Week 1

Welcome to 42028

Deep Learning & Convolutional Neural Network

Teaching Team

- **Subject Co-ordinator & Lecturer:**

  A/Prof. Nabin Sharma

  Email: Nabin.Sharma@uts.edu.au

- **Tutorial Staff**:
  - Mr. Suryansh Sharma
  - Mr. Gitarth Vaishnav

Objectives

- Introduction to Machine Learning fundamentals.
- Understand the difference between traditional machine learning and Deep Learning.
- Able to Build, Train, and Test a Convolutional Neural Network (CNN) from scratch.
- Learn to use libraries and framework for implementing Deep CNN architectures.
- Collaboratively Analyze, design, implement and test solutions to real-world computer vision related problems.

Assessment Items

- Assessment Task 1: Assignment-1 (Individual)
  - **Weight:** 30%
  - **Task:**
    1. Implement a simple kNN classifier for digit classification
    2. Implement a Linear classifier using SVM for digit classification
    3. Implement a Linear classifier using Neural Network for digit classification
    4. Compare the three implementations in terms of classification accuracy and top choices.
  - **Due Date: 11.59pm Friday 25 March 2022 (Week-5)**
  - **Deliverables:** Short Report and code, via Canvas.

Assessment Items

- Assessment Task 2: Assignment-2 (Individual)
  - **Weight:** 40%
  - **Task:**
    1. Customize AlexNet/GoogleNet/ResNet and reduce/increase the layers. Train and test on images
    2. Implement a custom CNN architecture for object detection and localization.
    3. Train and test the custom architecture on a given dataset for detection of multiple Objects, using Faster RCNN or custom object detection methods.

       (Training, validation and testing datasets will be provided.)
  - **Due Date: 11.59pm Friday 13 May 2022. (Week -11)**
  - **Deliverables:** Short Report and code, via Canvas.

Assessment Items

- Assessment Task 3: Final Project (Group)
  - **Weight:** 30%
  - **Task:** Any one of the following problems, or any combination:
    1. Design/Implement an image classification algorithm.
    2. Design/develop an object detection system for detecting specific objects in a video and localizing them.
    3. Develop a clear problem statement that is within the capabilities of CNNs and design and capture a dataset of significant size that addresses this problem.
    4. Compare a series of algorithms against each other to determine optimum performance, and then suggest new approaches that improve performance.

  Student must Develop a user interface that allows the operational use of an algorithm to achieve a real-world

- **Final Due Date:** **11.59pm Friday 20 May 2022, (Week -12)**
  Demo schedule to be informed later
- **Deliverables:** Multiple deliverables, via Canvas.

Assessment Items

Assessment Task 3: Final Project (Group)

Deliverable details:

1. Part-A(Report): Group details, Project title and abstract : Week 4, Monday: Weightage 10%
2. Part-B(Report): Dataset details, GUI design plan, Implementation/development plan: Week-7, Monday : Weightage 10%
3. Part-C(Report): Initial experimental results: Week 10, Monday: Weightage 10%
4. Part-D(GUI design/report): Week 11, Monday: Weightage 10%
5. Part-E: Complete project (GUI, experimental results etc.): Week 12, Weightage 50%
6. Part-F: Presentation and Demo, Recorded video: Week 14, Weightage 10%

**For extract dates and submission requirements, please check Canvas

Labs/Tutorials

- **Duration**: 3Hrs
  - 2Hrs (Tutor guided)
  - 1Hr (For project related activities, discussion, feedback, progress reporting, etc.), students to work in groups.
- Implementation of Algorithms using Python
- Google Colab will be used

Labs/Tutorials

- Deep Learning frameworks: TensorFlow, Keras, etc
- Datasets will be provided

Important Tasks:

- Project Groups to be formed by Week-3 Friday.
- Trello/other tools can be used for your project management.

Consultation

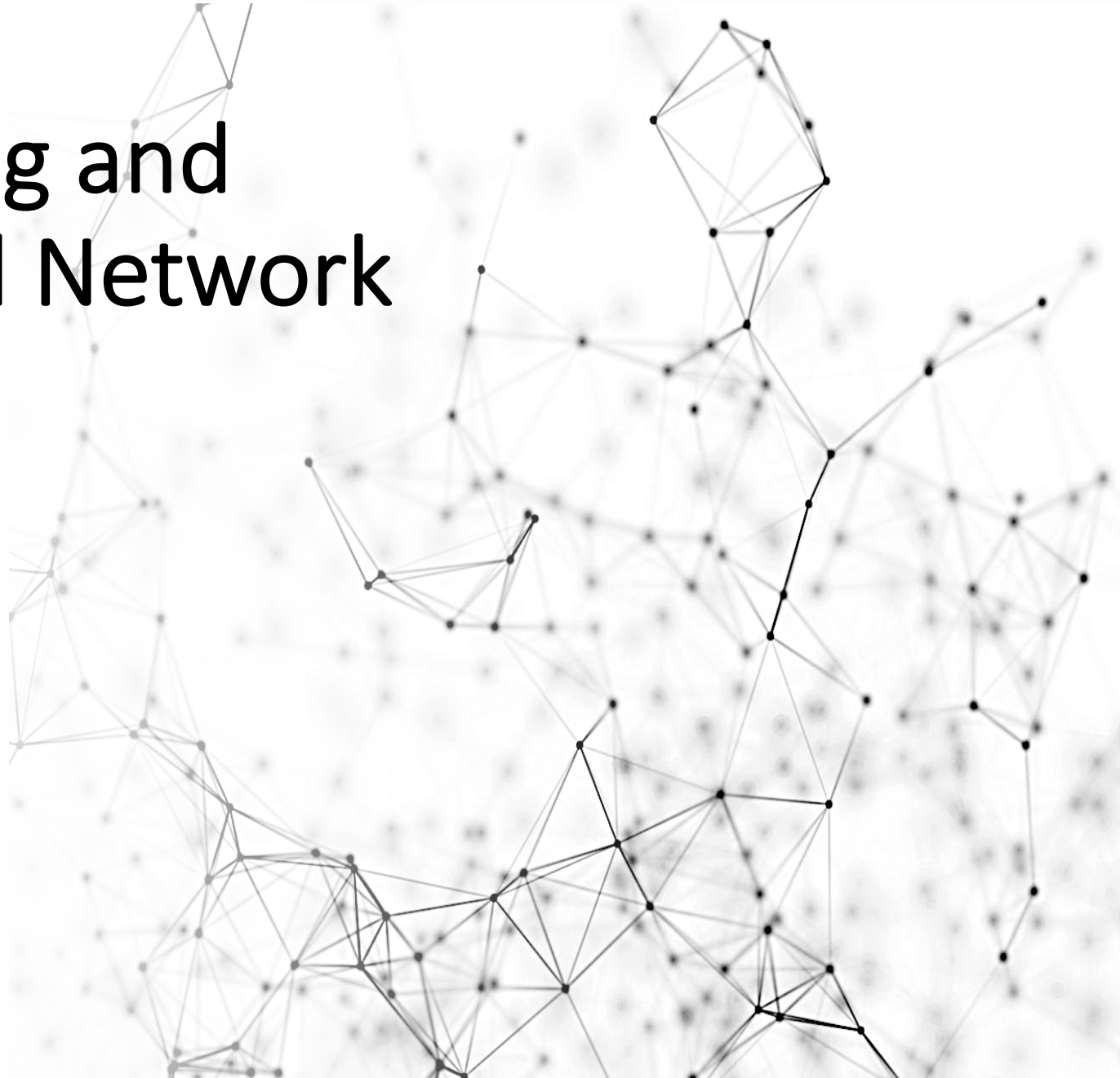
- **Schedule**: Every Thursday, 11:00-12:00 (Via Zoom or face-to-face)

- **Please check Canvas for further details.**

Week 1 Checklist

- ☐ Checked Subject outline
- ☐ Zoom link for Lecture and how to join
- ☐ Aware of my lab/tutorial schedule
- ☐ Room number for on-campus lab/tutorial session
- ☐ Have working knowledge of Python (Review materials available on Canvas)
- ☐ Clear about the assignment deadlines
- ☐ Clear about the assignment requirements
- ☐ I know how to use Canvas and can find the subject materials, assignments, FAQs, discussion board, etc.
- ☐ Prepare questions for having a clear understanding of the subject requirements and expectation.

42028: Deep Learning and Convolutional Neural Network



Week-2 Lecture

Machine Learning and Image Processing Basics

Outline

- Types of Machine Learning System
- Supervised and Un-supervised learning
- Support Vector Machine (SVM)
- K-Nearest Neighbour
- Evaluation Metrics
- Image Processing Basics
- Edge Detection using Convolution

Machine Learning Basics

Type of Machine Learning Systems

Supervised Learning
Unsupervised Learning
Semi-supervised Learning
Reinforcement Learning

Depending on whether the system is trained with human supervision

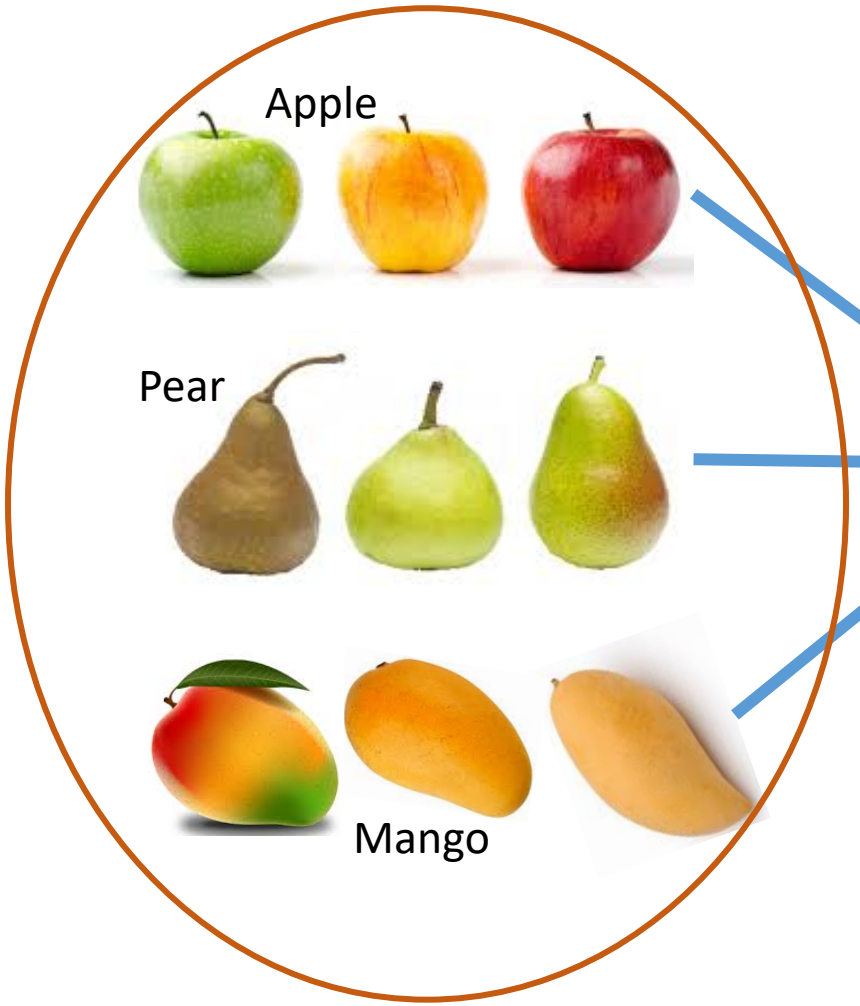
Whether System can learn on the fly

Batch and Online Learning

Instance-based and Model-based Learning

Comparing data points or detect patterns in training data to build a predictive model

Supervised Learning



Labelled data for training
(Object + Desired Output Label)

Supervised Learning

Predictive Model

?

Pear

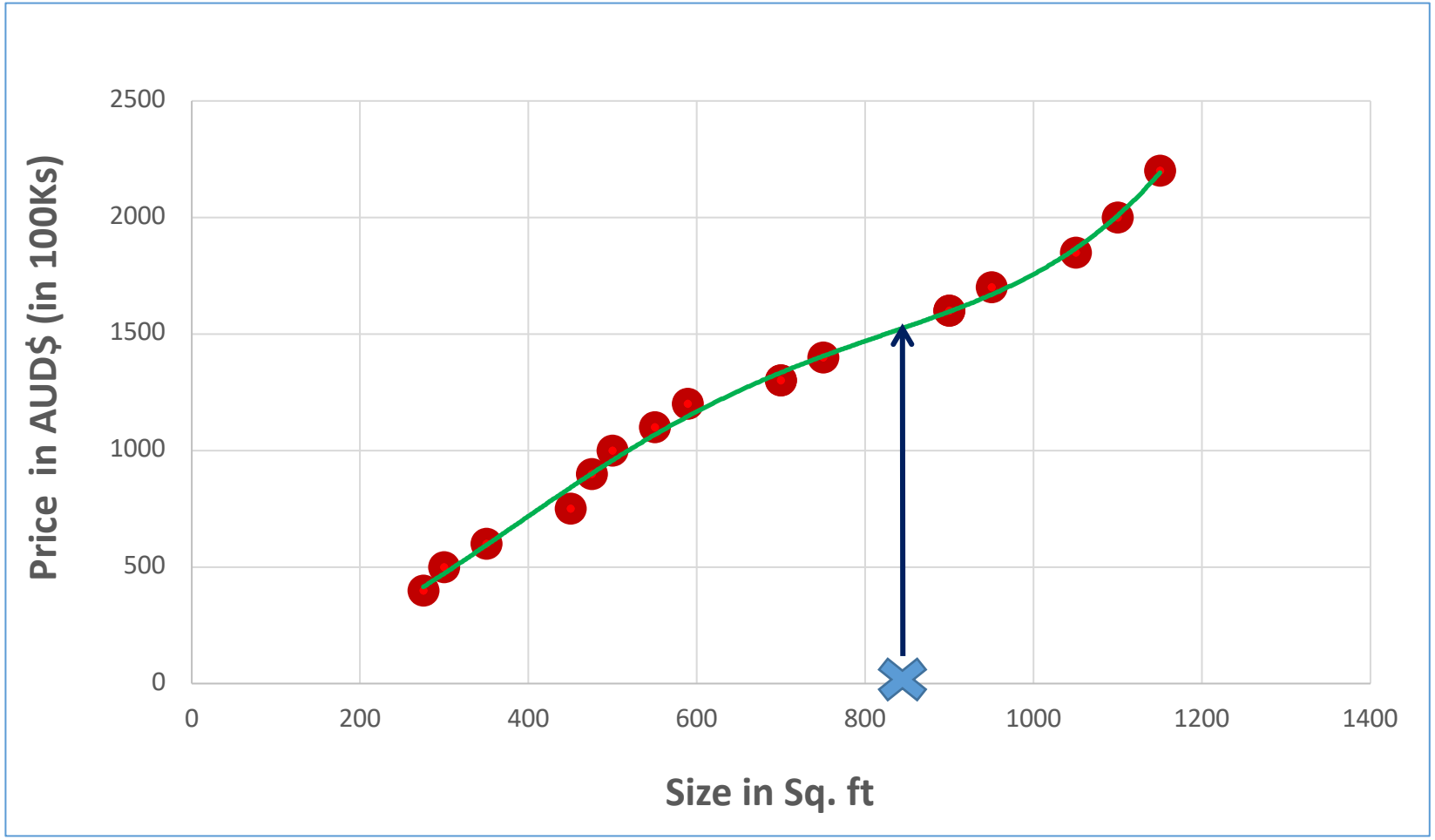
Classification Task

Supervised Learning

House Price prediction
Feature:
- Size of the house

To Predict:
- Price of the house

Regression Task

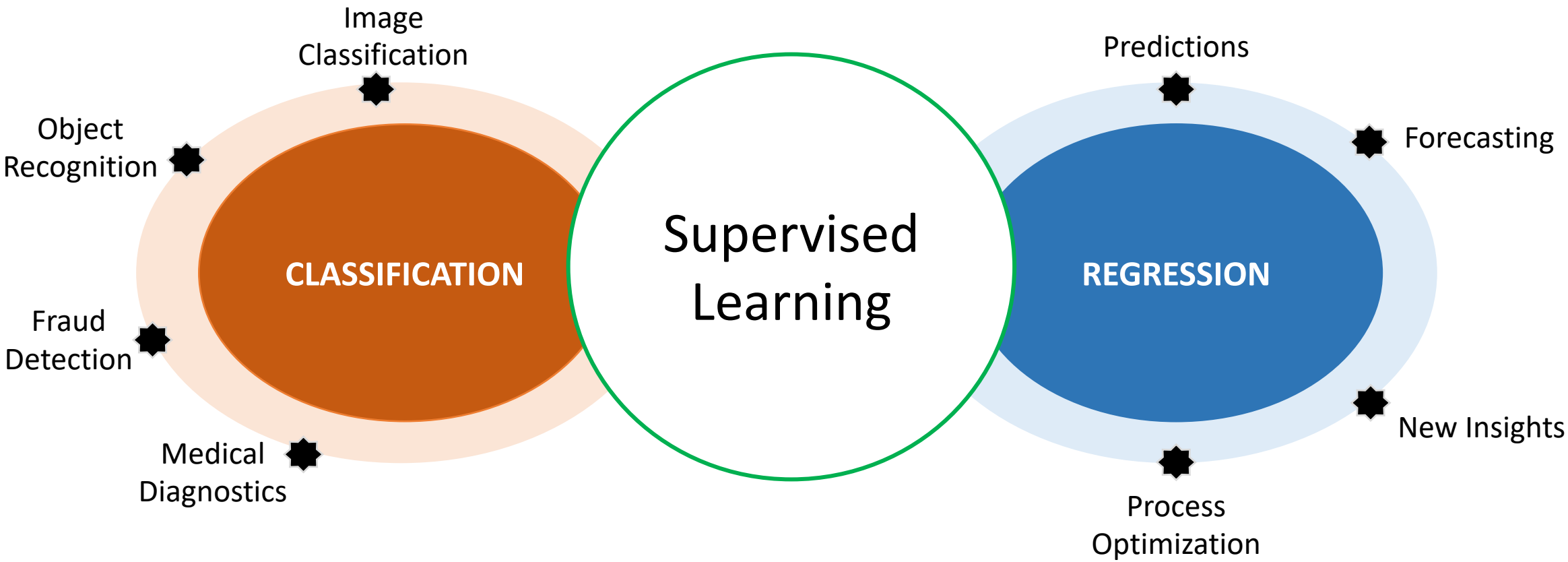


Supervised Learning

Important Algorithms:

