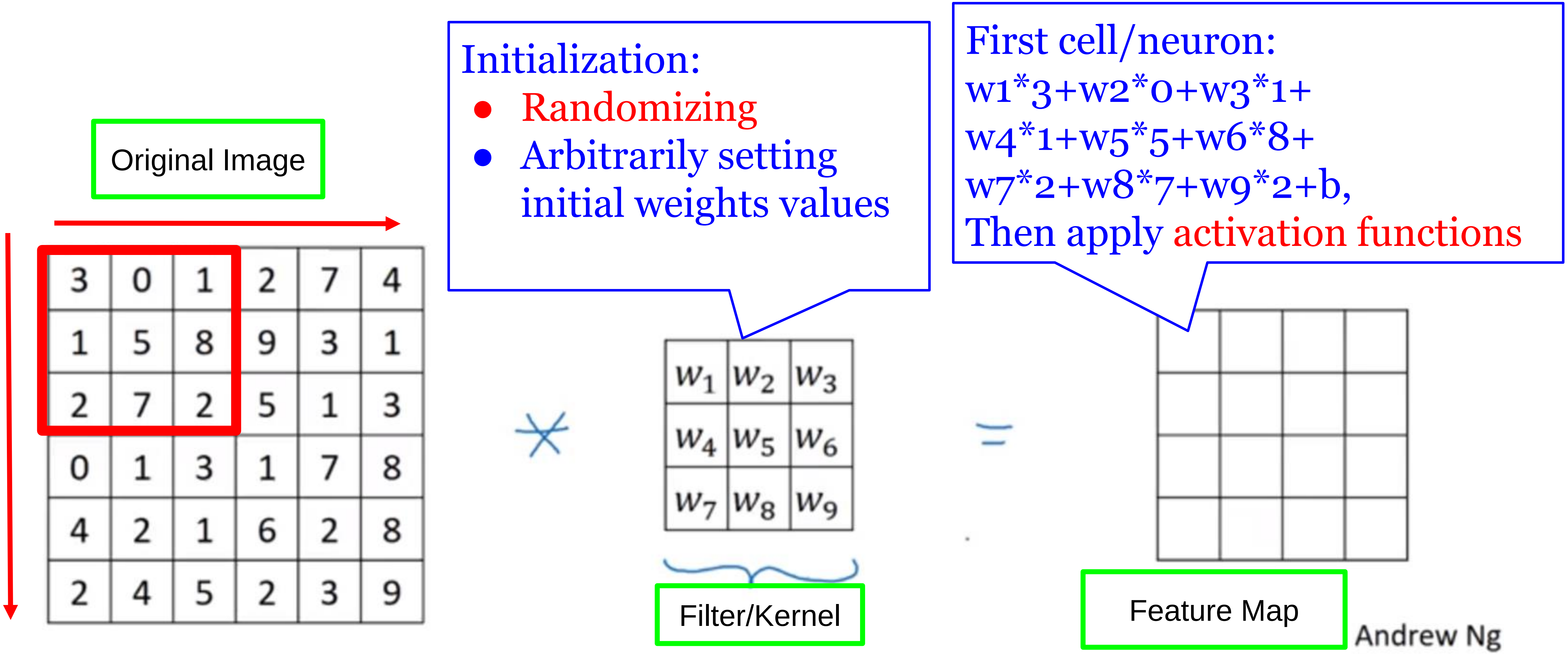
Use different filters/kernels to extract different image features

The more filters we have, the more image features get extracted and the better our network may become at recognizing patterns in unseen images.

How Do We Learn Filter/Kernel Weights in CNN?

CNN-Filter/Kernel

- FeedForward: Using Convolution layers to create feature maps
- In practice, a CNN **learns** the values of these filters weights on its own during **“backpropagation”** process. Let computer learn filters/kernels weights.



Original Image

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Initialization:

- Randomizing
- Arbitrarily setting initial weights values

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

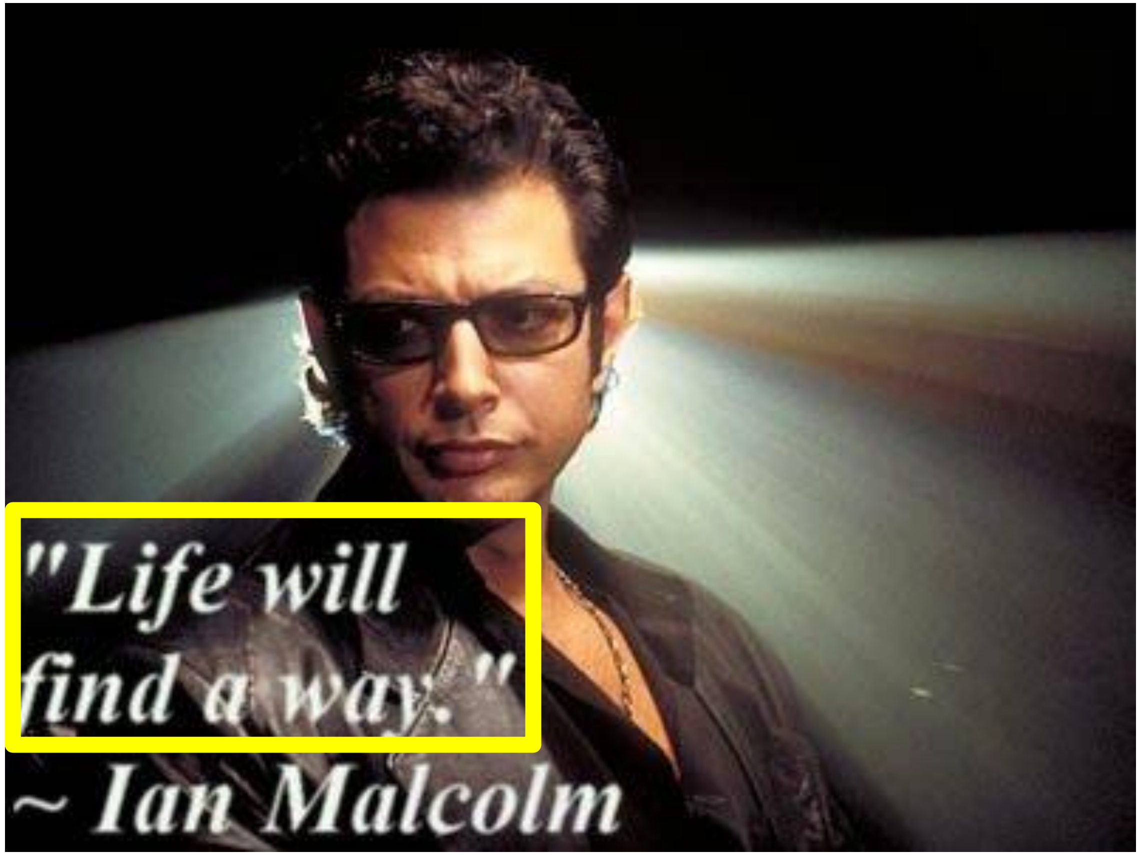
Filter/Kernel

First cell/neuron:
w1*3+w2*0+w3*1+
w4*1+w5*5+w6*8+
w7*2+w8*7+w9*2+b,
Then apply activation functions

Feature Map

CNN-Filter/Kernel

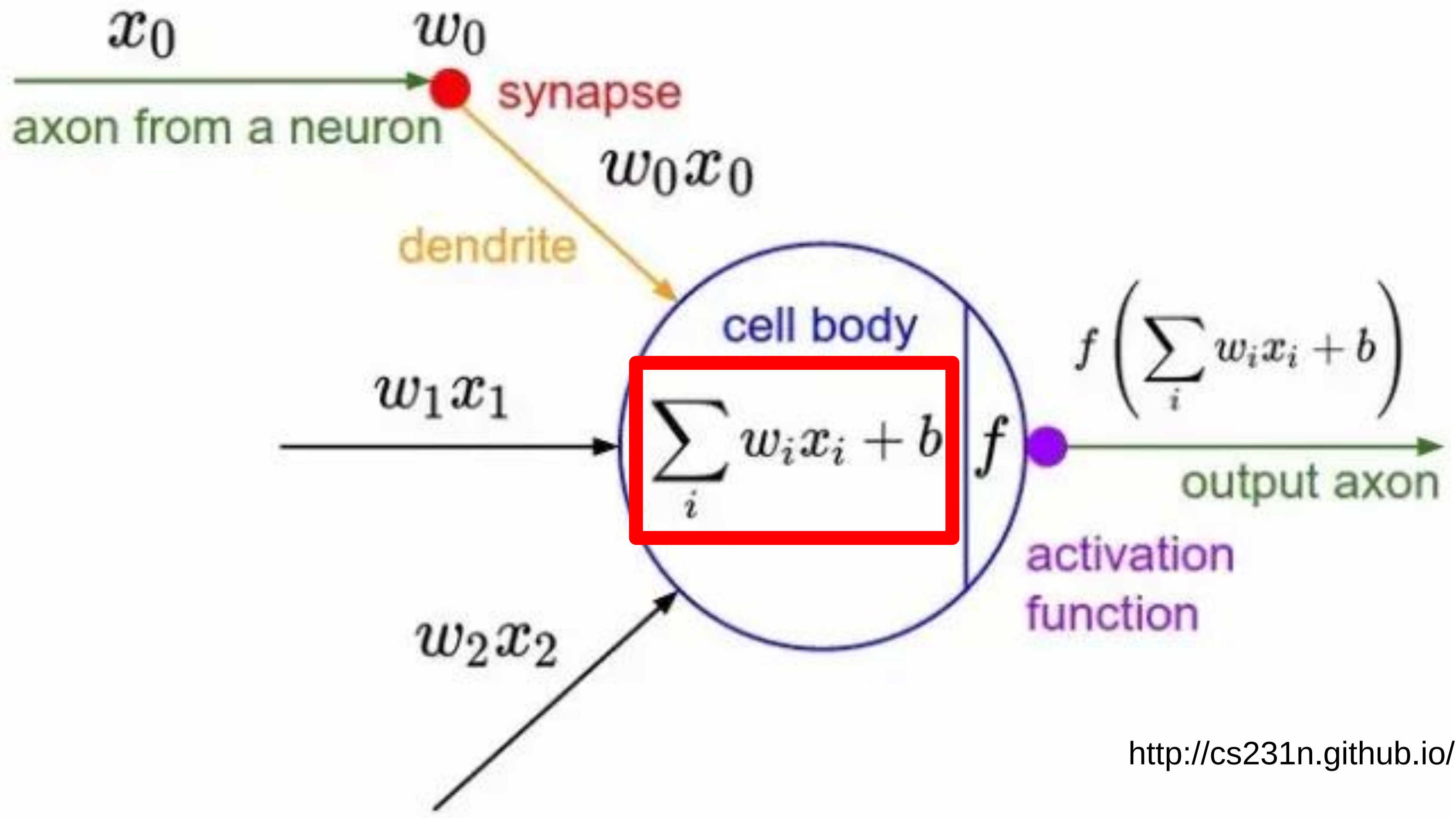
- **Black-Box: We don't know what filters/kernels CNN will learn, but CNN will learn the best filters/kernels from training data**
- In existing CNN architectures, usually we need to learn **millions or billions of weights/parameters**, we don't know what is going on in the training process



Where are Neurons and Hidden Layers in CNN?