- K-Nearest Neighbours
- Logistic regression
- Support Vector Machines (SVMs)
- Neural Networks (*some of them can be unsupervised)

Supervised Learning Examples



Unsupervised Learning

Unlabelled data for training

Unsupervised Learning

Clustering Task

Groups of similar fruits

Unsupervised Learning

Important Algorithms:

- k-means
- Expectation Maximization

Support Vector Machines (SVM)

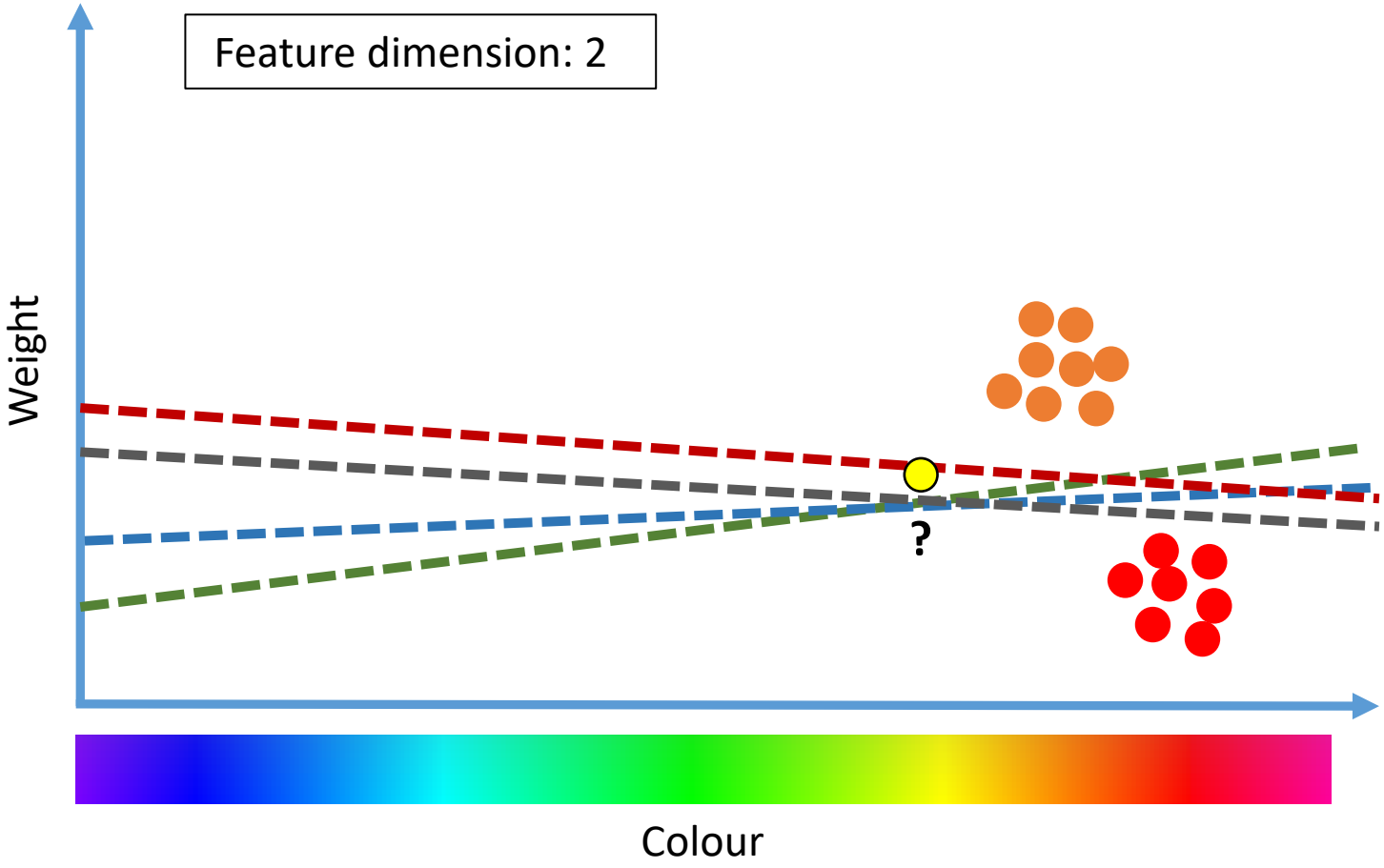
- A Support Vector Machine is a very powerful and versatile Machine Learning model, capable of performing linear or non-linear classification, regression, and also outlier detection.
- Defined by a separating hyperplane
- Suitable for small or medium sized datasets

Reference and Pre-Reading:

Theory: https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72

Implementation: https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-coding-edd8f1cf8f2d

Support Vector Machines (SVM)



Orange

Apple

SVM finds the best line or hyper-plane which will fairly separates the classes

Support Vector Machines (SVM)

Example: Using sklearn for SVM classification (Partial code snippet)

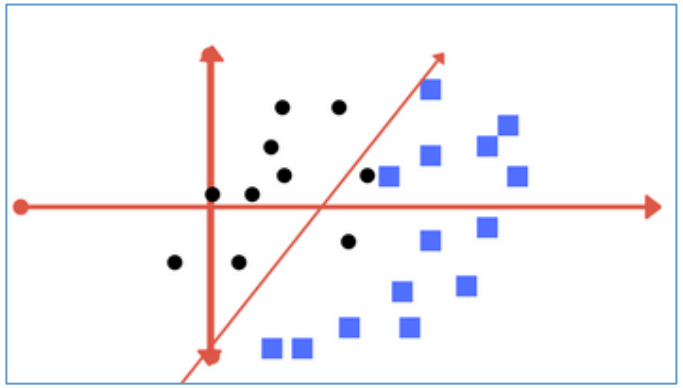
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm, datasets

# import some data to play with
iris = datasets.load_iris()
# Take the first two features. We could avoid this by using a two-dim dataset
X = iris.data[:, :2]
y = iris.target

# we create an instance of SVM and fit out data. We do not scale our
# data since we want to plot the support vectors
C = 1.0  # SVM regularization parameter
models = (svm.SVC(kernel='linear', C=C),
          svm.LinearSVC(C=C),
          svm.SVC(kernel='rbf', gamma=0.7, C=C),
          svm.SVC(kernel='poly', degree=3, C=C))
models = (clf.fit(X, y) for clf in models)
```

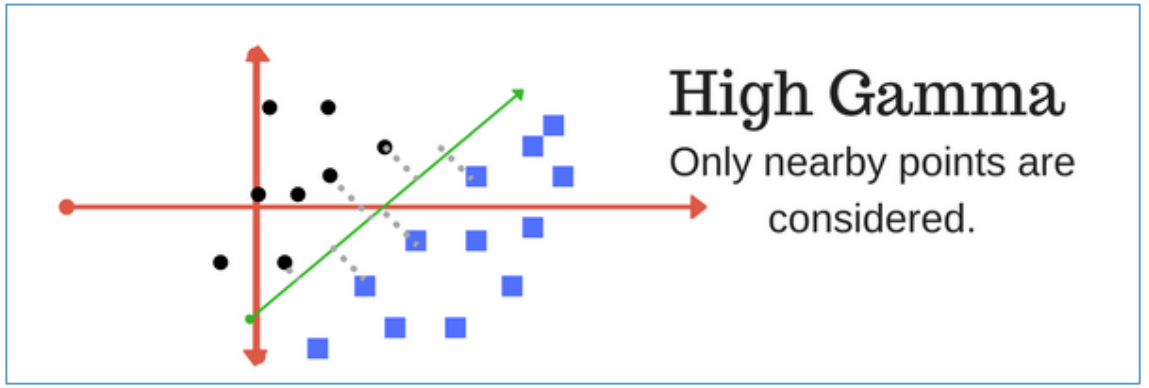
Reference: https://scikit-learn.org/stable/auto_examples/svm/plot_iris.html#sphx-glr-auto-examples-svm-plot-iris-py
https://en.wikipedia.org/wiki/Iris_flower_data_set

Support Vector Machines (SVM)

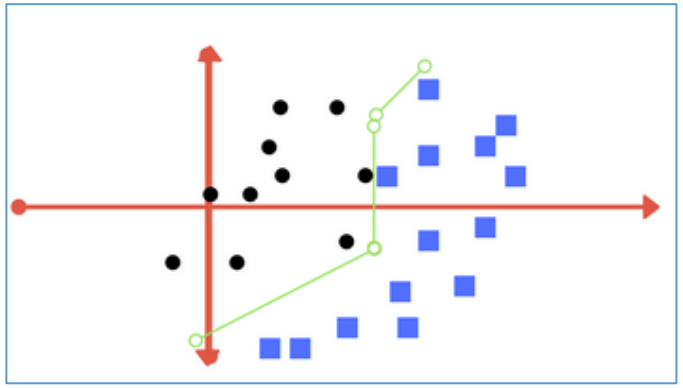
SVM Parameters: Kernel, Gamma, Regularization (C)

Low Regularization value





High Regularization value



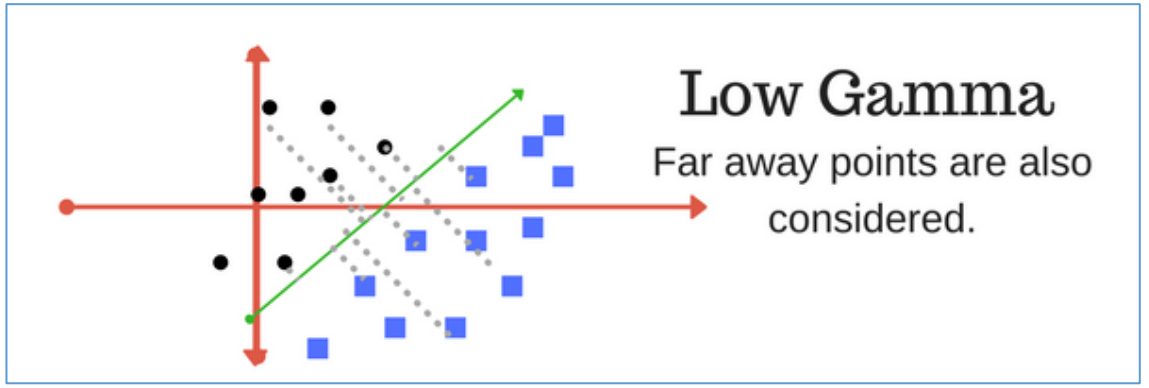
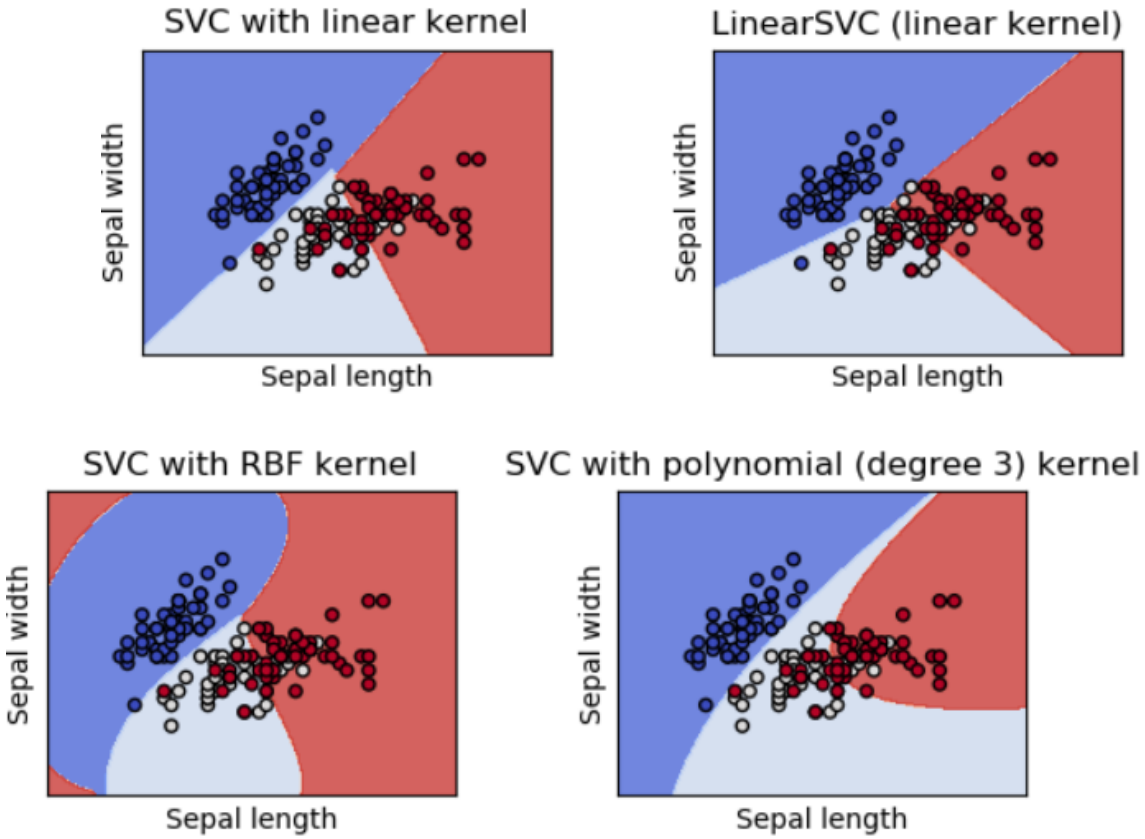


Image source and Reference: https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72

Support Vector Machines (SVM)

Example: Using sklearn for SVM classification



Iris flower data set

Reference: https://scikit-learn.org/stable/auto_examples/svm/plot_iris_svc.html
https://en.wikipedia.org/wiki/Iris_flower_data_set

K-Nearest Neighbour (KNN)

- A simple supervised learning algorithm.
- Can be used for both classification and regression
- Non-parametric: doesn’t make any assumption on the data distribution
- Training data is retained to make future predictions

Reference and Pre-Reading: https://towardsdatascience.com/knn-using-scikit-learn-c6bed765be75
https://scikit-learn.org/stable/modules/neighbors.html

K-Nearest Neighbour (KNN)

How does it work?

- Computes distance between the new sample and all training samples
- Distance measure: Euclidean, Manhattan etc.
- Picks ‘k’ entries in the training set which are closest to the new sample
- Majority voting decides the class of the new sample

Reference and Pre-Reading: https://towardsdatascience.com/knn-using-scikit-learn-c6bed765be75
https://scikit-learn.org/stable/modules/neighbors.html

K-Nearest Neighbour (KNN)

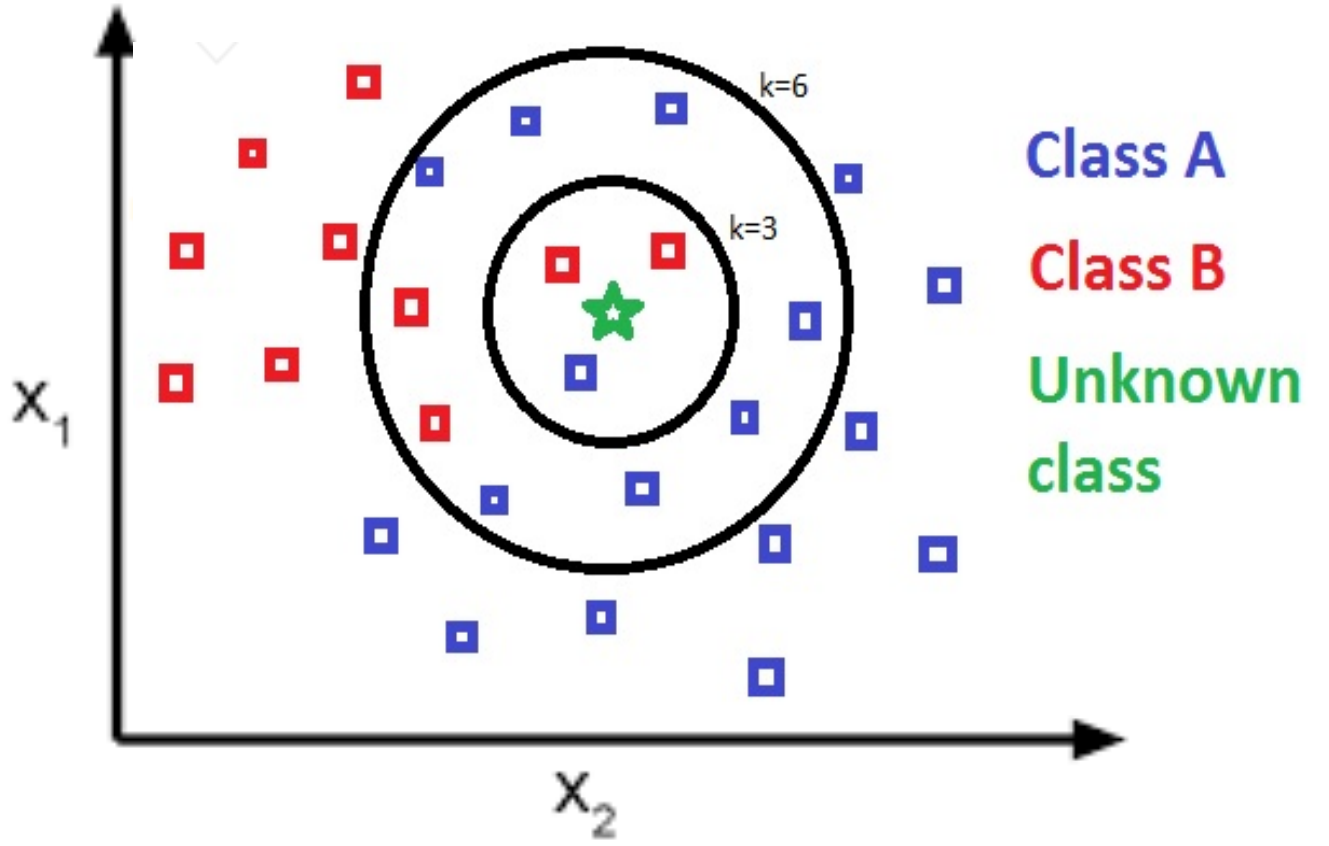


Image Source: https://towardsdatascience.com/knn-using-scikit-learn-c6bed765be75
https://scikit-learn.org/stable/modules/neighbors.html

Evaluation Metrics

- **Precision & Recall**

What are the "correct" cells?

- *TN:* (Number of True Negatives),
  i.e., patients who did *not* have cancer whom we correctly diagnosed as *not* having cancer.

- *TP:* (Number of True Positives),
  i.e., patients who did have cancer whom we correctly diagnosed as having cancer

Precision: TP/Cancer Diagnoses

True state \ Diagnoses	No Cancer	Cancer
No Cancer	TN	FP
Cancer	FN	TP

Recall: TP/Cancer True States

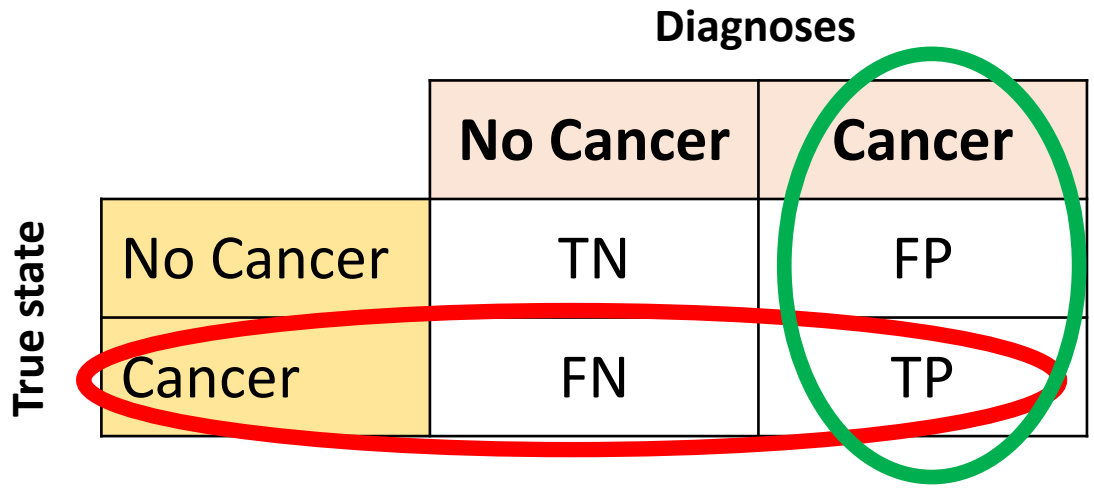
Evaluation Metrics

- **Precision & Recall**

what are the "error" cells are:

- *FN:* (Number of False Negatives), i.e., patients who did have cancer whom we incorrectly diagnosed as *not* having cancer
- *FP:* (Number of False Positives), i.e., patients who did *not* have cancer whom we incorrectly diagnosed as having cancer

Precision: TP/Cancer Diagnoses



Recall: TP/Cancer True States

$$\text{Precision} = (TP)/(TP+FP)$$

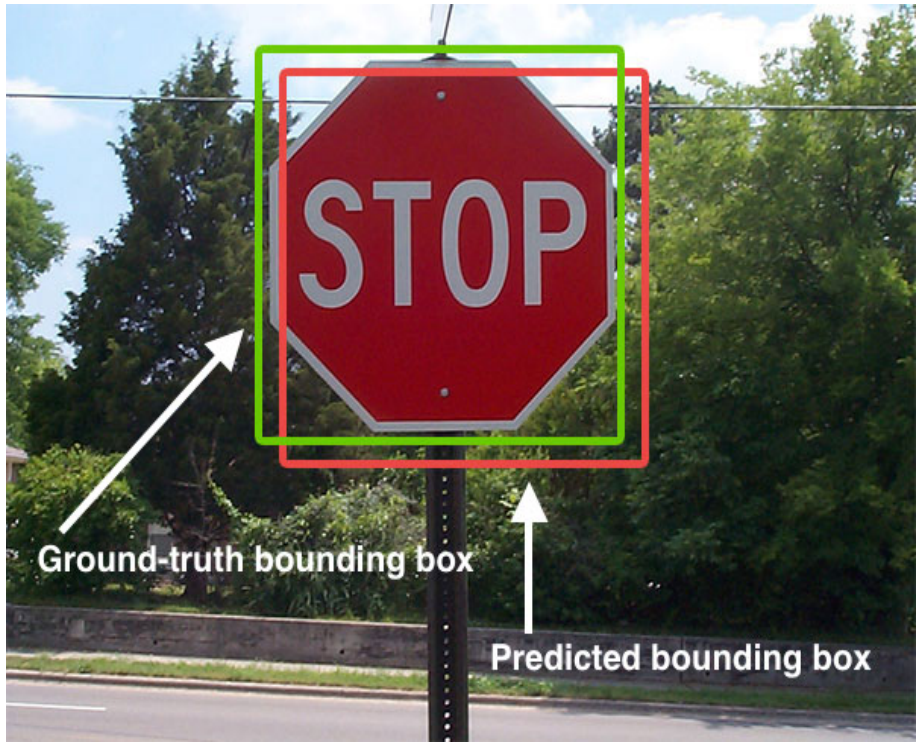
$$\text{Recall} = (TP)/(TP+FN)$$

Evaluation Metrics

- **Intersection over Union (IoU):**

Intersection over Union is a metric used for the evaluation of an object detector, i.e. how good is the predicted bounding box for an object detected closely matches

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$



Reference: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

Image Processing Basics

Image Processing Basics

What is a digital image?

- Digital images are made of picture elements called ***Pixels.***
- It is an array, or a matrix of ***Pixels*** arranges in *columns* and *rows*.
- Each ***Pixel*** has its own *intensity* value, or *brightness*

- Intensity values in digital images are defined by ***bits***
- For a standard 8 bits image, a pixel can have $2^8 = 256$ (0 – 255) values.
- Black & White images have a single 8-bits intensity range.

Image Processing Basics

Image dimension = 5 X 5

***f*(2, 3) = 170 (Pixel/intensity value)**

Hence, an image may be defined as a 2D function ***f*(x, y)** , where, *x* and *y* are spatial co-ordinates, and the amplitude of ***f*** at **(*x* , *y*)** is the intensity or Gray level of the image at that point/pixel.