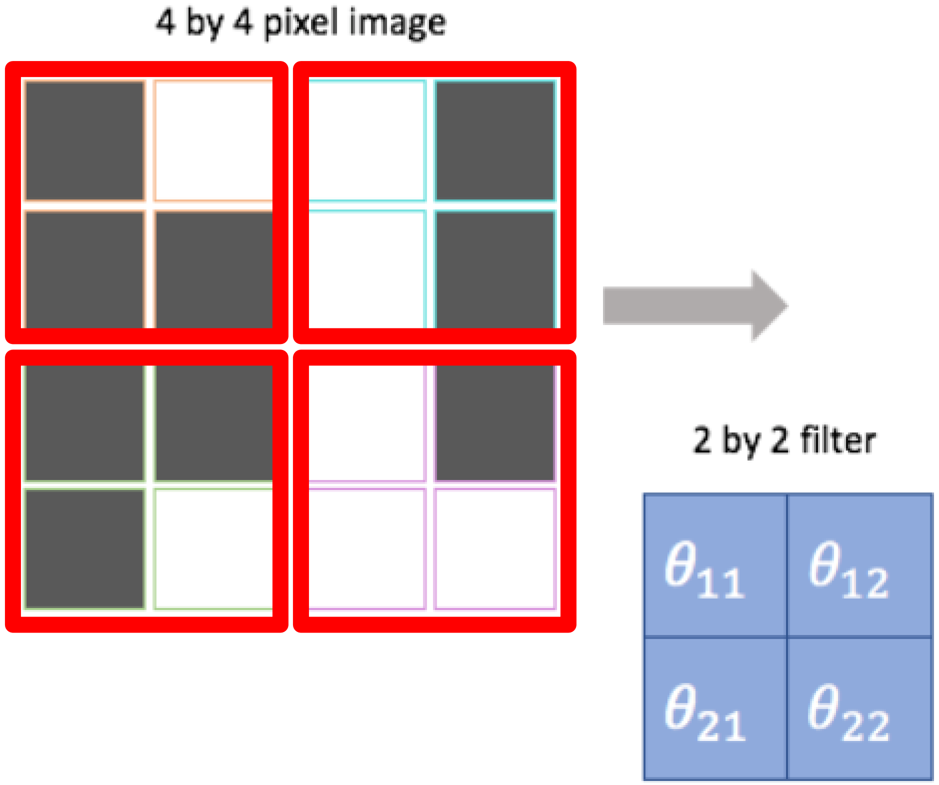
Where are Neurons and Hidden Layers in CNN?

A typical neuron in the hidden layer of a normal neural network

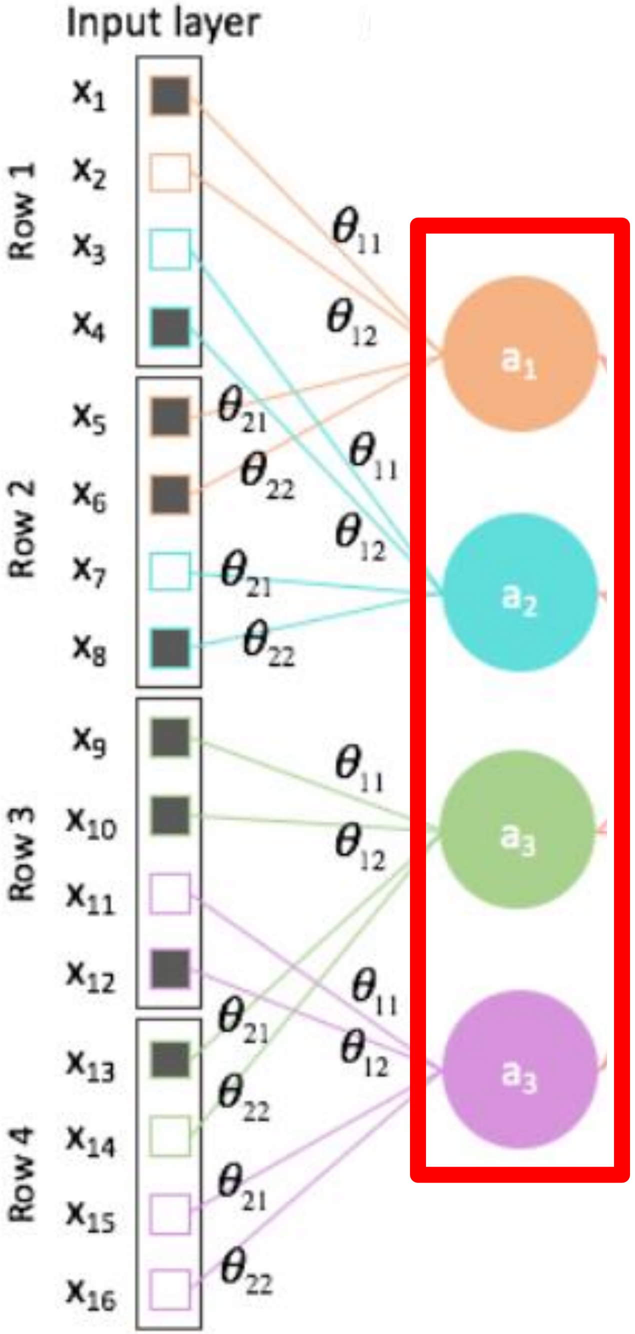


http://cs231n.github.io/convolutional-networks/

CNN-Neurons and Hidden Layers

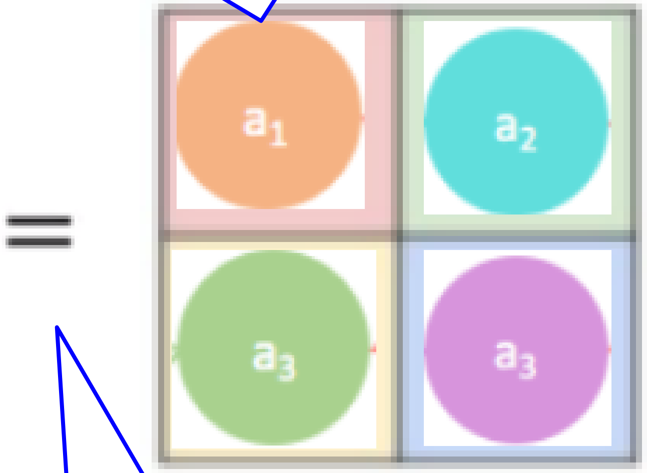


Filter/Kernel:
Suppose Slide Stride=2



First cell/neuron a1:
θ11*x1+θ12*x2+
θ21*x5+θ22*x6+b,

Then apply
activation functions



Feature Map

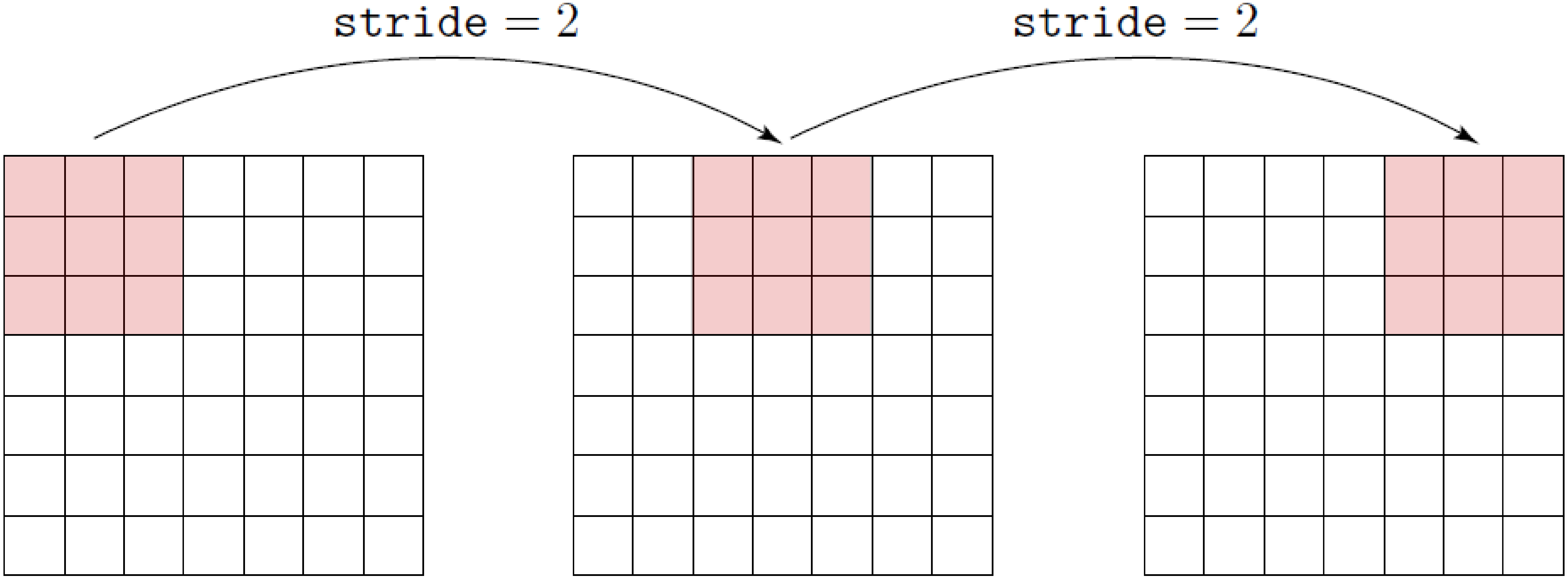
Arrange neurons
into feature map

https://www.jeremyjordan.me/convolutional-neural-networks/

CNN Building Blocks
Filter/Kernel Stride

CNN-Activation Function

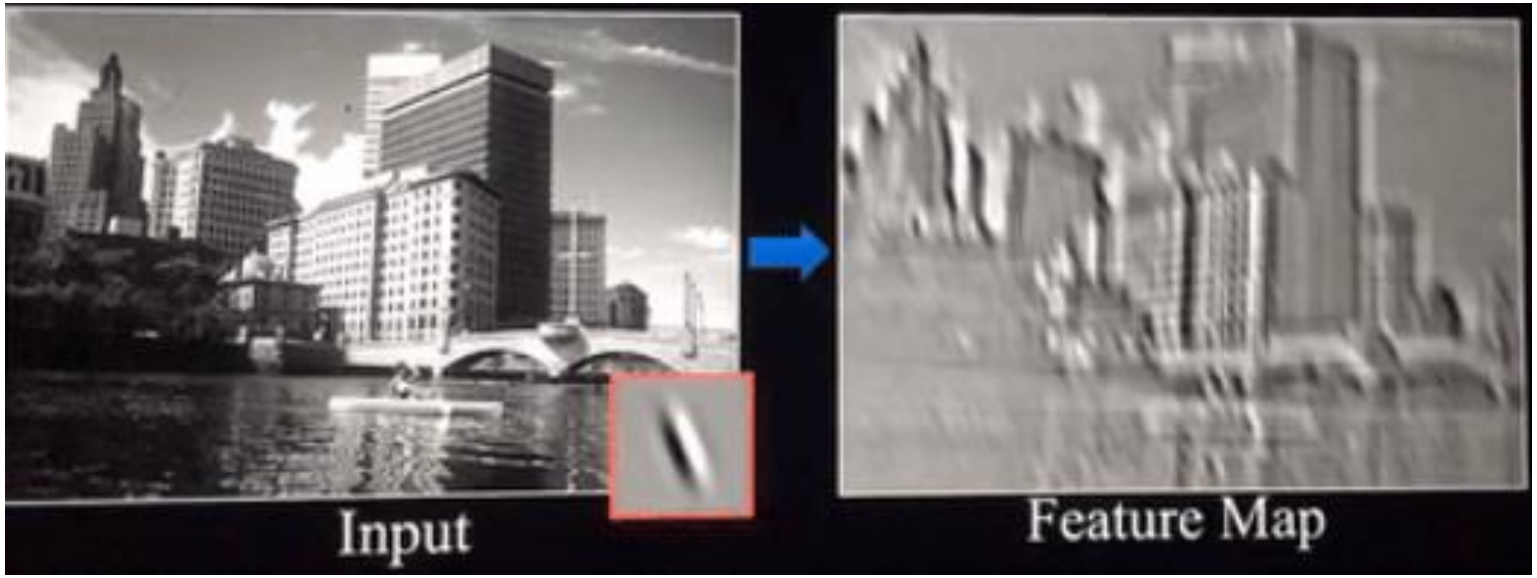
- You can choose other slide stride in moving filters/kernels (e.g., stride=1, 2, ...)

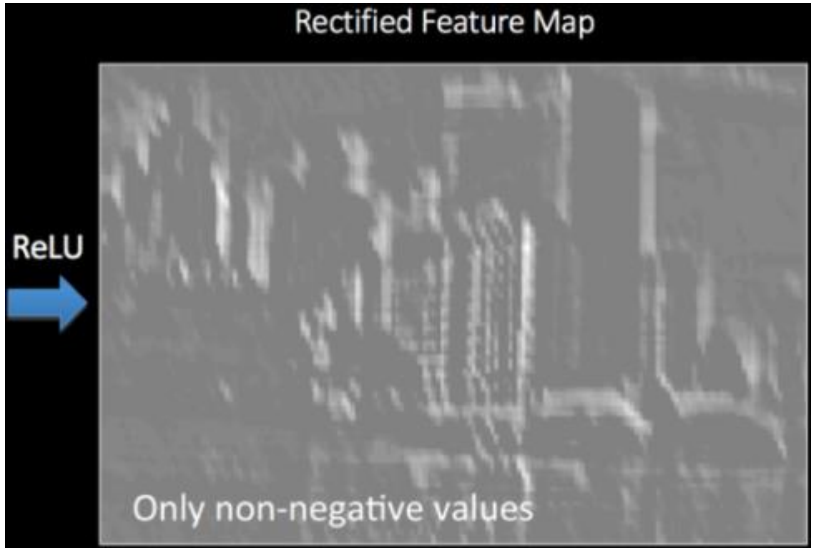


CNN Building Blocks
Activation Functions

CNN-Activation Function

- Introduce non-linearities of features; Rectify negative pixel values
- Usually we use ReLU: f(x)=max(0,x)



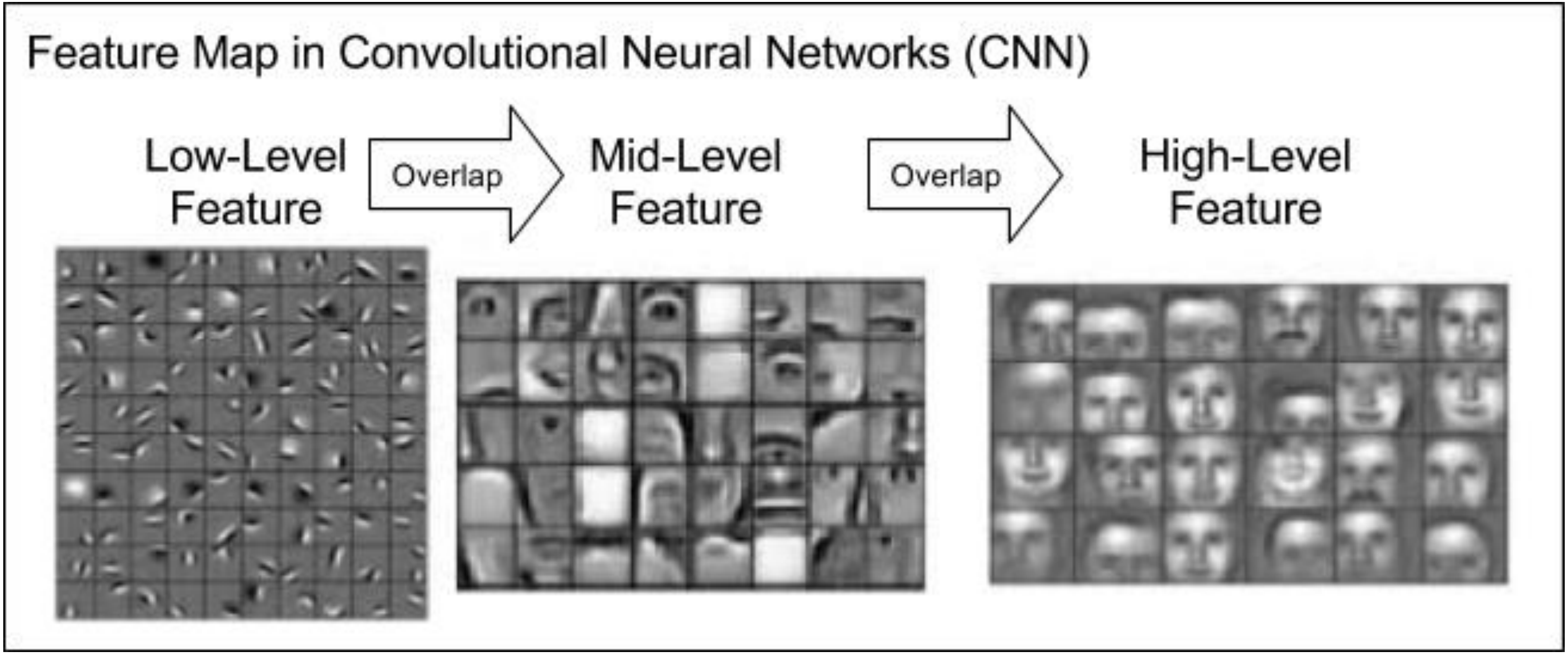


https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Visualize What Is Convolution Layer (Filters/Kernels) Doing?

CNN-An Example to Visualization

- Example: Face Recognition

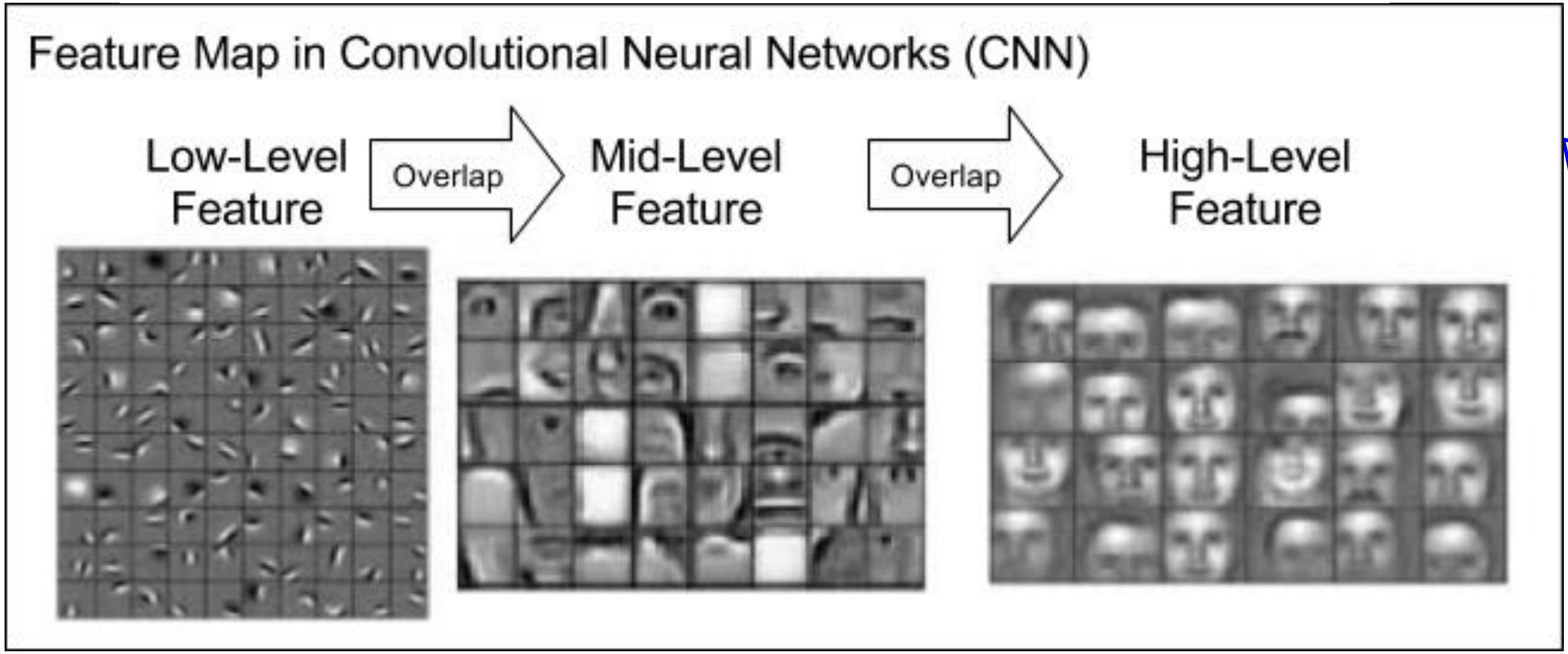


https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

CNN-An Example to Visualization

- Example: Face Recognition
- Understand how abstract features are extracted, processed and combined into high-level features to do classification

Edges -> Shape -> object (face shape)



https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

In Image Classification:

Convolution layers (ConvNet) may learn to detect low-level edges from raw pixels in the first layer, then use the low-level edges to detect simple shapes in the second layer, and then use these shapes to detect higher-level objects in higher layers.

CNN Building Blocks
Padding

CNN-Padding

- Zero-Padding: Create a feature map which is the **same size** of the original image

Original Image

$\text{pad} = 0$

$\text{pad} = 1$

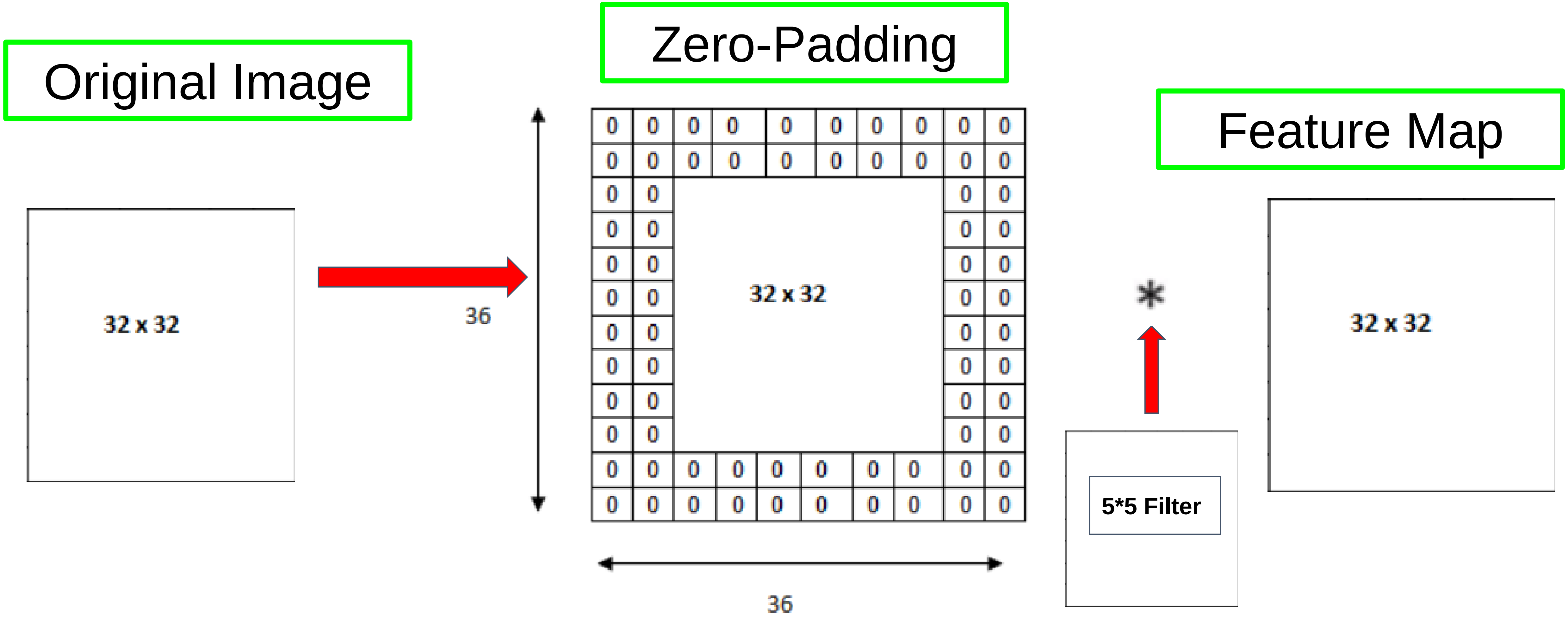
0	0	0	0	0
0				0
0				0
0				0
0	0	0	0	0

$\text{pad} = 2$

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0				0	0
0	0				0	0
0	0				0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