170	170	55	170	170
170	55	170	55	170
170	55	170	55	170
55	140	140	140	55
55	170	170	170	55

5 X 5 Gray scale image (8 bit)

Image Processing Basics

Image dimension = 5 X 5 X 3

No. of Channels = 3

Since, RGB image contains 3 X 8-bits of intensities, they are referred to as 24-bit colour images.

So, 24-bit colour depth

= 8 X 8 X 8 bits

= 256 X 256 X 256 colours

= ~16 million colours

5 X 5 X 3 colour image (24 bit)

Image Thresholding

- Easiest method for image segmentation!
- Converts gray-scale image into a binary image

  If *f(x,y) > Threshold*, then *f(x,y) = 0* else *f(x,y) = 255*

Binary Image (8-bit) has only two possible values of pixel intensity (0 and 1, or B & W)

170	170	55	170	170
170	55	170	55	170
170	55	170	55	170
55	140	140	140	55
55	170	170	170	55

Threshold = 100

255	255	0	255	255
255	0	255	0	255
255	0	255	0	255
0	255	255	255	0
0	255	255	255	0

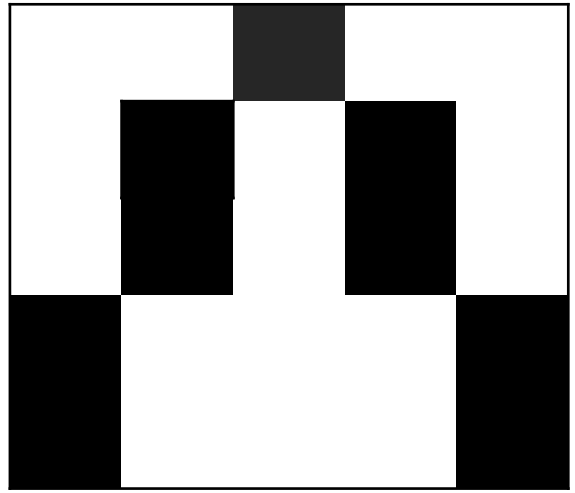


Image Thresholding

Original Image

Thresholding

Binary Image

Image Source: https://en.wikipedia.org/wiki/Thresholding_(image_processing)

Image Thresholding methods

- **Histogram shape:**

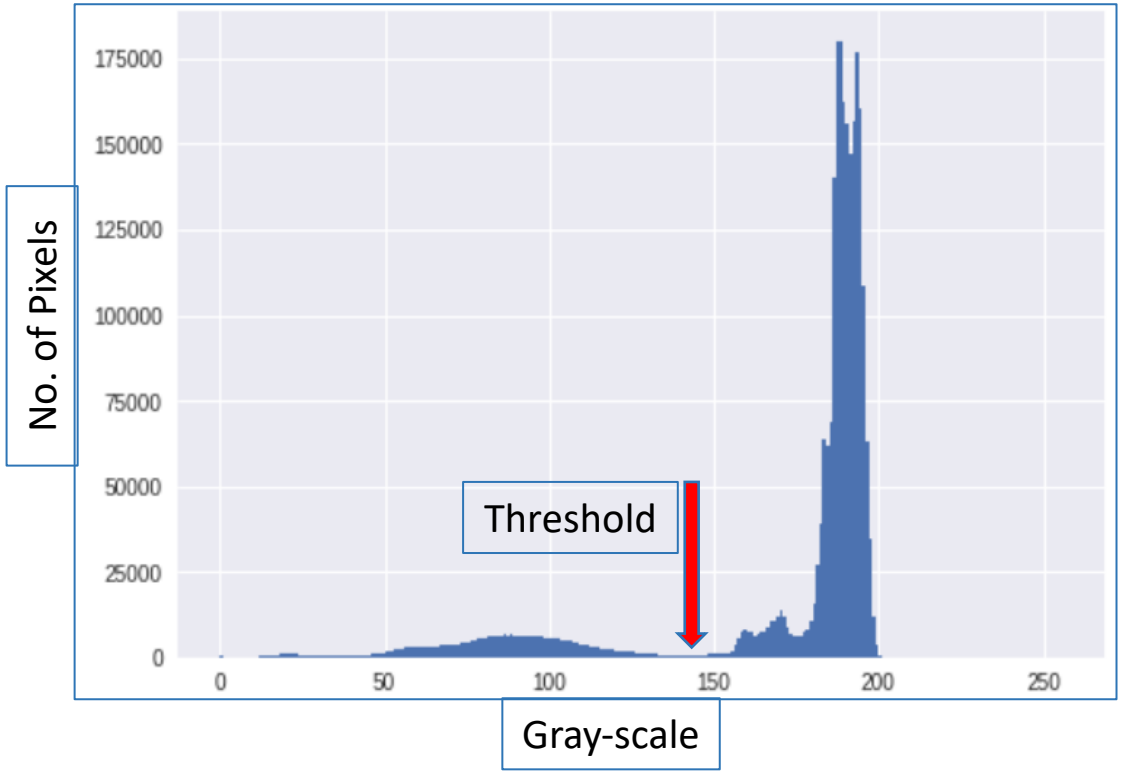
Peaks, valleys and curvature of the histogram are analysed.

- **Clustering based:**

The [1]Otsu method, very good for bimodal distribution

- **Adaptive thresholding:**

Instead of a single threshold, have thresholds for different regions in the image

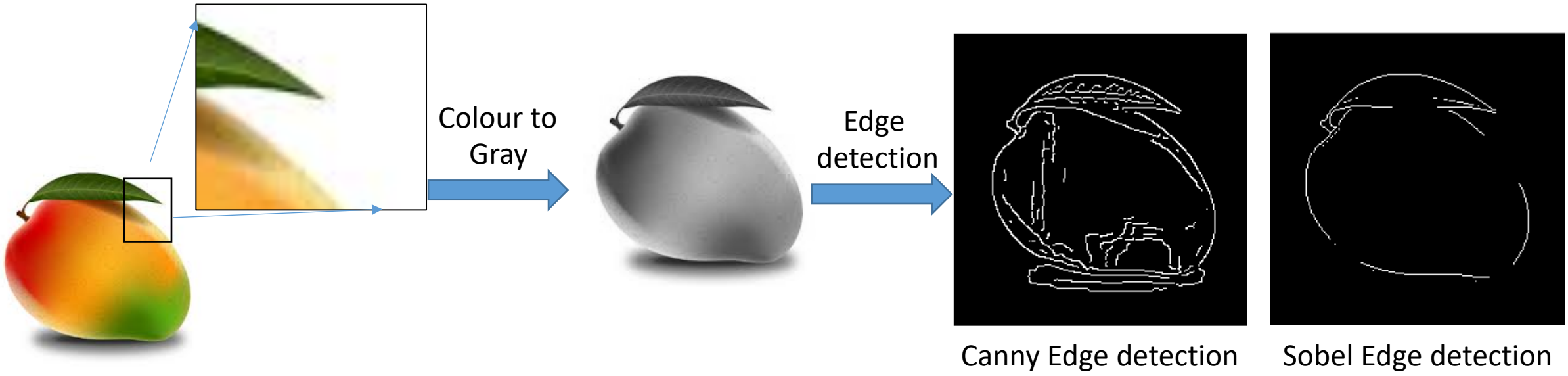


[1]Reference: https://en.wikipedia.org/wiki/Otsu%27s_method

Edge Detection

What is an edge?

- The points/pixels in an image where brightness/intensities changes sharply
- A simple and fundamental tools in image processing and computer vision, useful in feature detection/extraction



How to detect Edges in images? (Convolution)

100	**100**	**100**	**0**	**0**	**0**
100	100	100	**0**	**0**	**0**
100	100	100	**0**	**0**	**0**
100	100	100	**0**	**0**	**0**
100	100	100	**0**	**0**	**0**
100	100	100	**0**	**0**	**0**

6 X 6 dimension image

*

Convolution operator

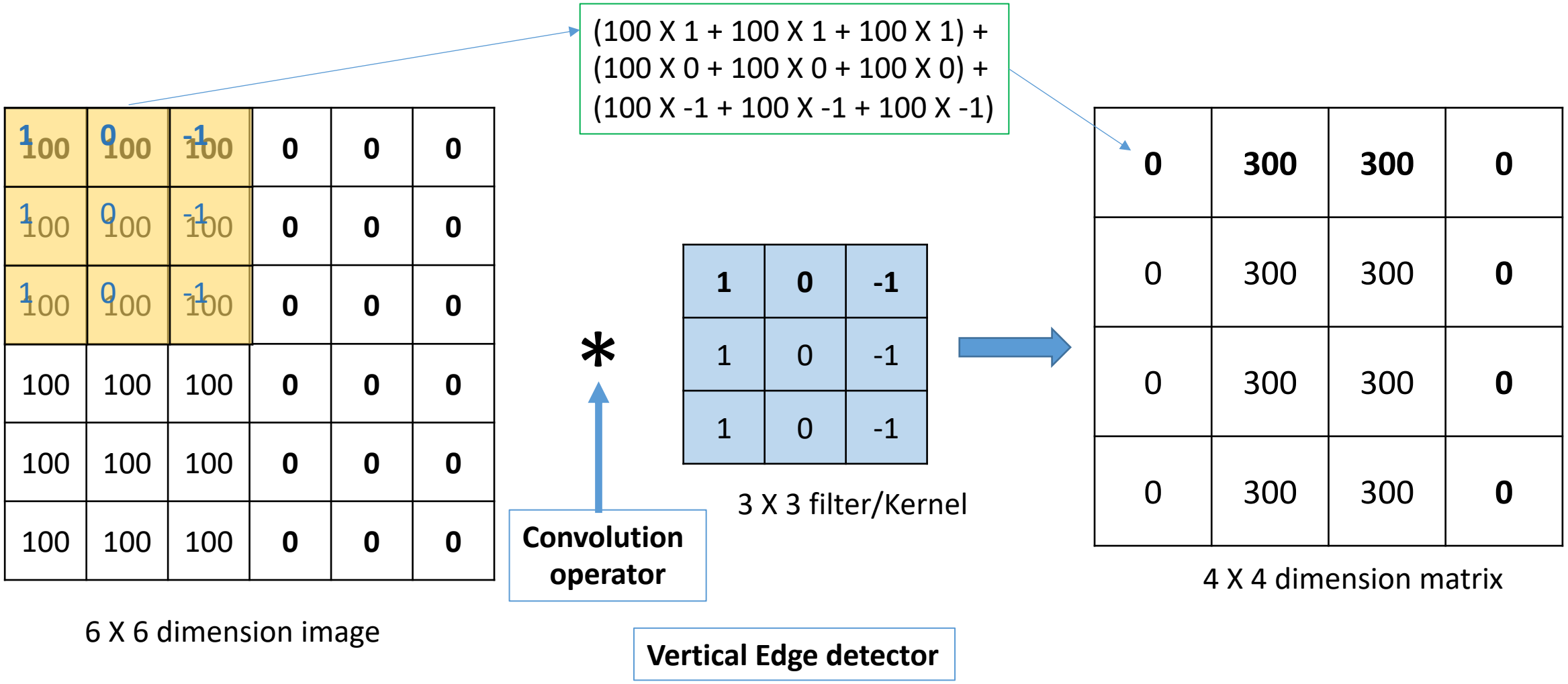
1	**0**	**-1**
1	0	-1
1	0	-1

3 X 3 filter/Kernel

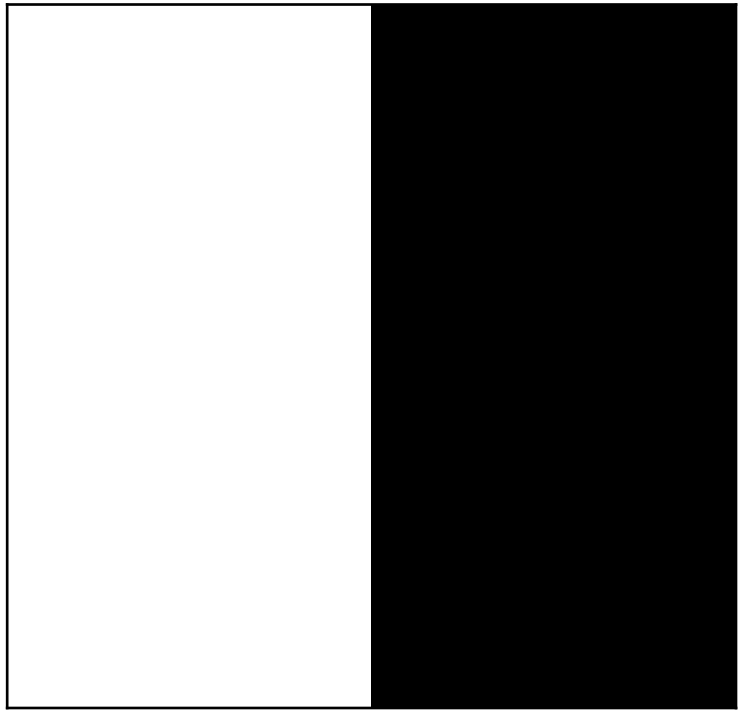
0	**300**	**300**	**0**
0	300	300	**0**
0	300	300	**0**
0	300	300	**0**

4 X 4 dimension matrix

How to detect Edges in images? (Convolution)



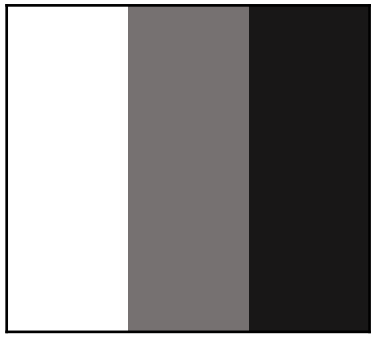
How to detect Edges in images? (Convolution)



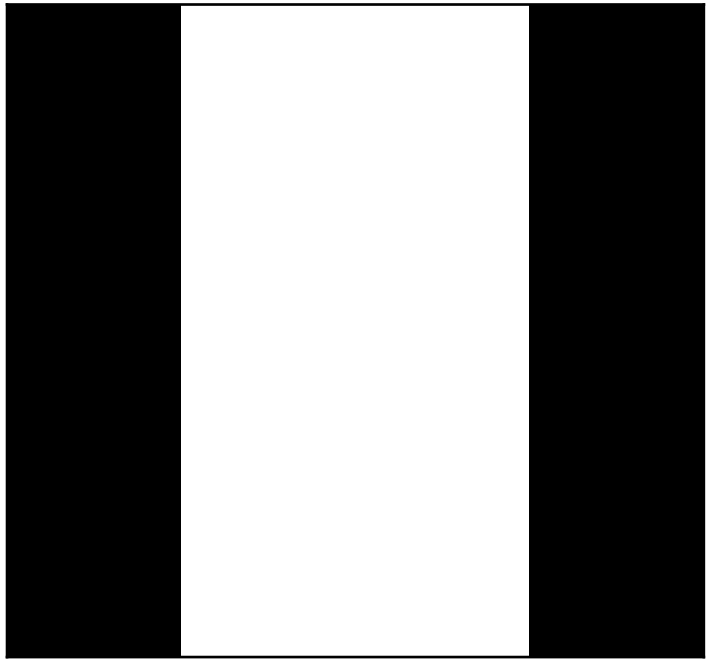
6 X 6 dimension image

*

Convolution operator



3 X 3 filter/Kernel



4 X 4 dimension matrix

Vertical Edge detector

How to detect Edges in images? (Convolution)

100	**100**	**100**	**100**	**100**	**100**
100	100	100	100	100	100
100	100	100	100	100	100
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

6 X 6 dimension image

*

Convolution operator

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

3 X 3 filter/Kernel

0	**0**	**0**	**0**
300	300	300	300
300	300	300	300
0	0	0	**0**

4 X 4 dimension matrix

Horizontal Edge detector

How to detect Edges in images? (Convolution)

6 X 6 dimension image

*

Convolution operator

3 X 3 filter/Kernel

4 X 4 dimension matrix

Horizontal Edge detector

Edge detection filters

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

3 X 3 filter/Kernel
For Vertical edges

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

3 X 3 filter/Kernel
For Horizontal edges

Prewitt Filters

1	**2**	**1**
0	0	0
-1	-2	-1

3 X 3 filter/Kernel
For Horizontal edges

1	**0**	**-1**
2	0	-2
1	0	-1

3 X 3 filter/Kernel
For Vertical edges

Sobel Filters

Sobel edge detection - Example

Original

Sobel X

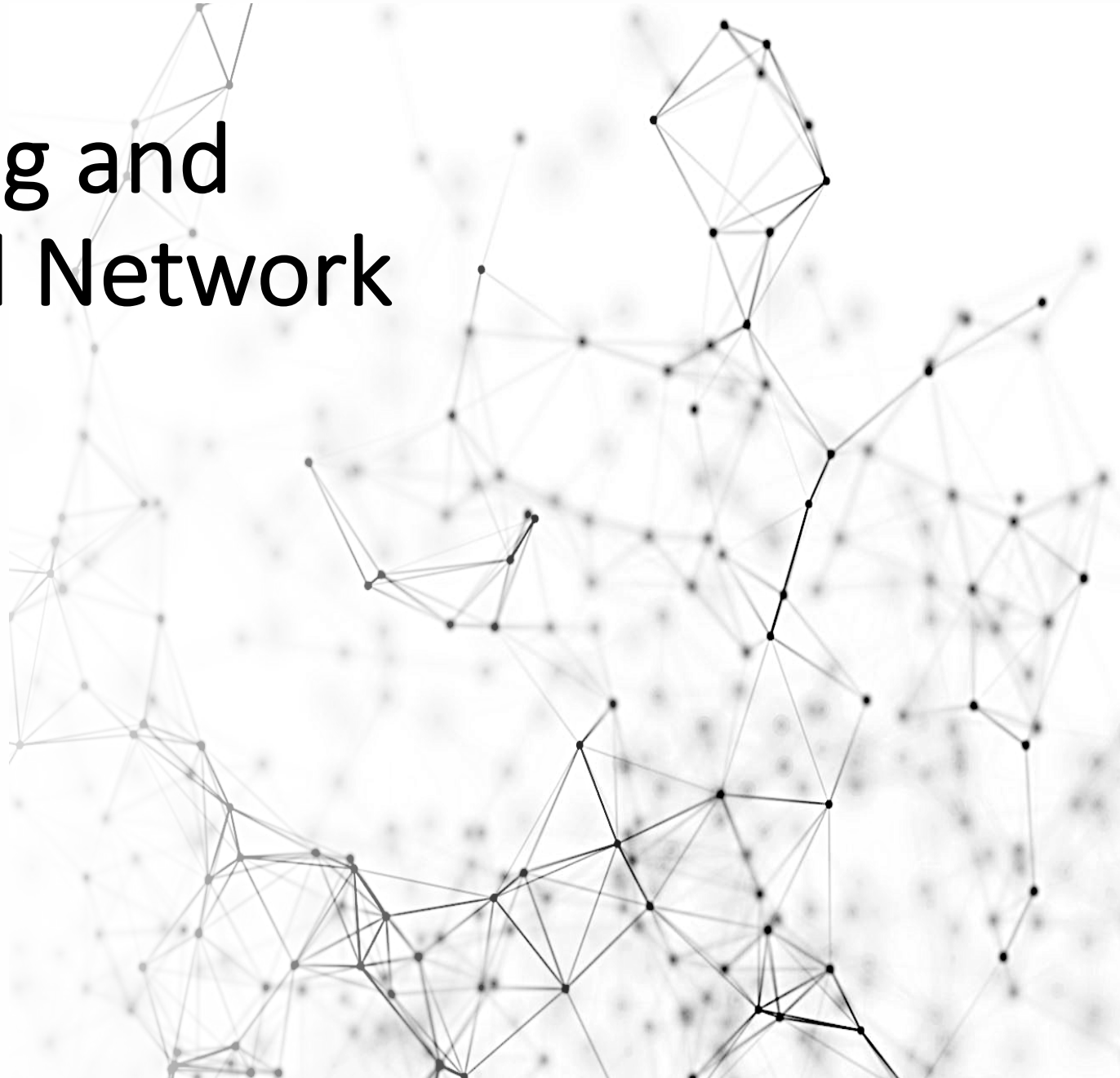
Sobel Y

Image Source: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_gradients/py_gradients.html

Convolutions in CNN

- Convolutions are very important operation in a Convolutional Neural Networks (CNN)
- Filters weights are not fixed, but learned during the training operations of a CNN for a specific task!
- Multiple filters are used in CNNs

42028: Deep Learning and Convolutional Neural Network



Week-3 Lecture

Feature Extraction and Neural Network Basics

Outline

- Image Gradient
- Histogram of Oriented Gradient (HoG)
- Local Binary Pattern
- ANN Basics
- ANN Learning Process
- Logistic Regression using ANN