CNN-Padding

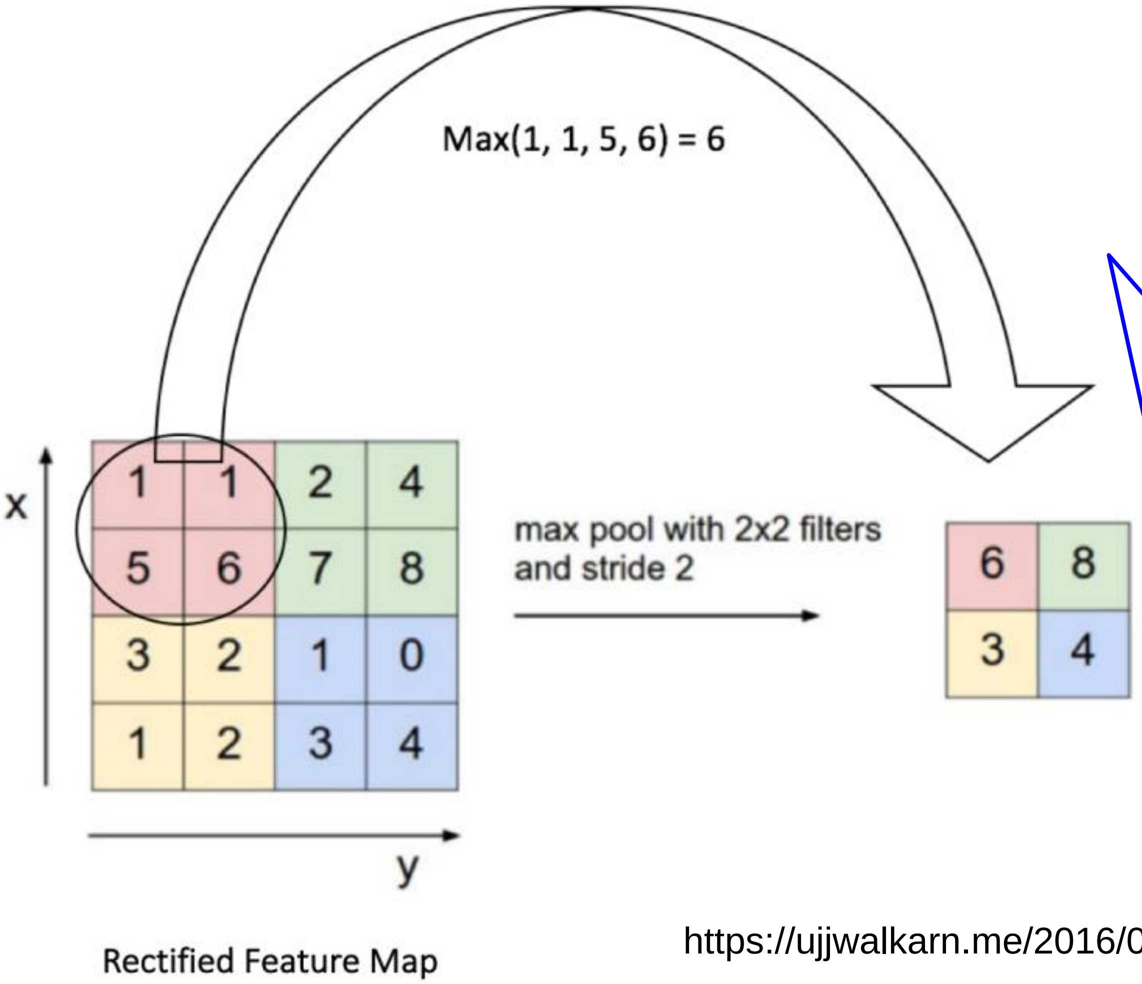
- Zero-Padding: Create a feature map which is the same size of the original image



CNN Building Blocks
Pooling

CNN-Pooling

- Reduce the amount of parameters and computation
- Example:
- Maxpooling:



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

6	8
3	4

Pooling (a.k.a, subsampling or downsampling):

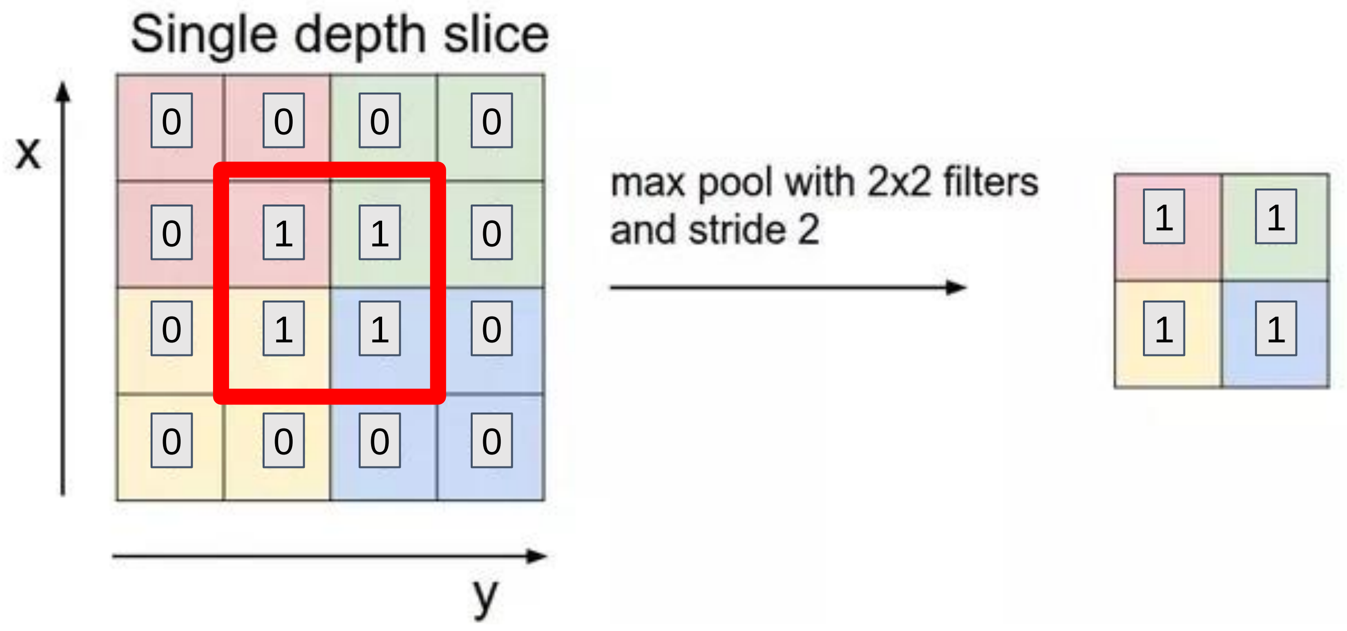
Reduces the dimensionality of each feature map but retains the most important information.

Extracting key features/objects from background color

Pooling can be of different types: Max, Average, Min, Sum etc.

https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

CNN-Pooling



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

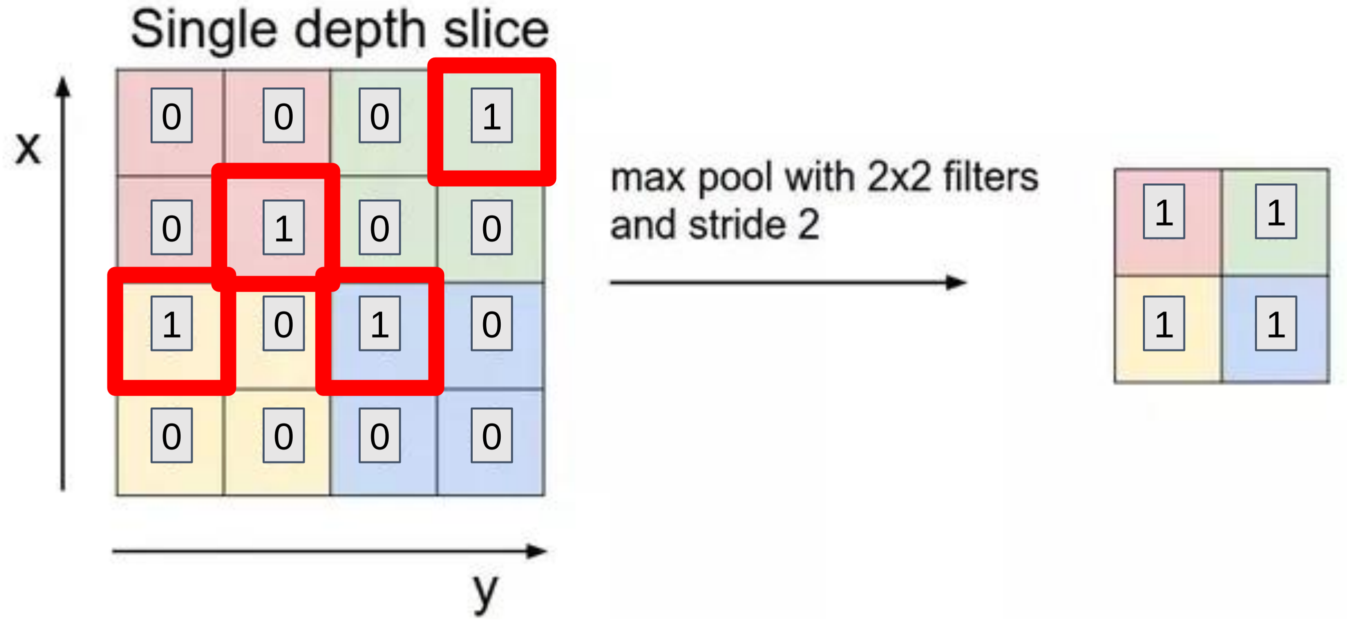
1	1
1	1

Extracting key features/objects from background color

Invariant to small transformations or distortions in original image

http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

CNN-Pooling



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

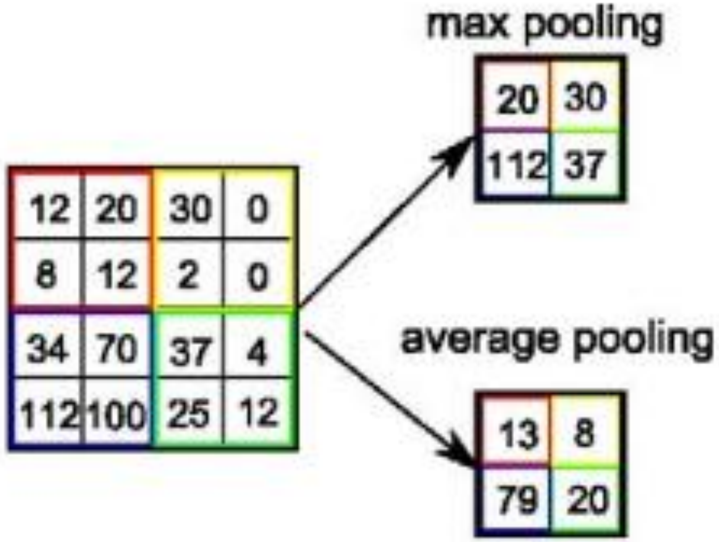
1	1
1	1

Extracting key features/objects from background color

Invariant to small transformations or distortions in original image

http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

CNN-Pooling



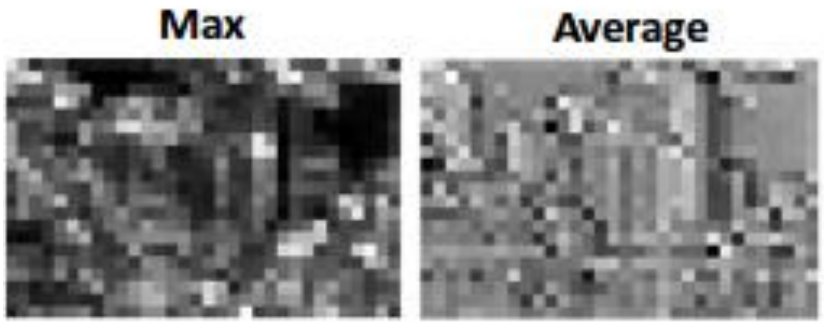
12	20	30	0
8	12	2	0
34	70	37	4
112	100	25	12

max pooling

20	30
112	37

average pooling

13	8
79	20

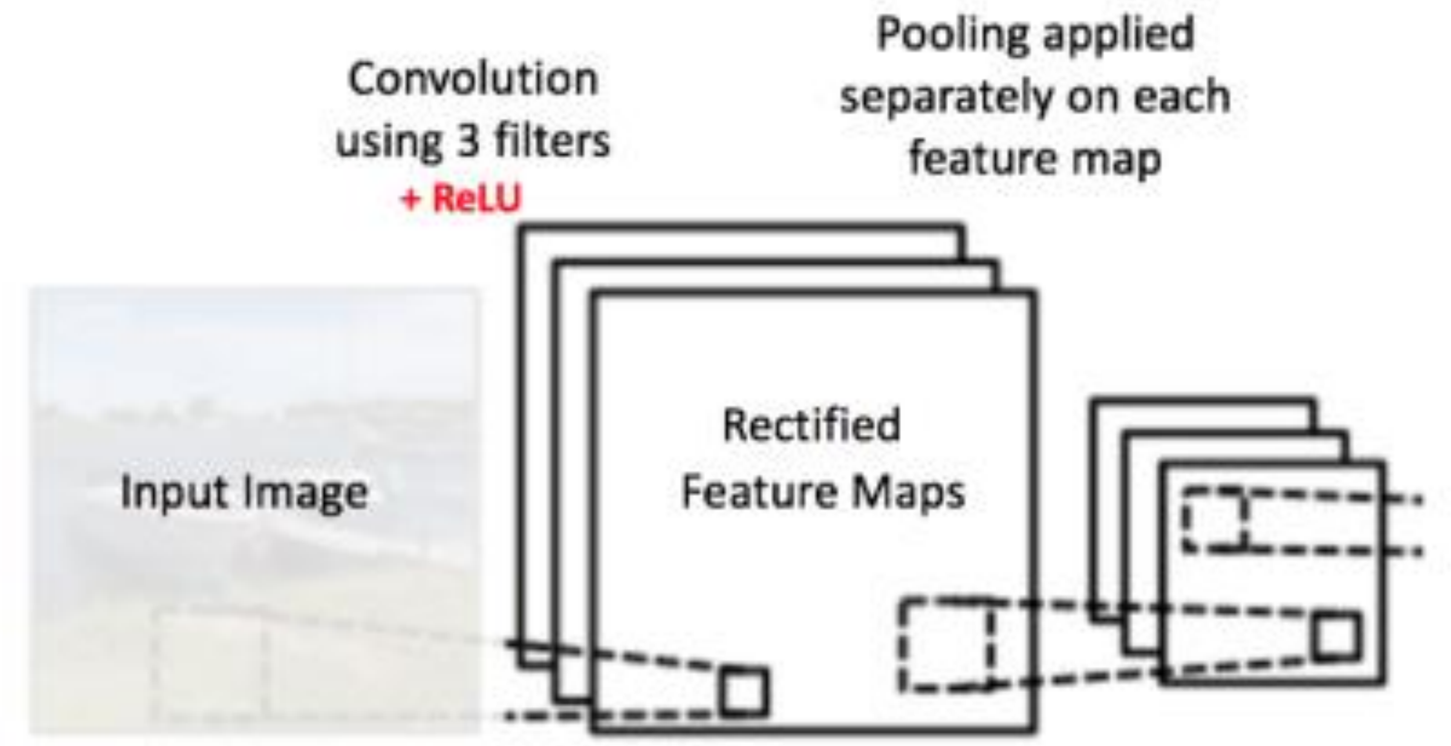


https://www.quora.com/What-is-the-benefit-of-using-average-pooling-rather-than-max-pooling

Max pooling extracts the most important/salient features (e.g., edges or textures) from the background color.

Whereas, Average pooling extracts features so smoothly.

CNN-Pooling



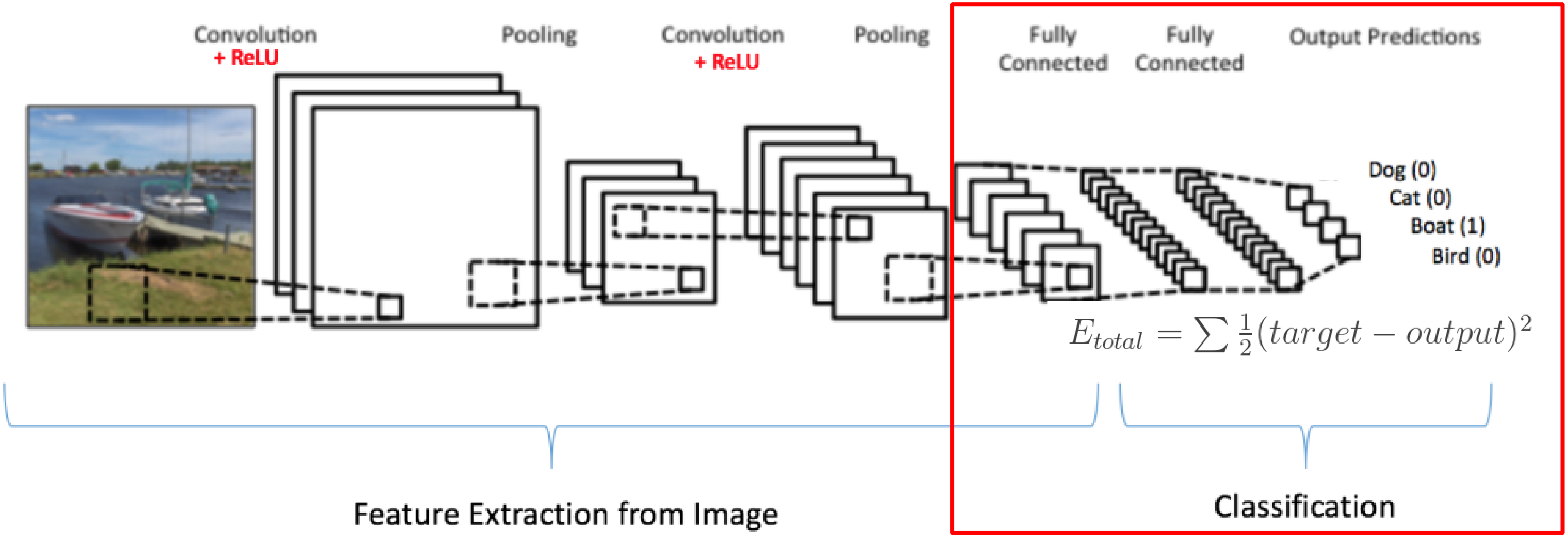
Pooling operates on each feature map independently.

https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

CNN Building Blocks
Fully Connected Layer (FC)

CNN-FC Layer

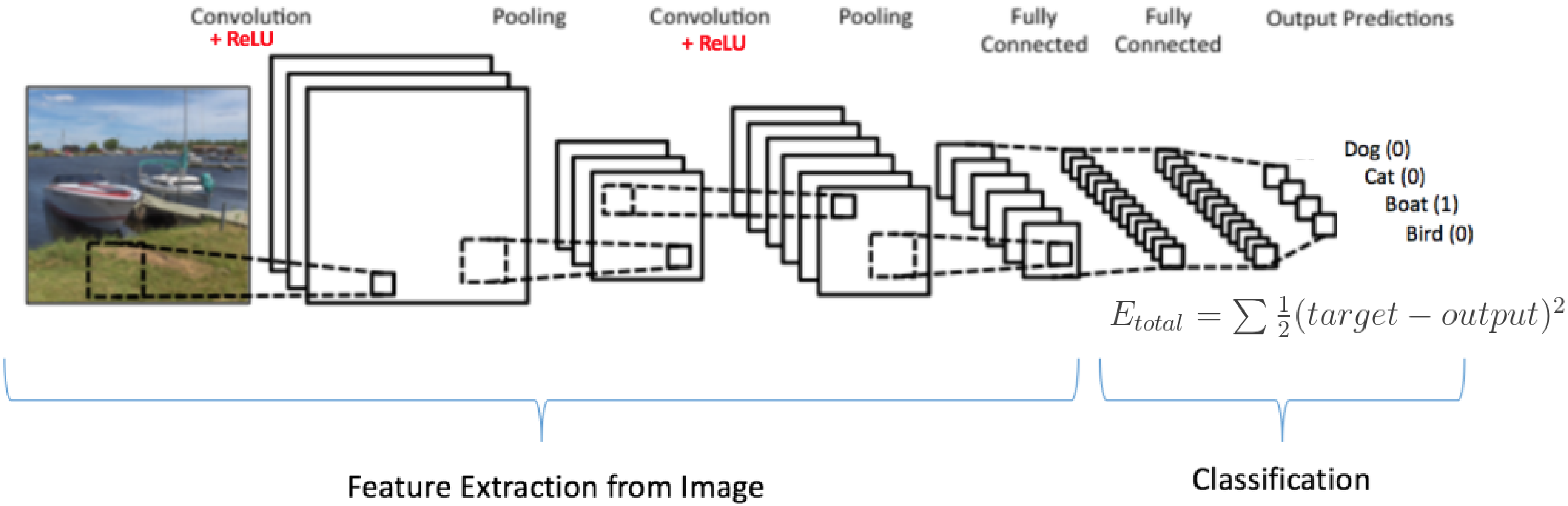
- Can view as the final learning phase, which maps extracted visual features to desired outputs



https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

How to Adjust Filters/Kernels Weights? Backpropagation

CNN-Training Process



$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

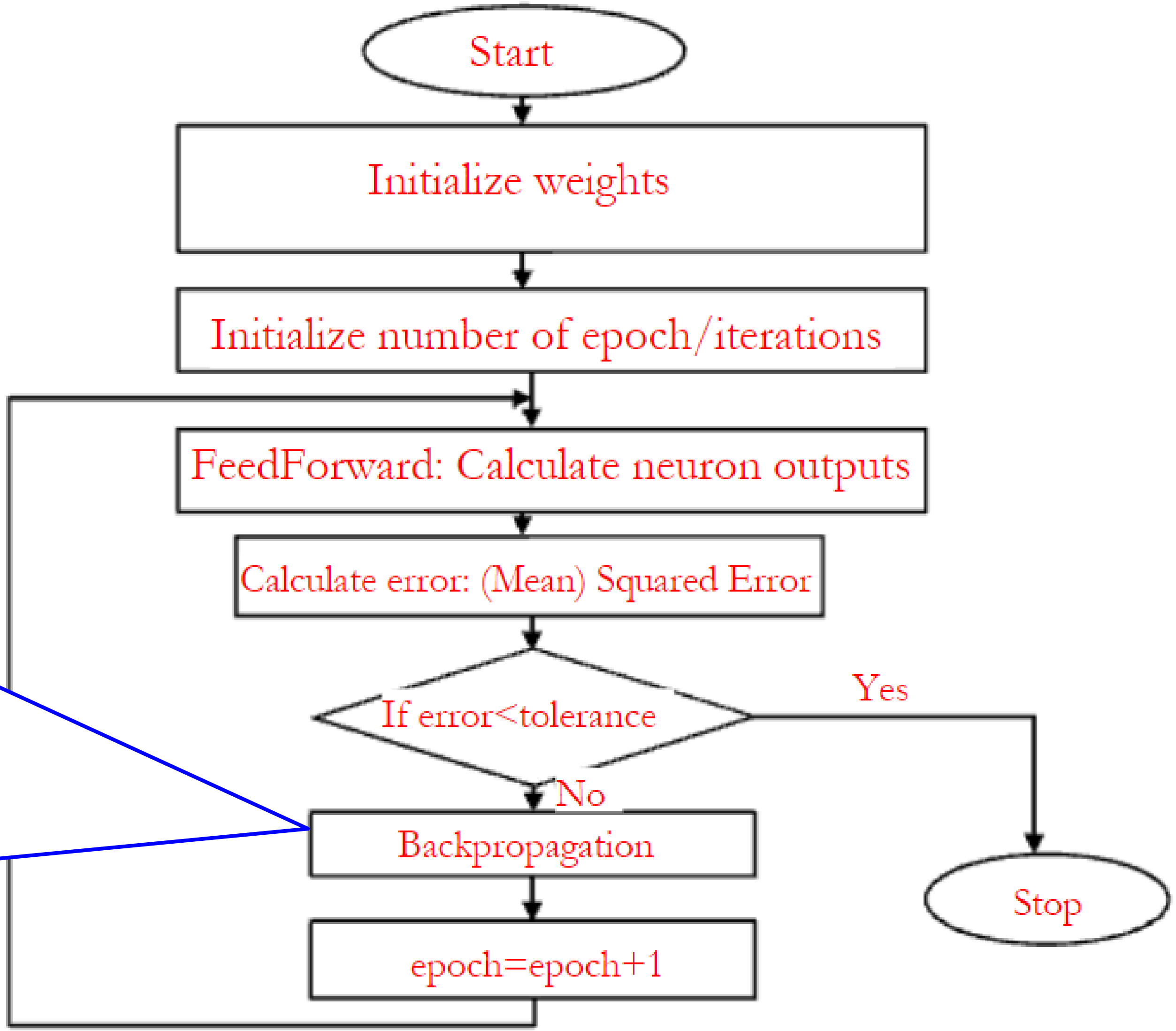
CNN-Training Process

Input -> Conv -> ReLU -> Conv -> ReLU -> Pool -> ReLU -> Conv -> ReLU -> Pool ->Fully Connected

Backpropagation:

Use Gradient Descent to adjust weights in convolution layer filters/kernels.