- Gradient Descent

Features Extraction

HoG (Histogram of Oriented Gradient*)

What is an Image Gradient?

- It is a directional change in the intensity or color in an Image.
- Can be used to extract valuable information from images.
- Commonly used in edge detection.

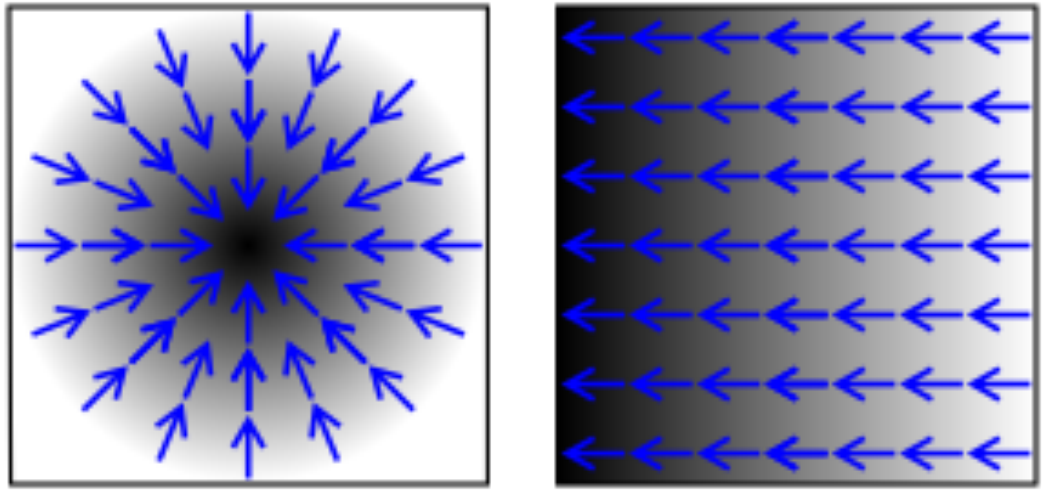
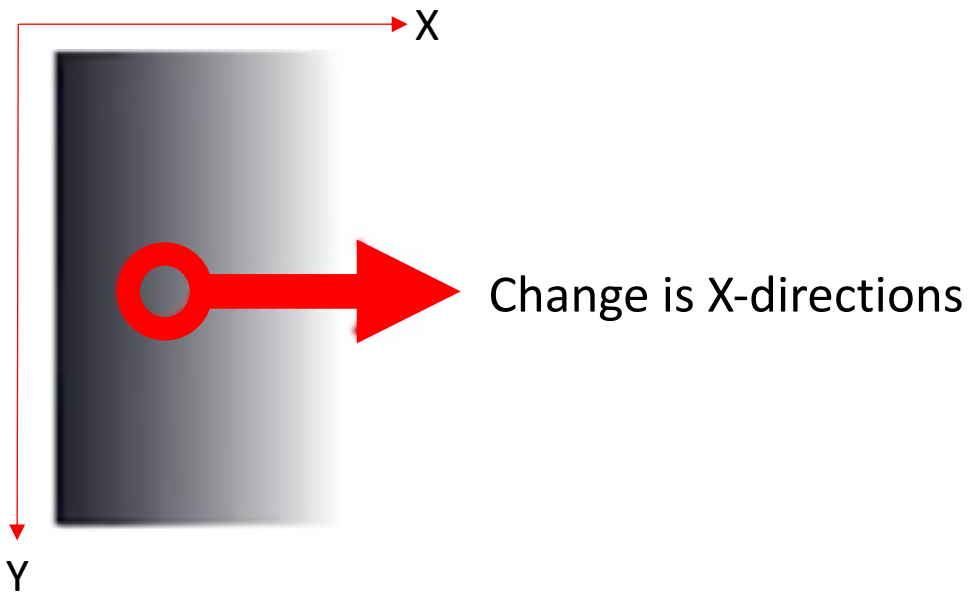


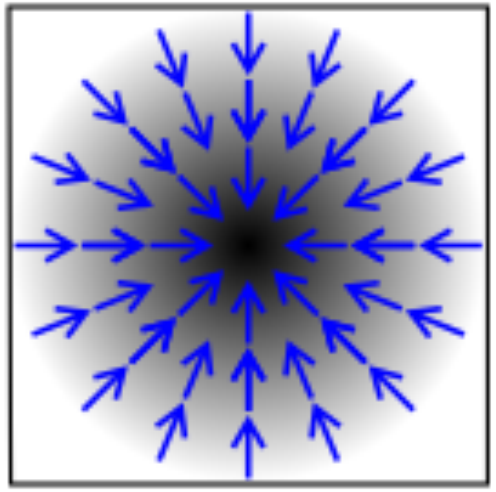
Image source: https://en.wikipedia.org/wiki/Image_gradient

Reference: http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf

HoG (Histogram of Oriented Gradient*)

What is an Image Gradient?





Combining both X and Y direction to estimate if changes are in both directions

Image source: https://en.wikipedia.org/wiki/Image_gradient

Reference: http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf

HoG (Histogram of Oriented Gradient)

Step -1: Computing Image *Gradient*:

1. Use the horizontal and vertical filters to compute gradient values

$g_x =$ | -1 | 0 | 1 | $* I$

Horizontal filter

Gradient is X-directions

$g_y =$ | -1 | / | 0 | / | 1 | $* I$

Vertical filter

Gradient is y-directions

Image source: https://en.wikipedia.org/wiki/Image_gradient

HoG (Histogram of Oriented Gradient)

2. Compute the strength/magnitude and direction of gradient.

Strength/Magnitude(g) = $\sqrt{g_y^2 + g_x^2}$

Direction $\theta = \tan^{-1}\left[\frac{g_y}{g_x}\right]$

Example →

X	100	X
70	60	120
X	50	X

g_x = |-70 + 120| = 50
g_y = |-100 + 50| = 50

Gradient Magnitude = ~70.7
Direction/Angle = 45°

Image source: https://en.wikipedia.org/wiki/Image_gradient

HoG (Histogram of Oriented Gradient)

Step -2: Create orientation histogram:

- Divide the image into small connected regions called *Cells*

which is a 8 X 8 patch

- Create cell histogram based on gradient direction and magnitude
- 64 (8 X 8) gradient vectors are put into a 9-bin histogram
- The bins are the gradient directions (θ) quantized into 9-bins

Reference: https://tanasecucliciu.wordpress.com/2016/06/08/programming-histogram-of-oriented-gradients-hog-explained/
https://www.learnopencv.com/histogram-of-oriented-gradients/

HoG (Histogram of Oriented Gradient)

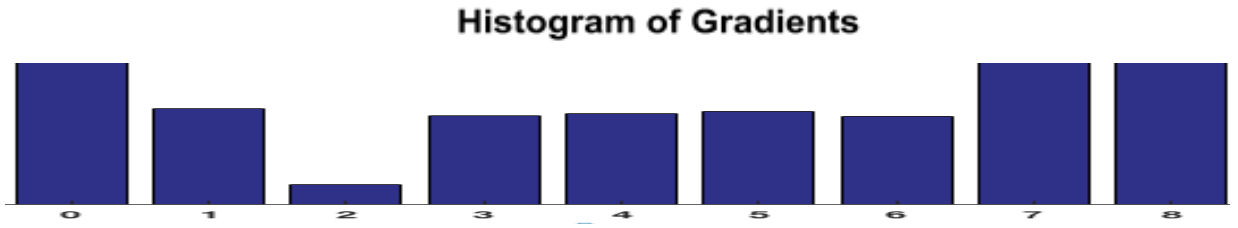
Pixel with blue circle has an angle of 80 degrees and magnitude of 2

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

Gradient Direction

2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

Gradient Magnitude



Reference: https://tanasecucliciu.wordpress.com/2016/06/08/programming-histogram-of-oriented-gradients-hog-explained/
Image source: https://www.learnopencv.com/histogram-of-oriented-gradients/

HoG (Histogram of Oriented Gradient)

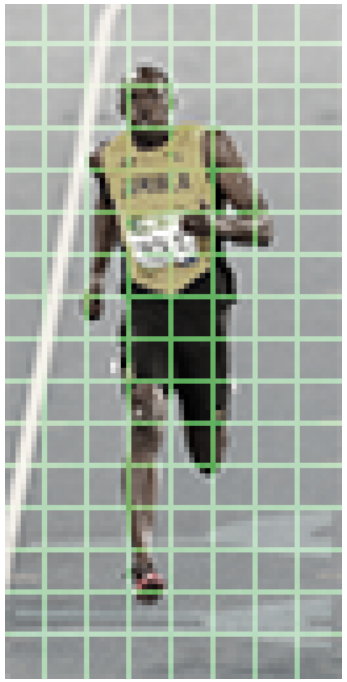
Step -3: Block Normalization:

- 16 X 16 pixels blocks or 2X2 cells are used for normalization, which has 4 histograms.
- Normalization will make it scale/multiplication invariant
- Each block will represent 36 X 1 element vector

Reference: https://tanasecucliciu.wordpress.com/2016/06/08/programming-histogram-of-oriented-gradients-hog-explained/
https://www.learnopencv.com/histogram-of-oriented-gradients/

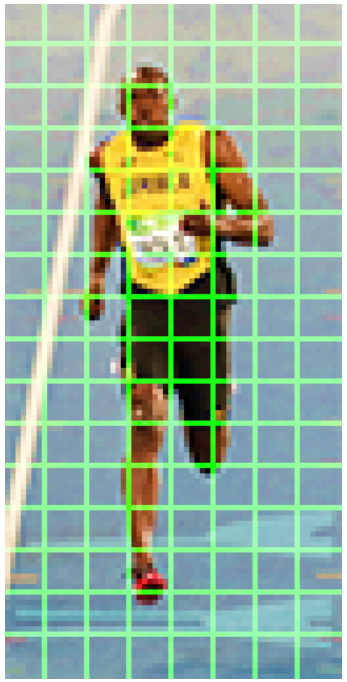
HoG (Histogram of Oriented Gradient)

Step -3: Block Normalization:



Brightness reduced

Original image



Brightness increased

Normalization example:

(3, 9) $\rightarrow\sqrt{3^2 + 9^2} = 9.48$

(3/9.48 , 9/9.48) = (0.32, 0.95)

Multiple (3, 9) by 2 to increase brightness

(6, 18) $\rightarrow\sqrt{6^2 + 18^2}$ = 18.97

(6/18.97, 18/18.97) = (~0.32, ~0.95)

Reference: https://tanasecucliciu.wordpress.com/2016/06/08/programming-histogram-of-oriented-gradients-hog-explained/
https://www.learnopencv.com/histogram-of-oriented-gradients/

HoG (Histogram of Oriented Gradient)

Step -4: Calculate the HOG feature vector:

- Each of the 36 X 1 vectors in each blocks are concatenated into one big vector.
- Size of the vector will be:

  Number of blocks X 36

Example: For an Image size: 64 X 128, will have 8 X 16 cells,

and 7 X 15 block (with 50% overlap),

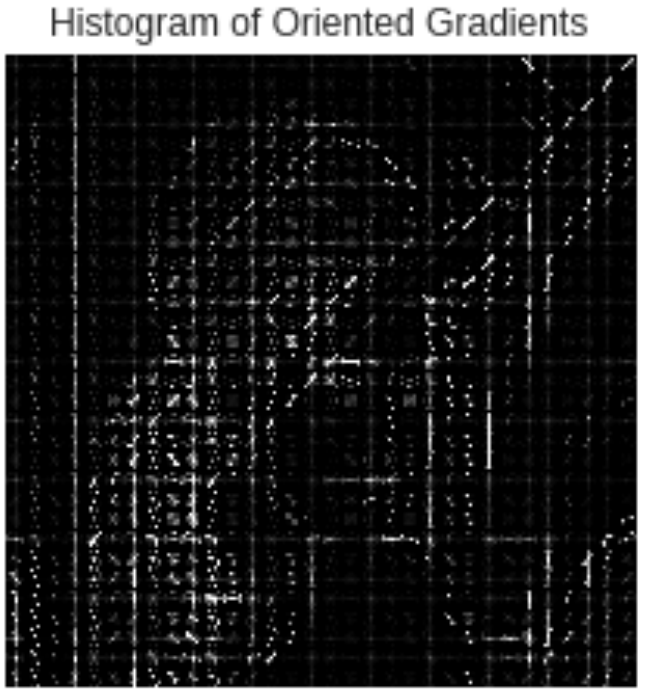
hence size of HOG feature vector: 7 X 15 X 36 = 3,780

Reference: https://tanasecucliciu.wordpress.com/2016/06/08/programming-histogram-of-oriented-gradients-hog-explained/
https://www.learnopencv.com/histogram-of-oriented-gradients/

HoG (Histogram of Oriented Gradient)

Example:

```
from skimage.feature import hog
from skimage import data, color, exposure
import cv2

import matplotlib.pyplot as plt
image = cv2.imread('new_image.png')
image = color.rgb2gray(image)

fd, hog_image = hog(image, orientations=8, pixels_per_cell=(16, 16),
                    cells_per_block=(1, 1), visualise=True)

plt.figure(figsize=(8, 4))

plt.subplot(121).set_axis_off()
plt.imshow(image, cmap=plt.cm.gray)
plt.title('Input image')

# Rescale histogram for better display
hog_image_rescaled = exposure.rescale_intensity(hog_image, in_range=(0, 0.02))

plt.subplot(122).set_axis_off()
plt.imshow(hog_image_rescaled, cmap=plt.cm.gray)
plt.title('Histogram of Oriented Gradients')
plt.show()
```



Visualisation of the histogram
(Magnitude and direction)

Reference: http://scikit-image.org/docs/0.6/auto_examples/plot_hog.html

Local Binary Pattern (LBP)

- An efficient texture operator which labels each pixels of an image by thresholding their neighbours.

- A powerful feature for texture classification

- The idea behind the LBP operator is to describe the image textures using two measures namely, local spatial patterns and the gray scale contrast of its strength.

Reference: https://ieeexplore.ieee.org/abstract/document/6889906

Local Binary Pattern (LBP)

- The basic $LBP_{P,R}$ operator is defined as follows:

$$LBP_{P,R}(x_c, y_c) = \sum_{p=0}^{P-1} S(g_p - g_c)2^P$$

$$S(x) = \begin{cases} 1, if x >= 0 \\ 0, otherwise \end{cases}$$

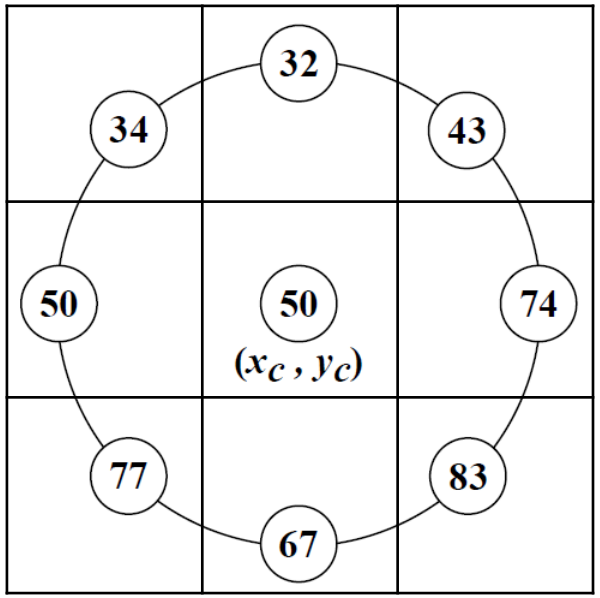
Where,

S(x) → a thresholding function
(x_c , y_c) → the centre pixel in the 8 pixel neighbourhood,
g_c → gray level of the centre pixel
g_p → gray value of a sampling point in an
equally spaced circular neighbourhood of P sampling points
and radius R around the point (x_c , y_c)

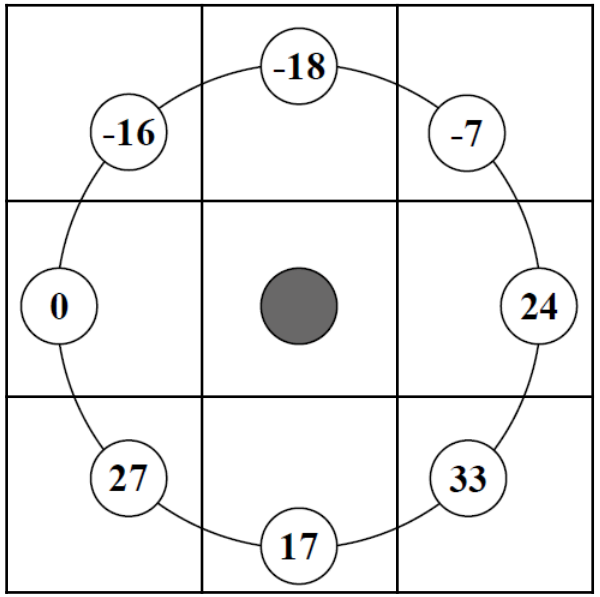
Reference: https://ieeexplore.ieee.org/abstract/document/6889906

Local Binary Pattern (LBP)

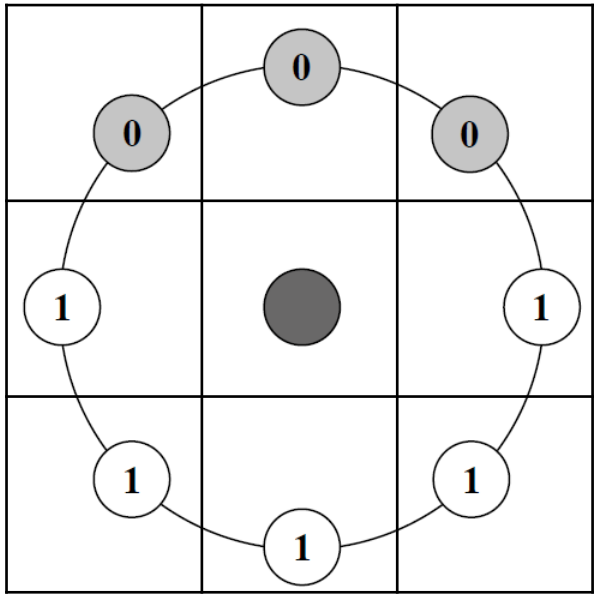
An Example of LBP Computation:



(a) Sample pixel neighbourhood



(b) Difference result



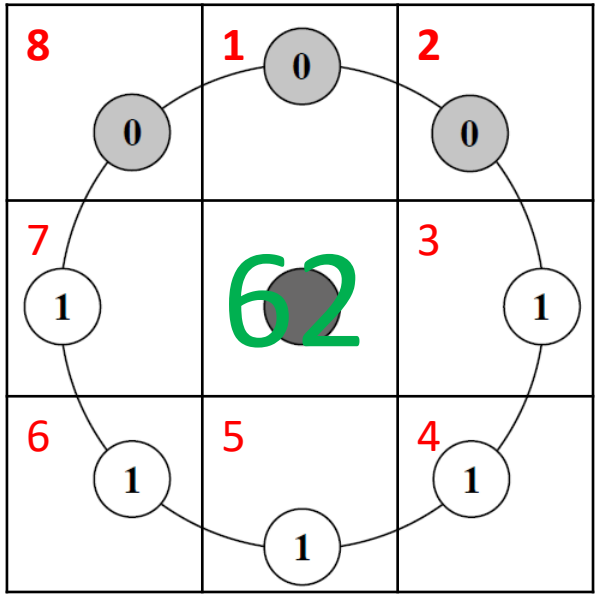
(c) Thresholding result

Reference: https://ieeexplore.ieee.org/abstract/document/6889906

Local Binary Pattern (LBP)

An Example of LBP Computation:

An 8-digit binary number is obtained by considering the thresholding result, starting from pixel 1 to 8, as marked in red.



- There can be $2^8 = 256$ possible values
- Hence, the LBP histogram will have **256 bins →feature vector**

00111110 =
$(0 \times 2^7) + (0 \times 2^6) + (1 \times 2^5) + (1 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) = 62$

Reference: https://ieeexplore.ieee.org/abstract/document/6889906

Local Binary Pattern (LBP)

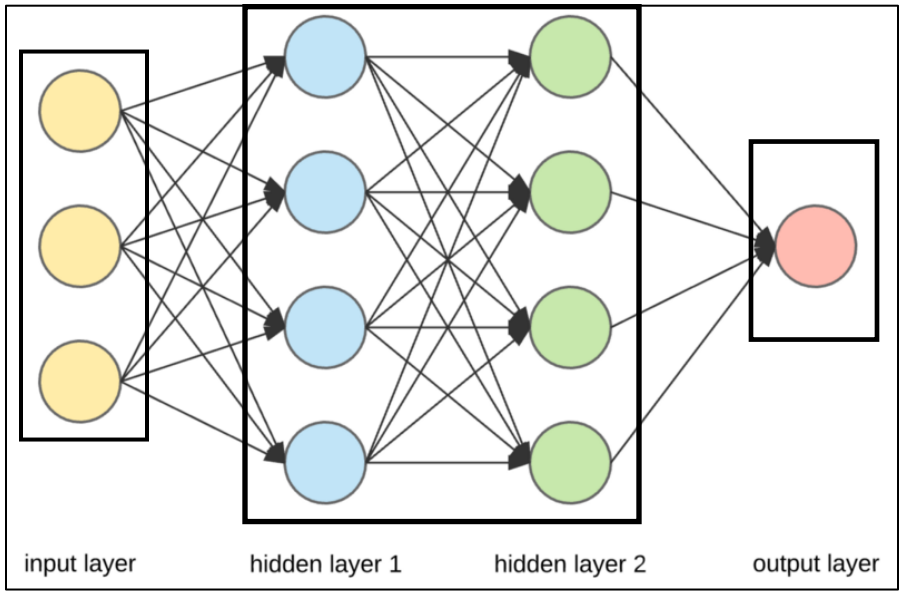
An Example of LBP Computation:

Reference: https://ieeexplore.ieee.org/abstract/document/6889906

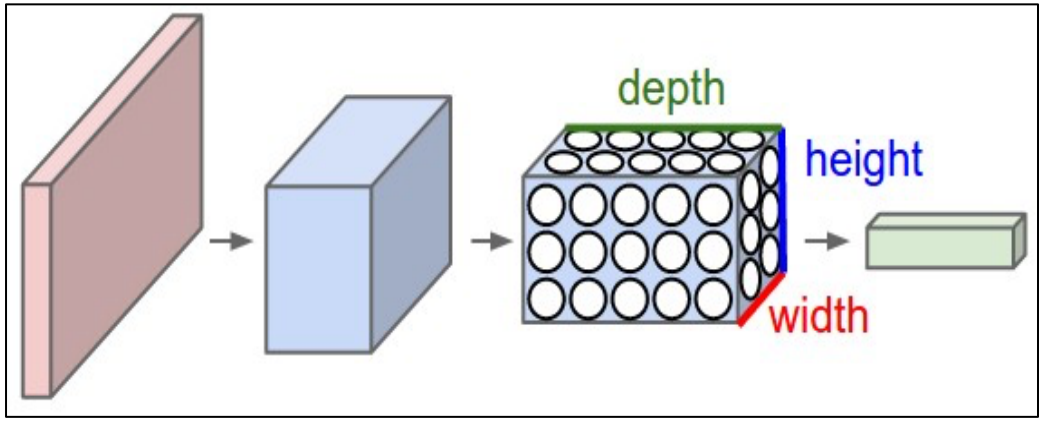
Neural Network Basics

What is Artificial Neural Network (ANN)?

- Artificial Neural Networks (ANN) are multi-layered fully-connected neural networks.
- It has an input layer, multiple hidden layers and an output layer



Standard ANNs

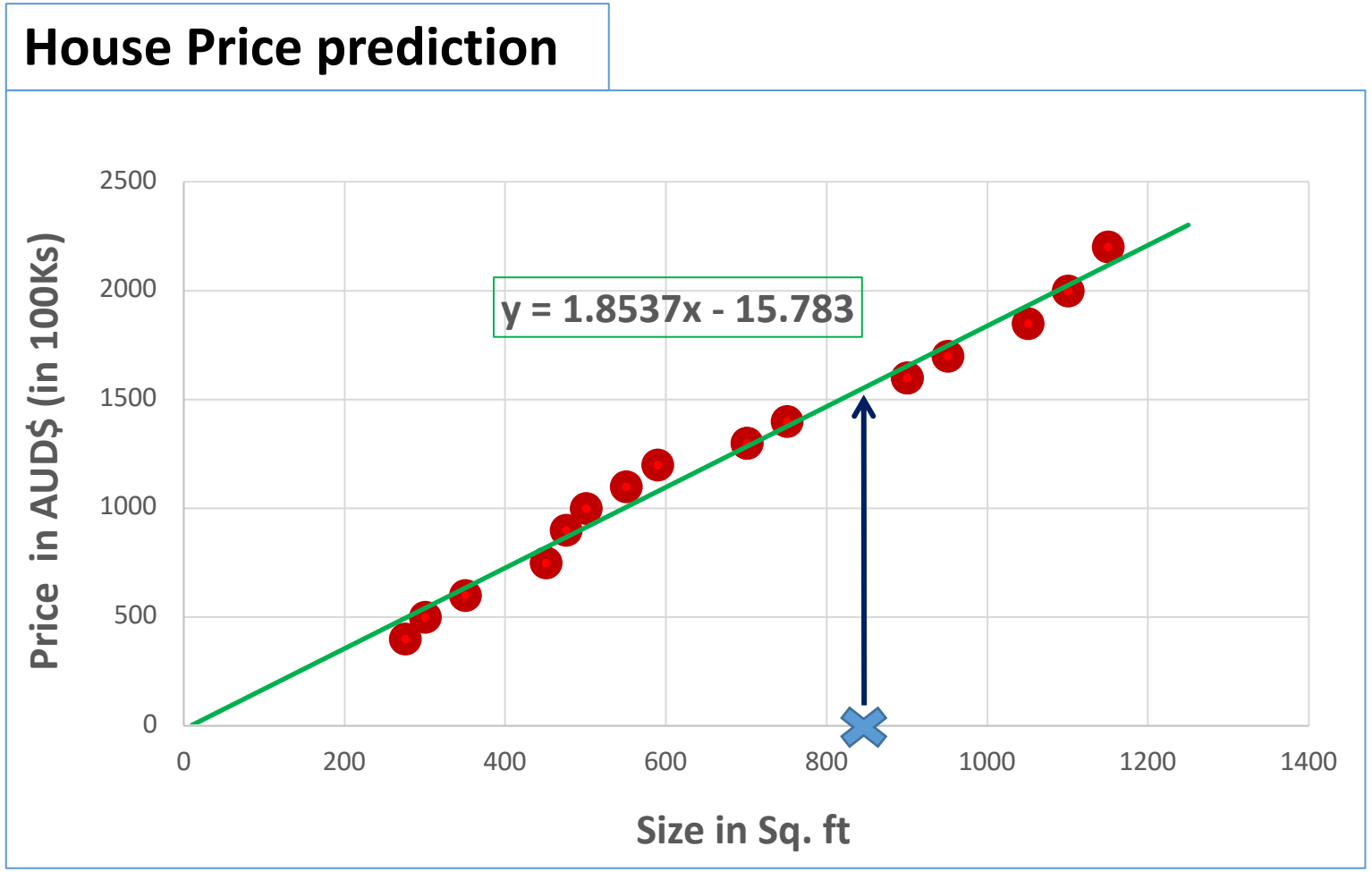


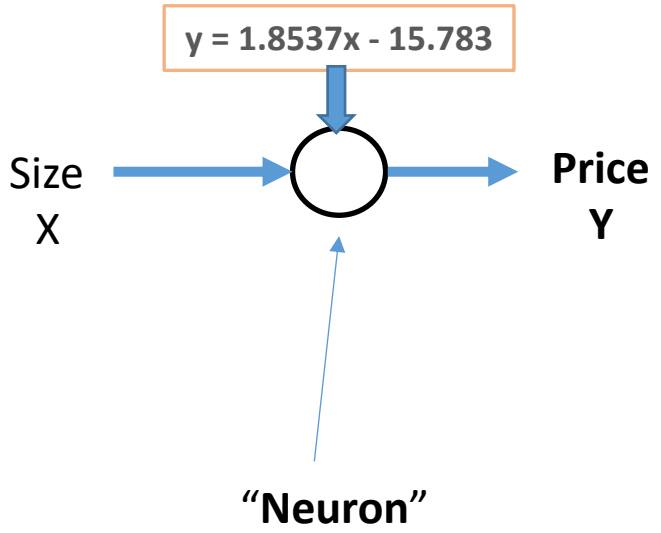
CNNs

Reference & Image source: https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6

Image source: http://cs231n.github.io/assets/cnn/cnn.jpeg

ANN Introduction





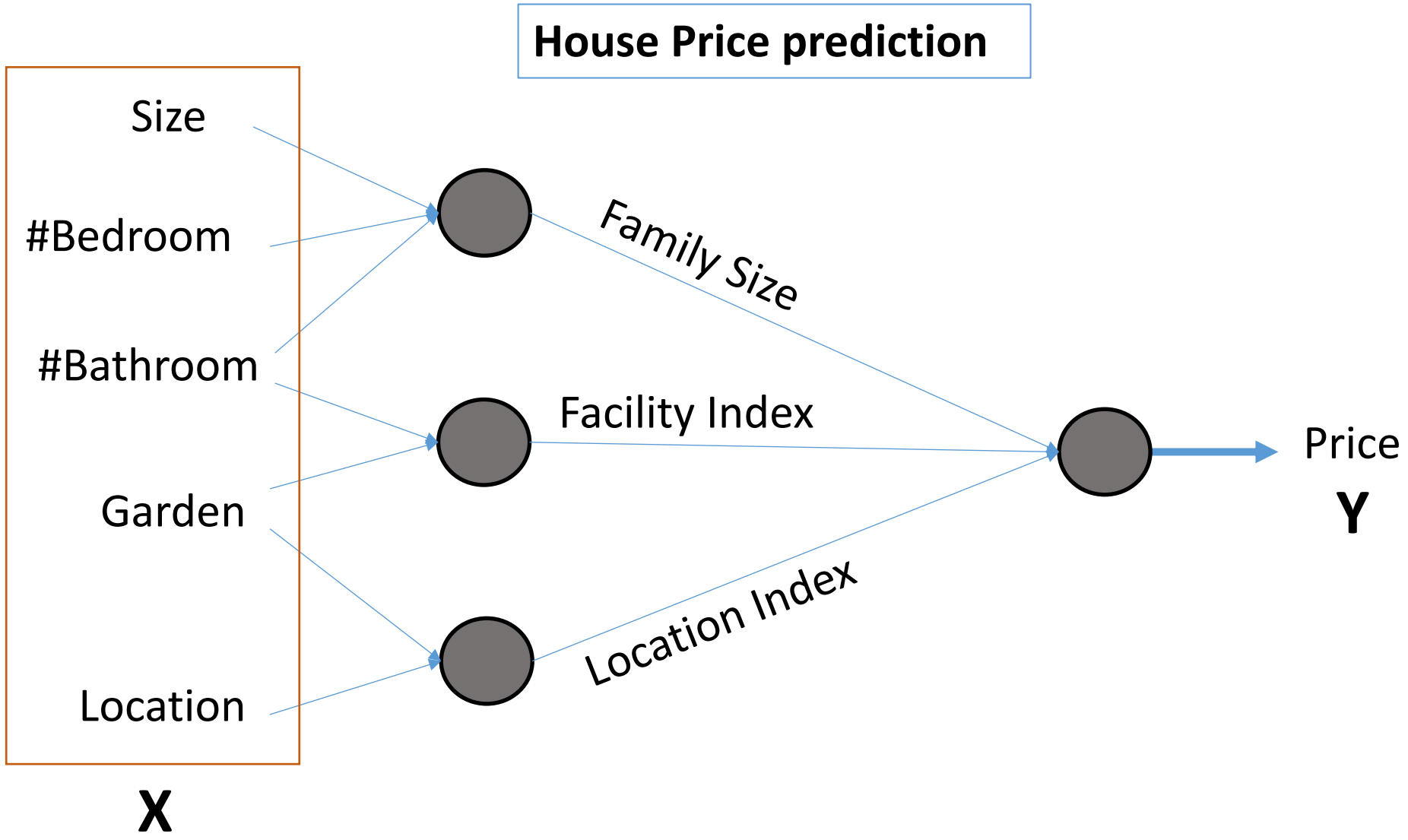