Hyperparameters like number of filters, filter sizes, architecture of the network etc. are pre-determined and do not change during training process.

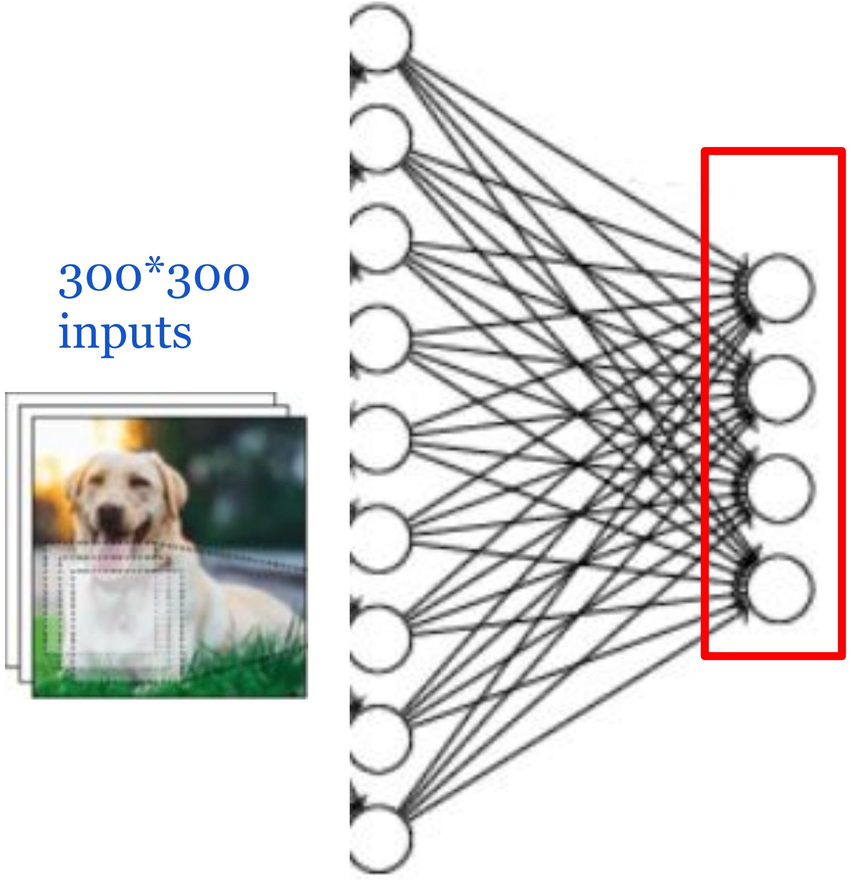

Start
Initialize weights
Initialize number of epoch/iterations
FeedForward: Calculate neuron outputs
Calculate error: (Mean) Squared Error
If error<tolerance
Yes
No
Backpropagation
epoch=epoch+1
Stop

CNN: Benefits

The Benefits of CNN

Fully Connected Normal NN

300*300 inputs

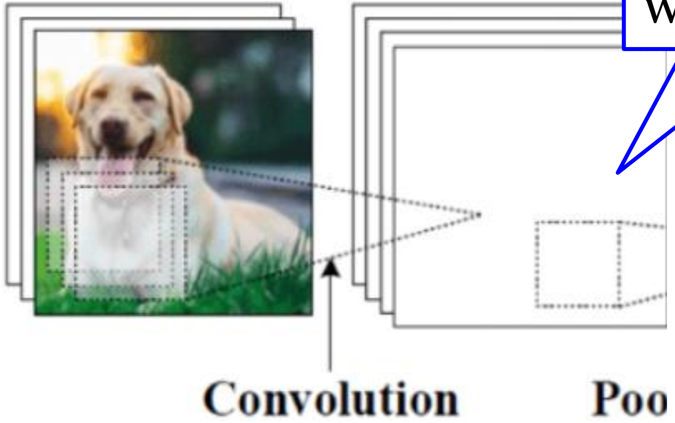


300*300*4+1
About 360,001
weights

Suppose you have four neurons in the hidden layer

CNN

300*300 inputs



(5*5+1)*4
Only 104
weights

Suppose you have four 5*5 kernels in the convolution layer

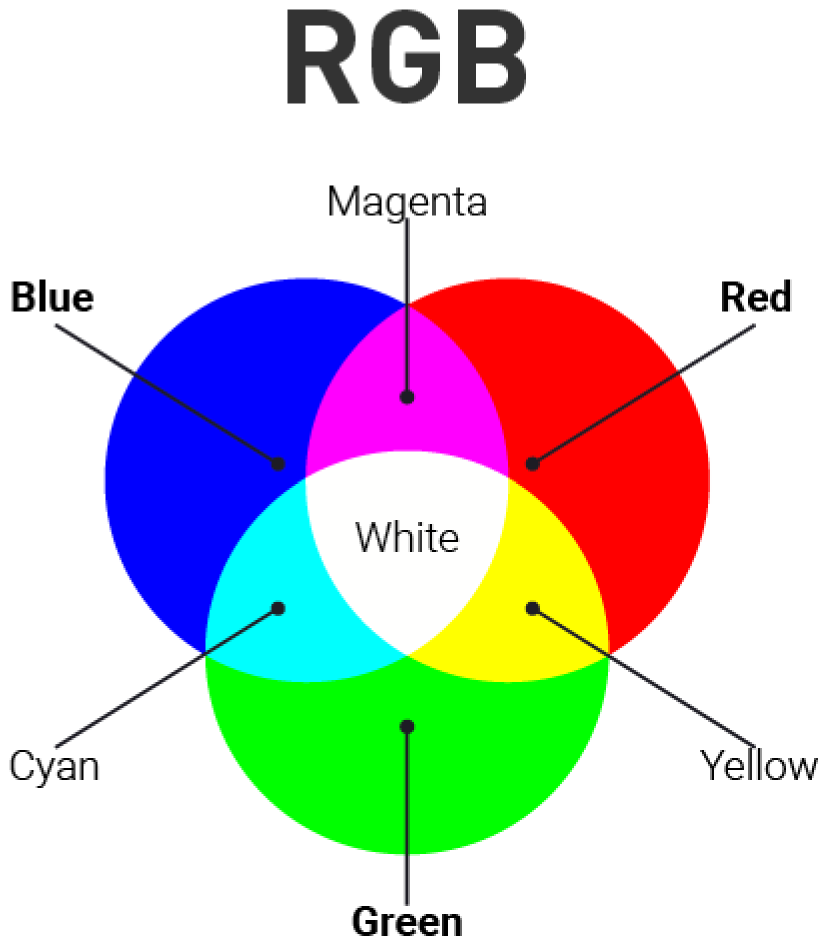
What we have introduced just now is grayscale image case

What about colourful image?

CNN-Colourful Image

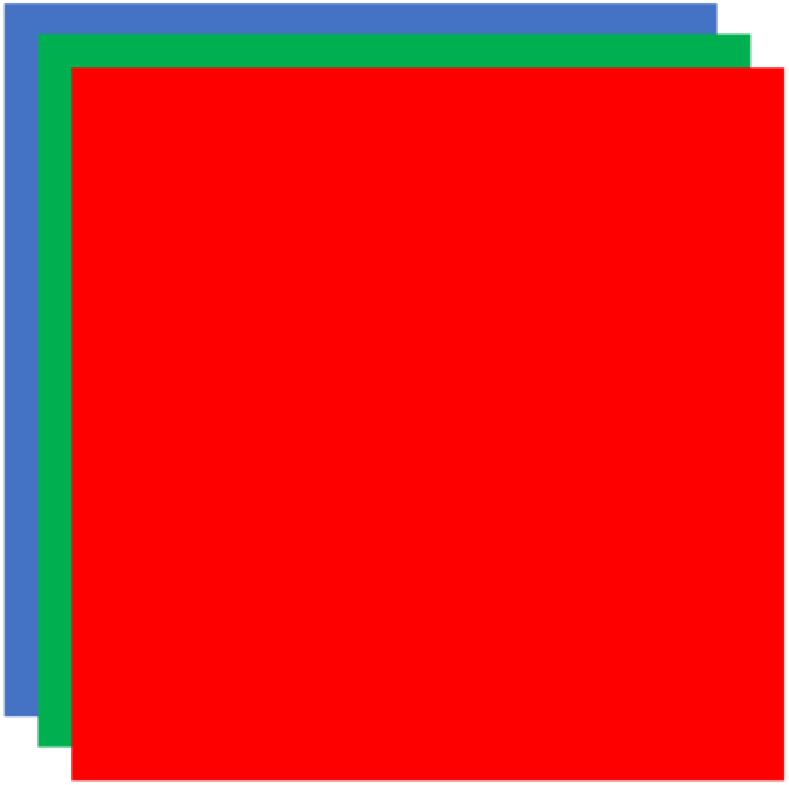
- RGB (Red-Green-Blue) Channels
- Any color is composed of R, G and B
- Any colourful pixel in the image is composed of pixels from R, G and B channels

Reminder:
The number of channels in our filter/kernel must match the number of channels in your input.
It means the depth of your filter/kernel should also be 3



https://www.jeremyjordan.me/convolutional-neural-networks/

CNN-Colourful Image



Original Image:
Depth=3

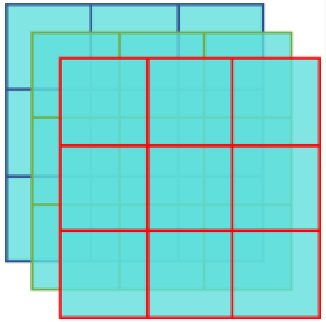
$$\text{cell} = (\sum_{R}\sum_{G}\sum_{B} \text{weight} \times \text{pixel}) + b$$

$$\text{cell} \leftarrow \text{ActivationFunction}(\text{cell})$$

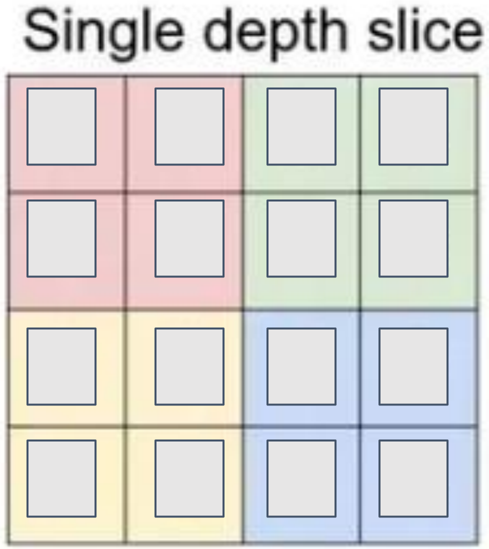
$$\begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \\ \theta_{31} & \theta_{32} & \theta_{33} \end{bmatrix}$$

$$\begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \\ \theta_{31} & \theta_{32} & \theta_{33} \end{bmatrix}$$

$$\begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \\ \theta_{31} & \theta_{32} & \theta_{33} \end{bmatrix}$$