ANN Introduction

House Price prediction

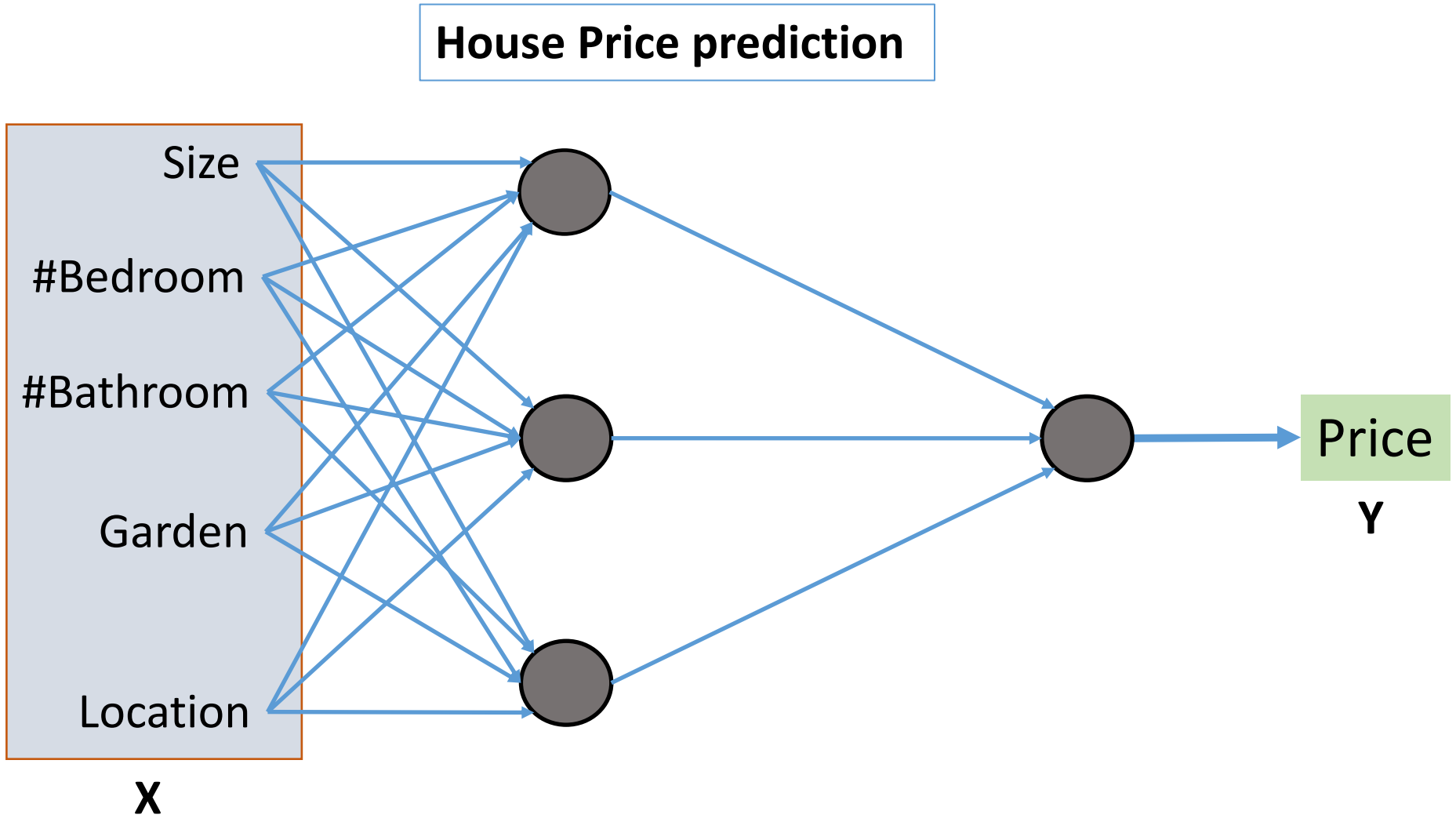
	VARIABLES USED
QUALITY INDEX	Floors, windows type, kitchen furniture and reformations
FACILITIES INDEX	Pool, tennis, garden
BUILDING INDEX	Age, lift, laundry
EXTERNAL DATA INDEX	Orientation, terrace
EXTRAS INDEX	Garage, storage
LOCATION INDEX	Geographical position within the city

Source and Reference: https://www.econstor.eu/bitstream/10419/113851/1/756619068.pdf

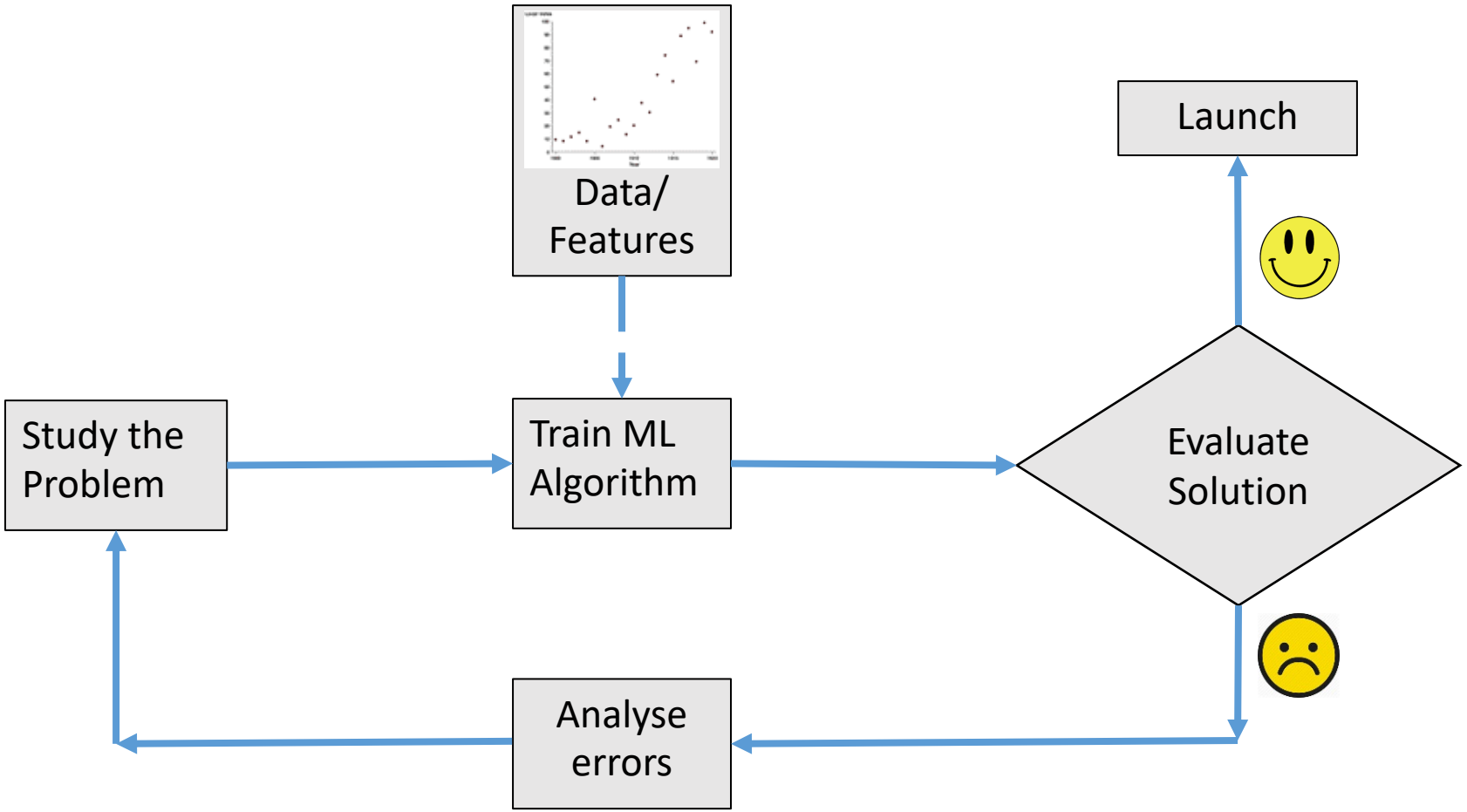
ANN Introduction



ANN Introduction



ANN Introduction – Learning Process



ANN Introduction – Learning Process

Problem of Binary Classification:
Shark ? → 1
Not Shark? → 0

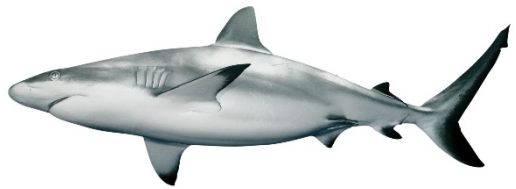
Function to calculate the loss/error

Mechanism to reduce the loss in the model

Gradient

Error/ Loss

1
Target



Input (x)

Model
ANN Architecture
+
Parameters

0

Output (y)

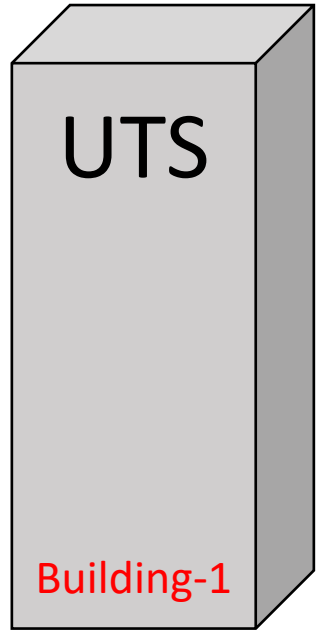
ANN Introduction – Learning Process: Example

S

Position: (x, y)

Position: (x+dx, y+dx)

Target



d

D

Distance need to cover
to reach target

Distance remaining = (D – d)

(Error/Loss to minimize)

Update Position (parameter):

x = x + dx

y = y + dy

Logistic Regression as Neural Network

Problem of Binary Classification → Logistic Regression (Shark ? → 1 | Not Shark? → 0)

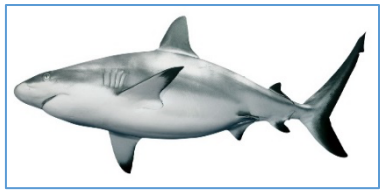
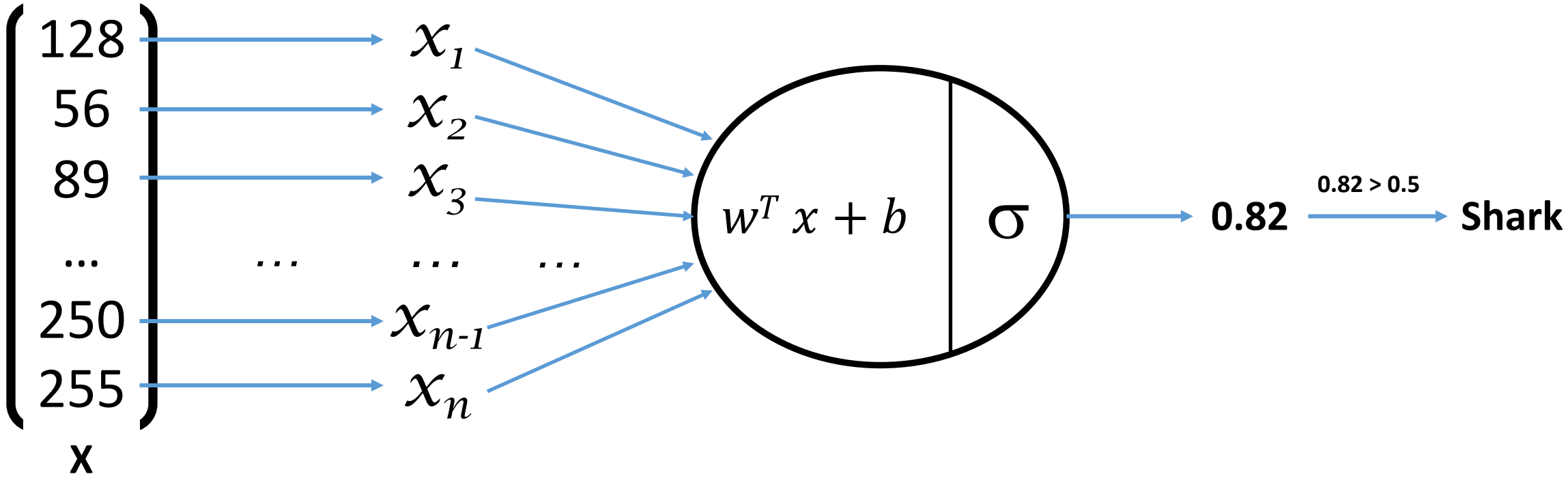


Image dimension:
64X128 = **8192 Pixels**

image.reshape(image.shape[0]*image.shape[1]*image.shape[2],1)



Source and Reference: http://cs230.stanford.edu/files/C1M2.pdf

Logistic Regression as Neural Network

Problem of Binary Classification → Logistic Regression (Shark ? → 1 | Not Shark? → 0)

$w^T x + b$ → Linear function of input *x*

$w_1x_1 + w_2x_2 + \ldots + wnxn + b$

Weighted sum of inputs

Where,

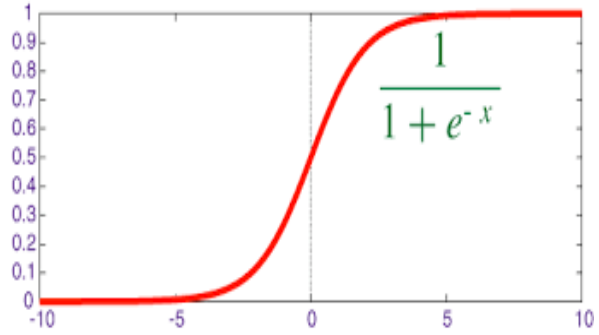
W → Weights

X → Inputs

b → Bias term

σ → Activation function

$\sigma =$ (Sigmoid function)



Rule of thumb:
In case of binary classification, Sigmoid function is the obvious choice for output layer

Source and Reference: http://cs230.stanford.edu/files/C1M2.pdf

Logistic Regression as Neural Network

Problem of Binary Classification → Logistic Regression (Shark ? → 1 | Not Shark? → 0)

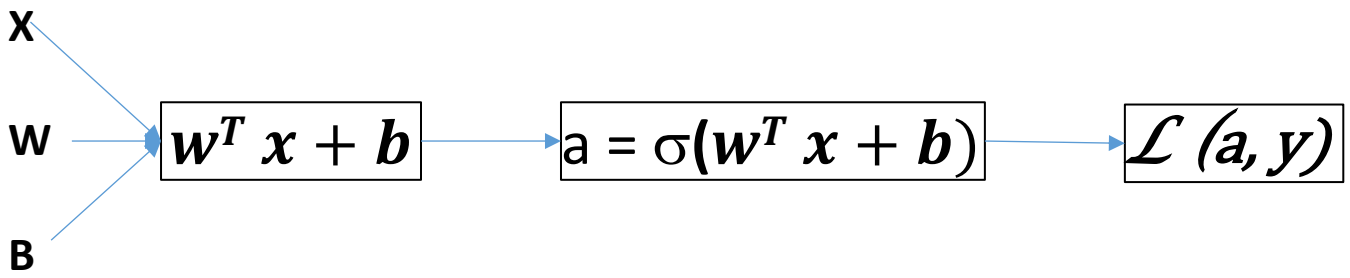
Parameters:

1. **w (weight)**
2. **b (bias)**
3. **Output a = $\sigma(w^T x + b)$**

Loss function for Logistic Regression:

$$\mathcal{L}(a, y) = -(y \log a + (1 - y)\log(1 - a))$$

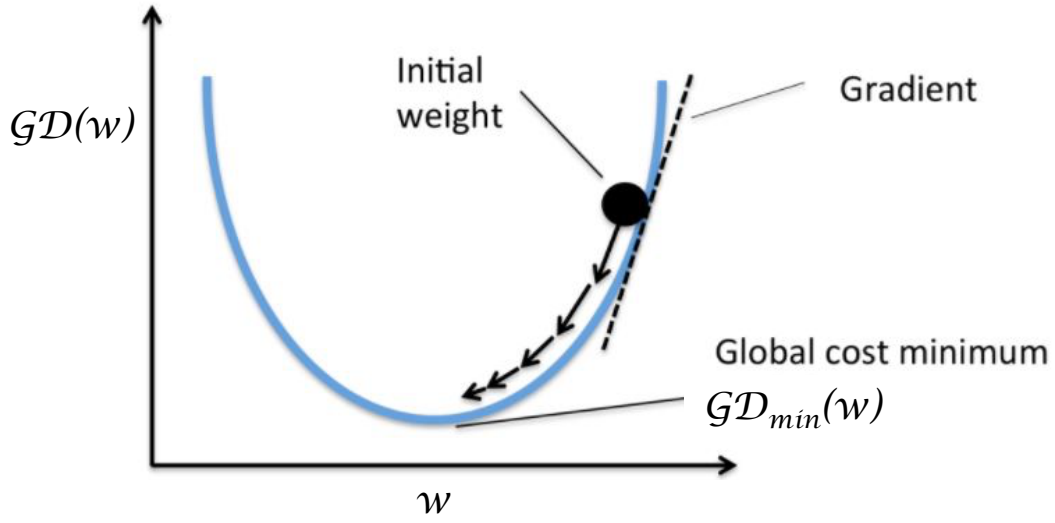
Logistic Regression pipeline with the math looks like:



Logistic Regression as Neural Network

Problem of Binary Classification → Logistic Regression (Shark ? → 1 | Not Shark? → 0)

Gradient Descent for learning parameters:
It is an iterative approach for error correction in a machine learning model.



For 1 Sample the loss function is:

$$\mathcal{L}\ (\mathrm{a}, \mathrm{y}) = -(y \log a + (1 - y)\log(1\ - a))$$

For m Sample the loss function is:

$$\mathcal{GD}(w,\ b) = x = \frac{1}{m}\sum_{\mathrm{i}=1}^{\mathrm{m}} \mathcal{L}\ (a, \mathrm{y})$$

Question: Find w and b that will minimize $\mathcal{GD}(w,\ b)$

Image Source: https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781788397872/1/ch01lvl1sec22/gradient-descent
Source and Reference: http://cs230.stanford.edu/files/C1M2.pdf

Logistic Regression as Neural Network