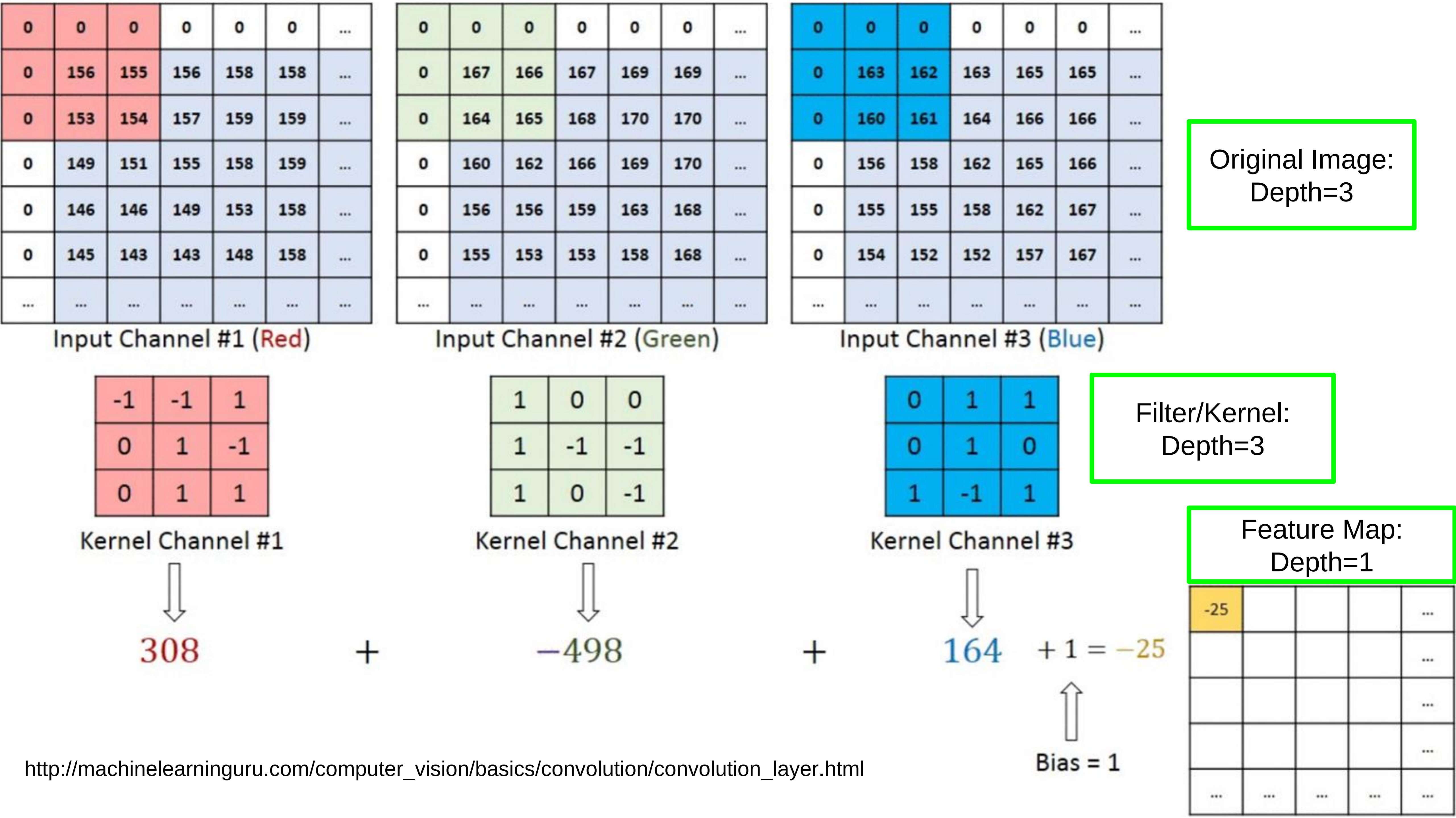


Filter/Kernel:
Depth=3



Feature Map:
Depth=1

RGB Image Example: Convolution Layer



Original Image:
Depth=3

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...	...	...	...	...	...	...

Input Channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...	...	...	...	...	...	...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...	...	...	...	...	...	...

Input Channel #3 (Blue)

Filter/Kernel:
Depth=3

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

Kernel Channel #1

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

Kernel Channel #2

0	1	1
0	1	0
1	-1	1

Kernel Channel #3

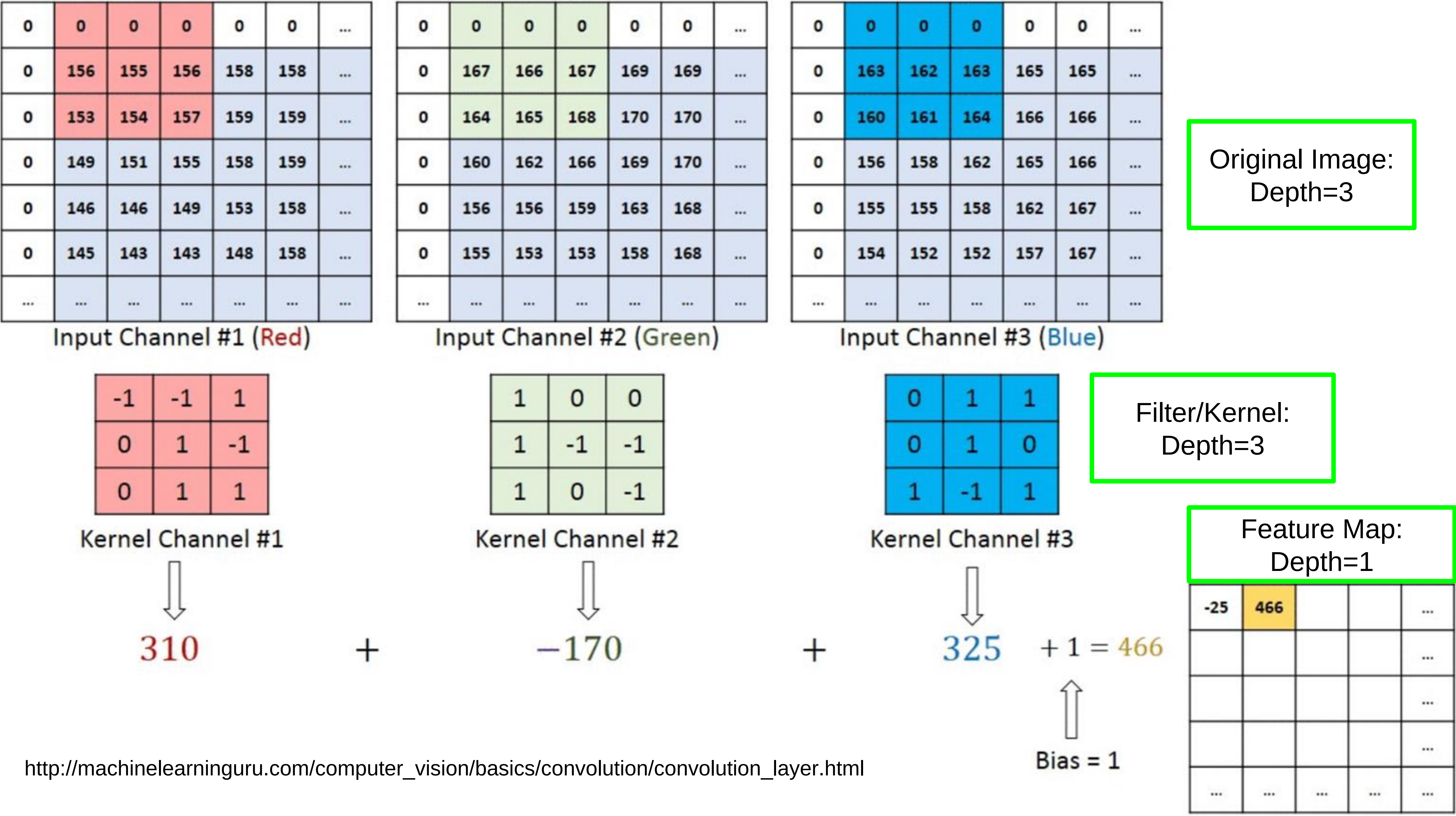
$$308 \quad + \quad -498 \quad + \quad 164 \quad + 1 = -25$$

Bias = 1

Feature Map:
Depth=1

-25				...
				...
				...
				...
...	...	...	...	...

http://machinelearninguru.com/computer_vision/basics/convolution/convolution_layer.html



0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...	...	...	...	...	...	...

Input Channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...	...	...	...	...	...	...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...	...	...	...	...	...	...

Input Channel #3 (Blue)

Original Image:
Depth=3

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

Kernel Channel #1

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

Kernel Channel #2

0	1	1
0	1	0
1	-1	1

Kernel Channel #3

Filter/Kernel:
Depth=3

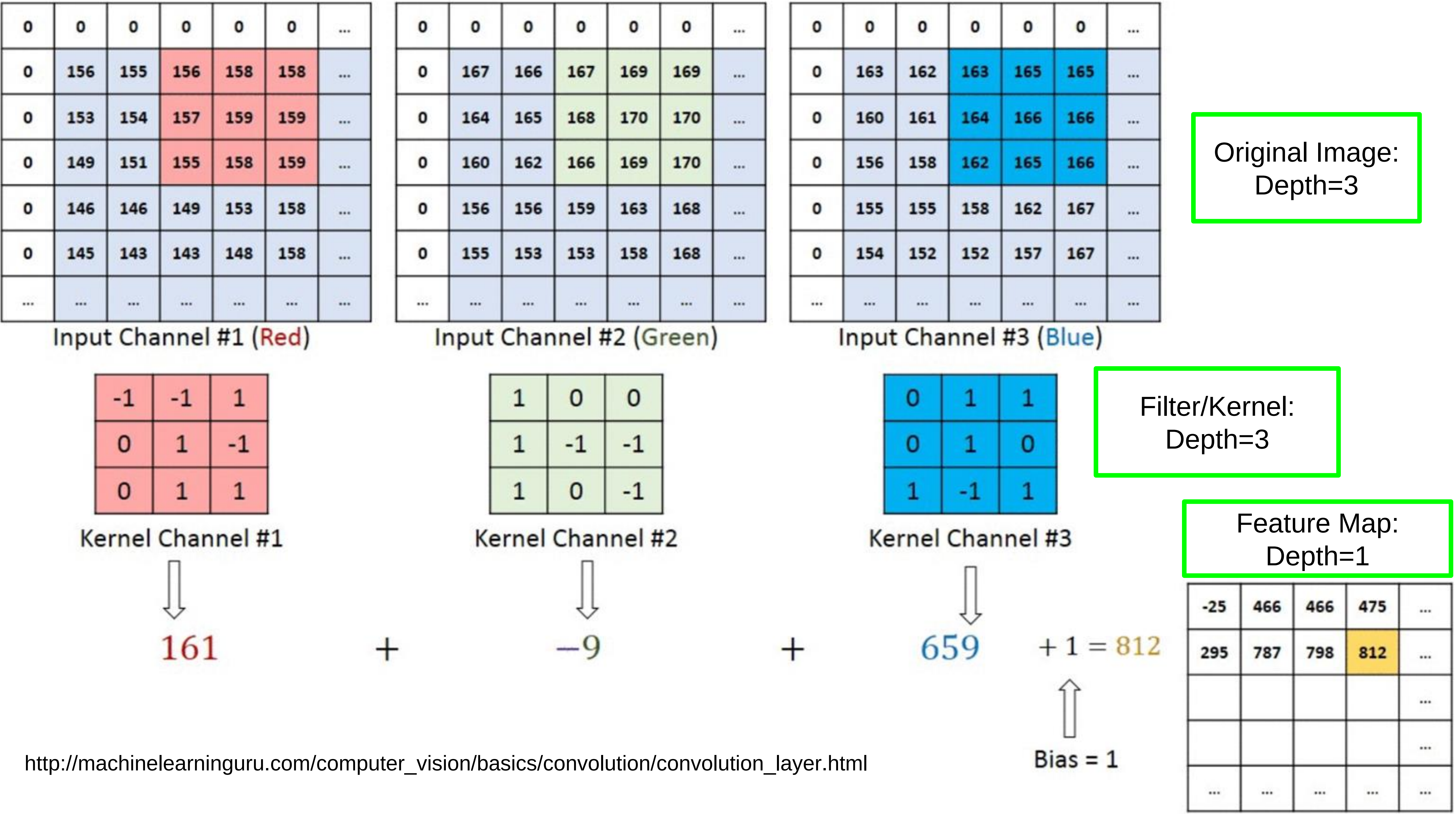
$$310 \quad + \quad -170 \quad + \quad 325 \quad +1 = 466$$

Bias = 1

Feature Map:
Depth=1

-25	466			...
				...
				...
				...
...	...	...	...	...

http://machinelearninguru.com/computer_vision/basics/convolution/convolution_layer.html



0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...	...	...	...	...	...	...

Input Channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...	...	...	...	...	...	...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...	...	...	...	...	...	...

Input Channel #3 (Blue)

Original Image:
Depth=3

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

Kernel Channel #1

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

Kernel Channel #2

0	1	1
0	1	0
1	-1	1

Kernel Channel #3

Filter/Kernel:
Depth=3

$$161 + -9 + 659 + 1 = 812$$

Bias = 1

Feature Map:
Depth=1

-25	466	466	475	...
295	787	798	812	...
				...
				...
...	...	...	...	...

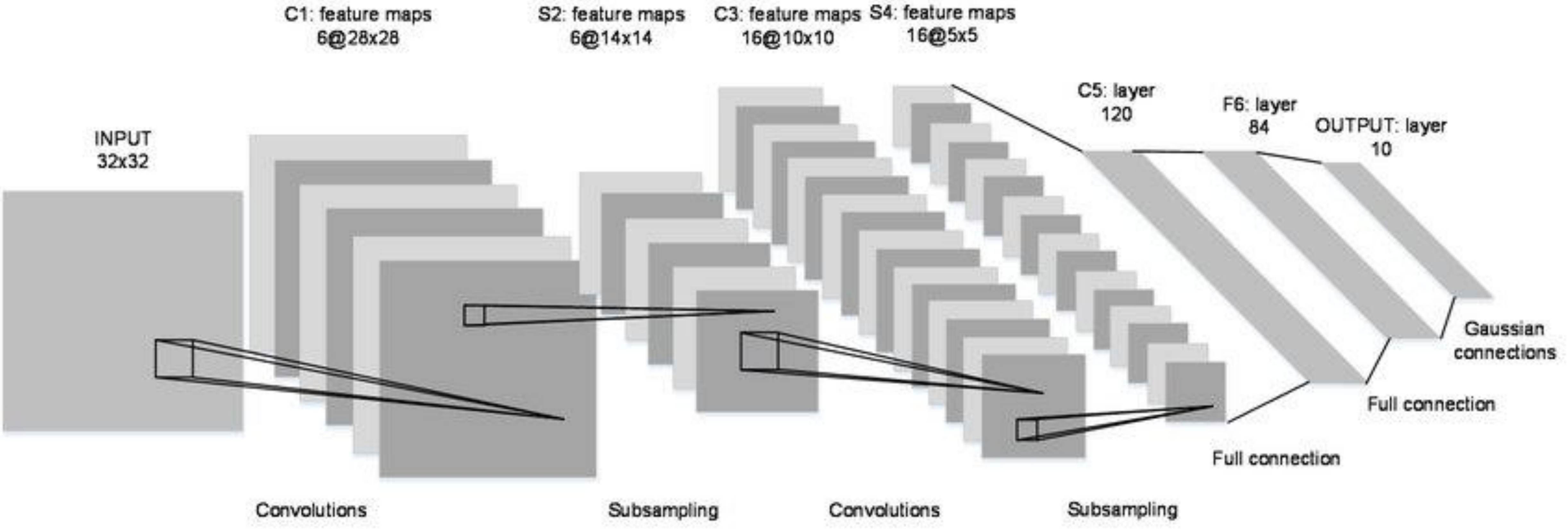
http://machinelearninguru.com/computer_vision/basics/convolution/convolution_layer.html

CNN
Off-the-Shelf Architectures

In general, the more convolution layers/steps we have, the more complicated/sophisticated features our network will be able to learn to recognize.

CNN-Variations in CNN Architecture

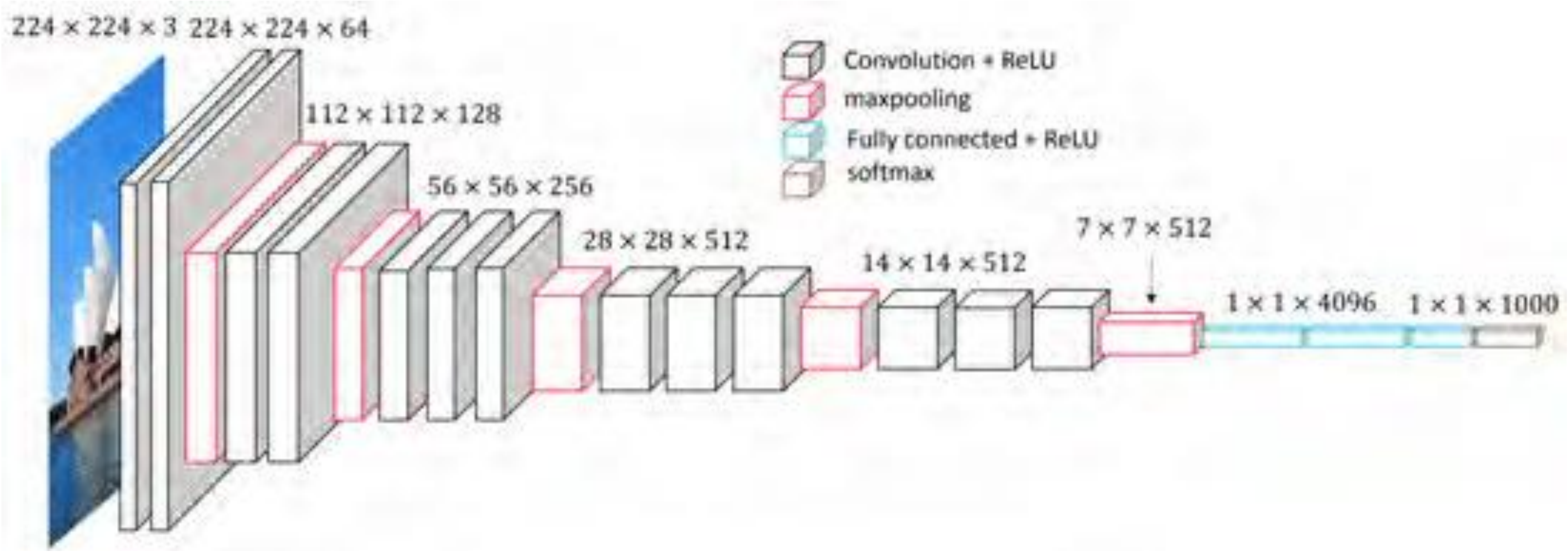
- LeNet Architecture



http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

CNN-Variations in CNN Architecture

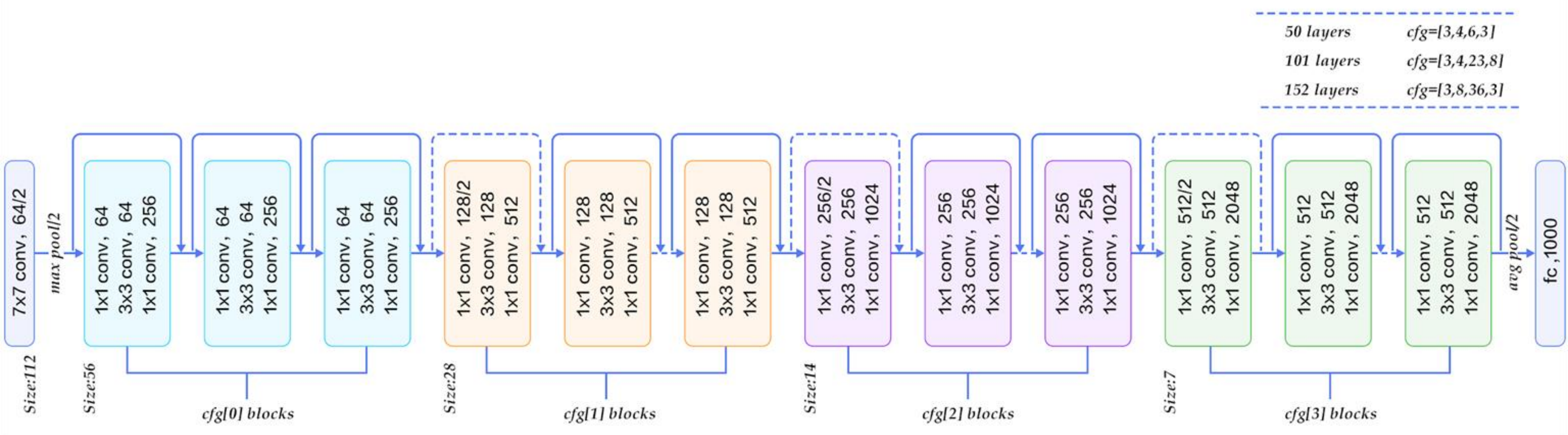
- VGGNet Architecture (about 140M parameters)



https://arxiv.org/pdf/1409.1556.pdf
http://www.robots.ox.ac.uk/~vgg/research/very_deep/
https://medium.com/coinmonks/paper-review-of-vggnet-1st-runner-up-of-ilsvlc-2014-image-classification-d02355543a11

CNN-Variations in CNN Architecture

- ResNet Architecture (>100 layers)



50 layers	cfg=[3,4,6,3]
101 layers	cfg=[3,4,23,8]
152 layers	cfg=[3,8,36,3]

7x7 conv, 64/2

max pool/2

1x1 conv, 64
3x3 conv, 64
1x1 conv, 256

1x1 conv, 64
3x3 conv, 64
1x1 conv, 256

1x1 conv, 64
3x3 conv, 64
1x1 conv, 256

1x1 conv, 128/2
3x3 conv, 128
1x1 conv, 512

1x1 conv, 128
3x3 conv, 128
1x1 conv, 512

1x1 conv, 128
3x3 conv, 128
1x1 conv, 512

1x1 conv, 256/2
3x3 conv, 256
1x1 conv, 1024

1x1 conv, 256
3x3 conv, 256
1x1 conv, 1024

1x1 conv, 256
3x3 conv, 256
1x1 conv, 1024

1x1 conv, 512/2
3x3 conv, 512
1x1 conv, 2048

1x1 conv, 512
3x3 conv, 512
1x1 conv, 2048

1x1 conv, 512
3x3 conv, 512
1x1 conv, 2048

avg pool/2

fc, 1000

Size:112

Size:56

cfg[0] blocks

Size:28

cfg[1] blocks

Size:14

cfg[2] blocks

Size:7

cfg[3] blocks

https://arxiv.org/abs/1512.03385

https://medium.com/@14prakash/understanding-and-implementing-architectures-of-resnet-and-resnext-for-state-of-the-art-image-cf51669e1624

https://www.codeproject.com/Articles/1248963/Deep-Learning-using-Python-plus-Keras-Chapter-Re

Programming Assignment 7

Make sure you install tensorflow and keras correctly in Python 3.5+ environment.

Programming Assignment 7

Using the BT2101 Tutorial 7 Notebook (Convolutional Neural Network.ipynb), please answer the questions in the jupyter notebook

Answer all in the jupyter notebook.

Instructions

Submit Python Notebook to the submission folder and Named:
AXXXX_T7_program.ipynb

Include your answers in the jupyter notebook

- You need to show outputs, instead of just showing functions.

Submit by Tuesday OCT-23 (by 12:00pm noon)

- Based on Convolutional Neural Network.ipynb

Thank You!

Appendix

1. The performance of Pooling Layer

- http://www.ais.uni-bonn.de/papers/icann2010_maxpool.pdf

2. Backpropagation details in CNN

- https://grzegorzgwardys.wordpress.com/2016/04/22/8/
- https://pdfs.semanticscholar.org/5d79/11c93ddcb34cac088d99bd0cae9124e5dcd1.pdf
- https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/