Problem of Binary Classification → Logistic Regression (Shark ? → 1 | Not Shark? → 0)

Gradient Descent for learning parameters:
It is an iterative approach for error correction in a machine learning model.

Updating the ***w*** and ***b*** iteratively, :
$w = w - \alpha dw$

Updating the ***b***:
$b = b - \alpha db$

Where,

$dw = \frac{\partial GD(w,b)}{\partial w}$

$db = \frac{\partial GD(w,b)}{\partial b}$

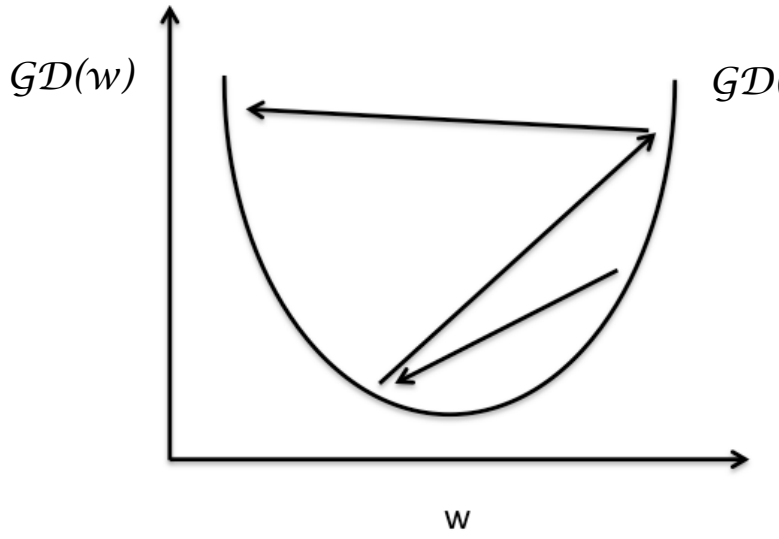
α → Learning rate

Logistic Regression as Neural Network

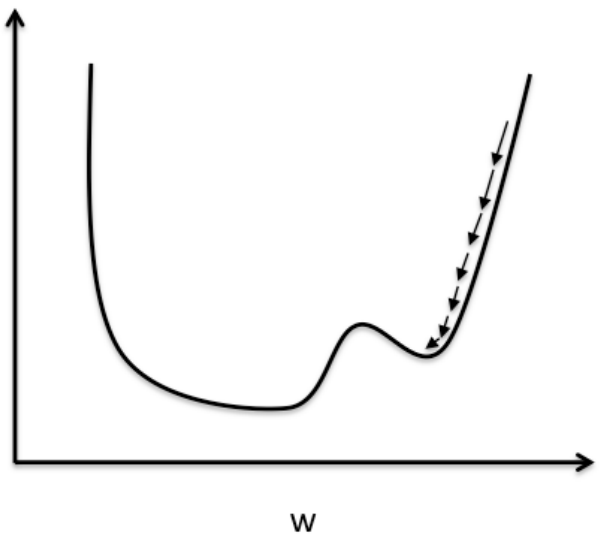
Problem of Binary Classification → Logistic Regression (Shark ? → 1 | Not Shark? → 0)

Gradient Descent for learning parameters:

Learning rate(α) issues:



Large learning rate: Overshooting.

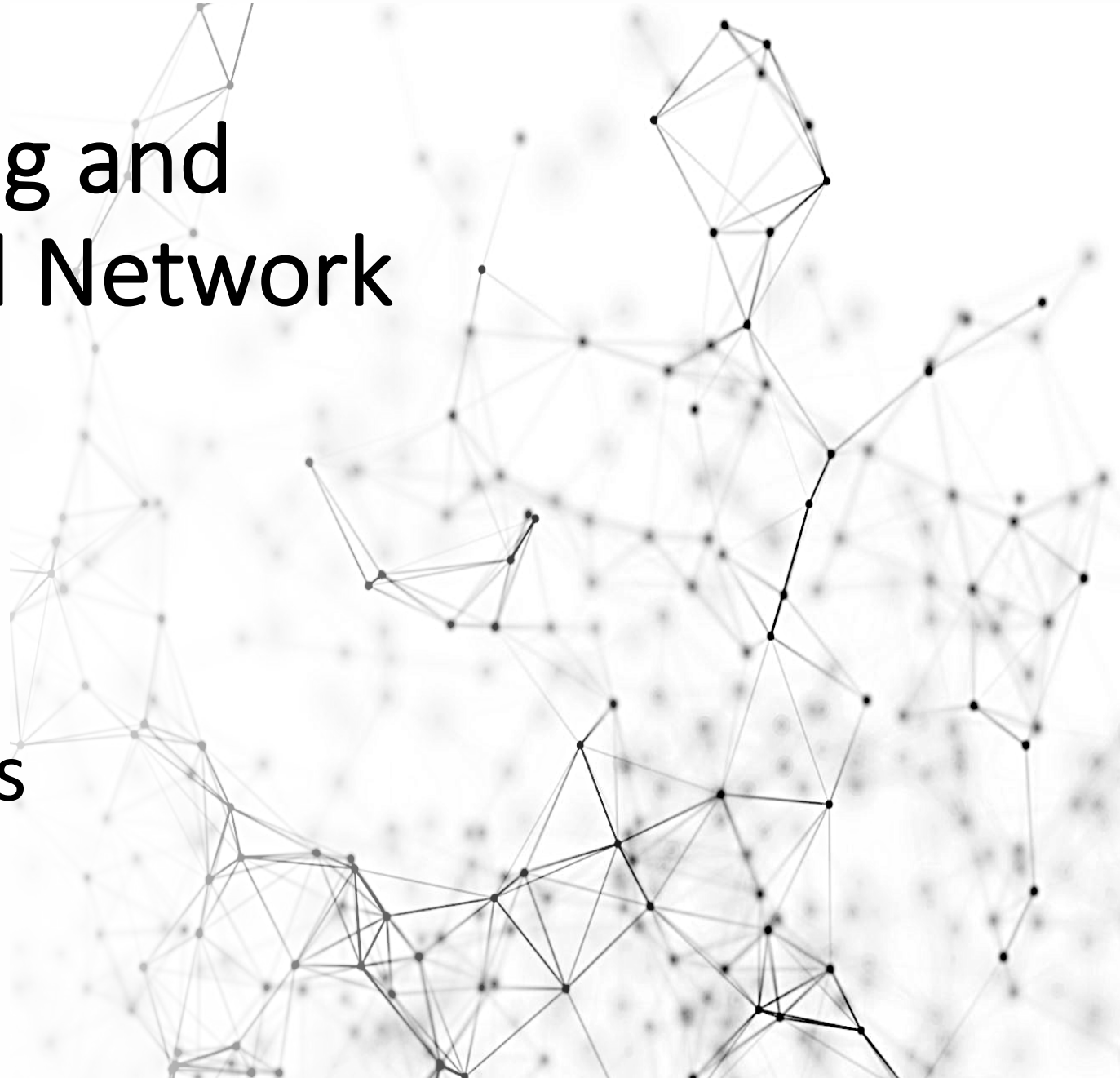


Small learning rate: Many iterations until convergence and trapping in local minima.

42028: Deep Learning and Convolutional Neural Network

Week-4 Lecture

Neural Network in details



Outline

- Logistic Regression Recap
- Back Propagation
- Gradient Descent and intuitions
- Types of Gradient Descent
- Activations Functions
- Logistic Regression with Back Propagation
- Multi-Layered Neural Network

Logistic Regression – Recap

Problem of Binary Classification:
Dog? → 1
Cat (≈ not Dog)? → 0

Input (x)

Model
ANN Architecture
+
Parameters

0
Output (y)

1
Target

Loss/Cost function

Error/ Loss

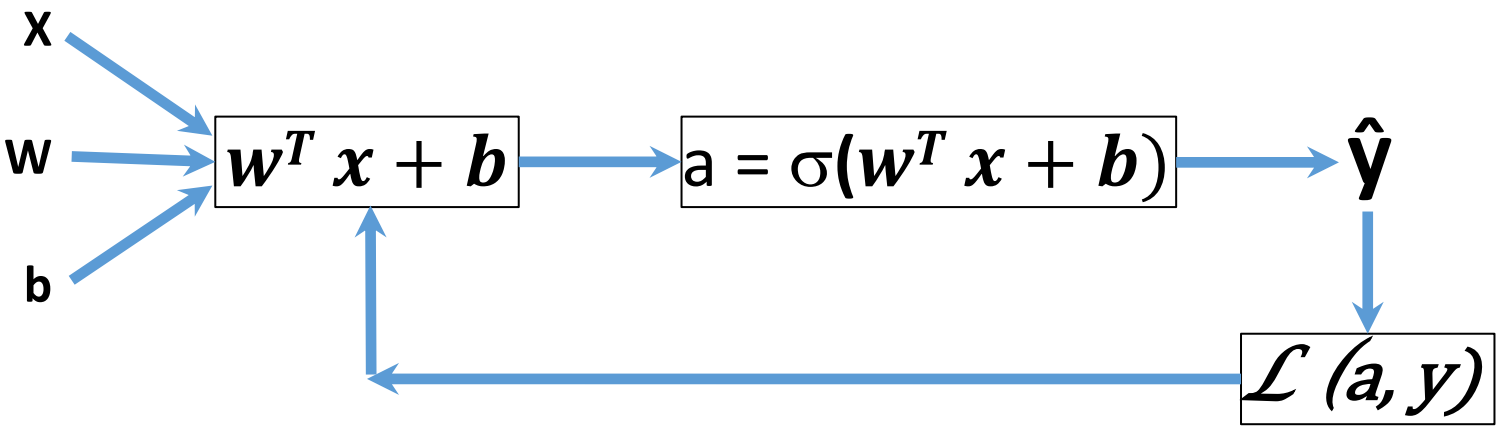
Gradient

Gradient Decent

Activation function

Logistic Regression as Neural Network

Logistic Regression pipeline with the math looks like:



Where,

W → Weights

X → Inputs

b → Bias term

σ → Activation function

Parameters:

1. w (weight)
2. b (bias)
3. Output a = $\sigma(w^T x + b)$

Activation function

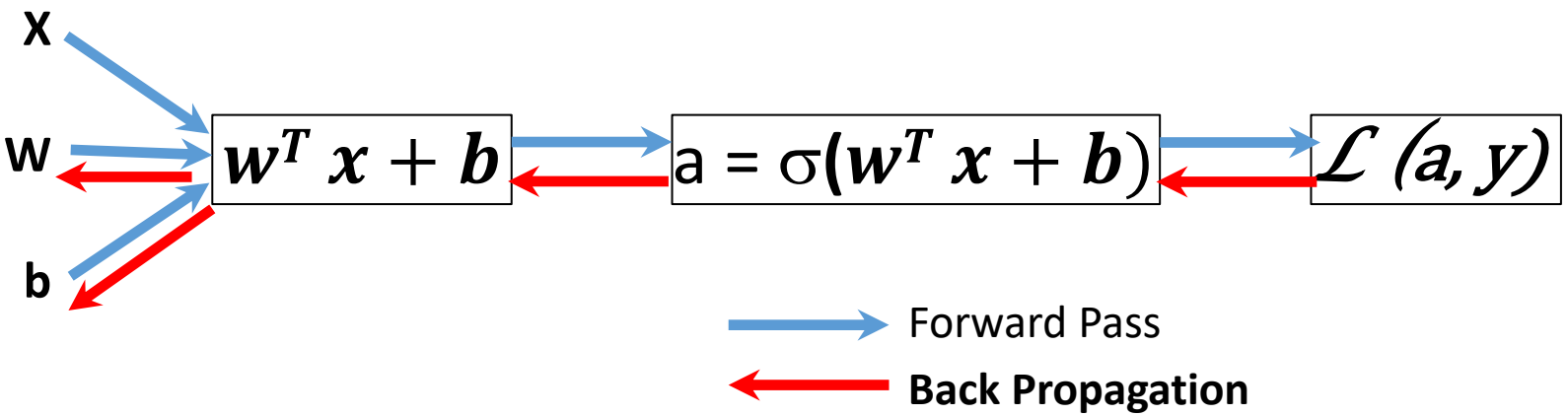
$$a = \sigma = \left(\frac{1}{1+e^{-x}}\right)$$

Loss function for Logistic Regression:

$$\mathcal{L}(a, y) = -(y \log a + (1 - y)\log(1 - a))$$

Logistic Regression as Neural Network

Logistic Regression pipeline with the math looks like:



repeatedly adjust the weights to minimize the difference between actual output and desired output

Where,

W → Weights

X → Inputs

b → Bias term

σ → Activation function

Activation function

$$a = \sigma = \left(\frac{1}{1+e^{-x}}\right)$$

Parameters:

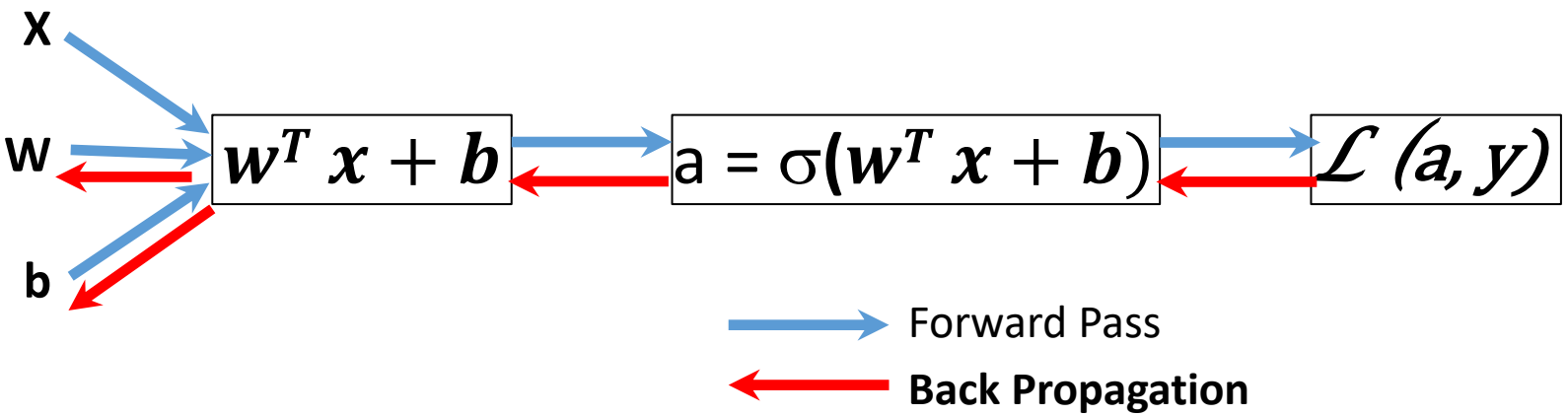
1. w (weight)
2. b (bias)
3. Output a = $\sigma(w^T x + b)$

Loss function for Logistic Regression:

$$\mathcal{L}(a, y) = -(y \log a + (1 - y)\log(1 - a))$$

Logistic Regression as Neural Network

Logistic Regression pipeline with the math looks like:



repeatedly adjust the weights to minimize the difference between actual output and desired output

Where,

W → Weights

X → Inputs

b → Bias term

σ → Activation function

Activation function

$$a = \sigma = \left(\frac{1}{1+e^{-x}}\right)$$

Parameters:

1. w (weight)
2. b (bias)
3. Output a = $\sigma(w^T x + b)$

Loss function for Logistic Regression:

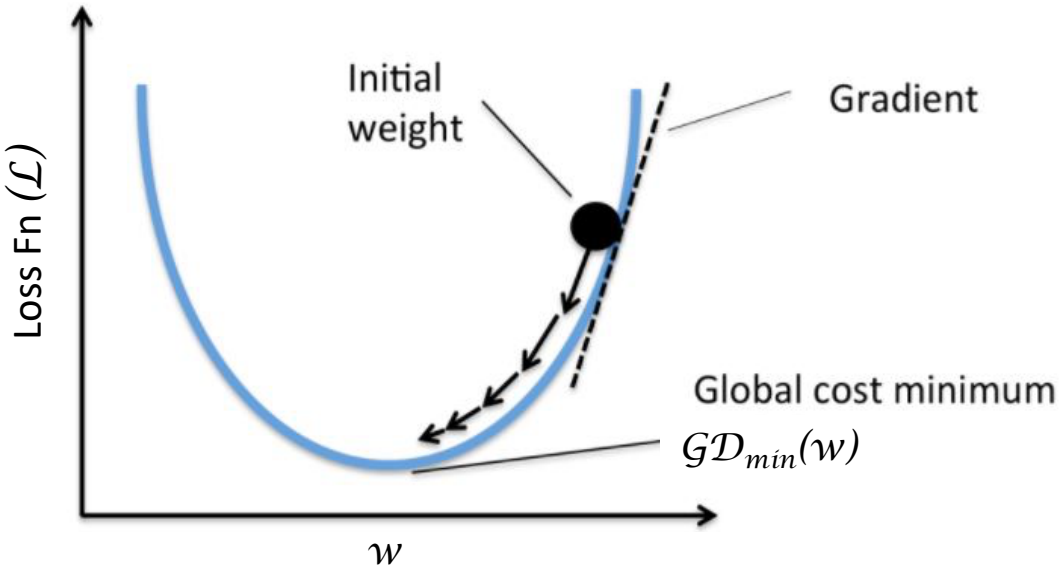
$$\mathcal{L}(a, y) = -(y \log a + (1 - y)\log(1 - a))$$

Gradient Descent

Gradient Descent for learning parameters:
It is an iterative approach for error correction in a machine learning model.

Question: Find w and b that will minimize $\mathcal{GD}(w, b)$

Required: Loss/cost function



Generic Algorithm:

Step 1: Initialize **w** and **b**
Step 2: Perform Forward pass operation/calculations
Step 2: Compute Loss/Cost function $\mathcal{L}(a, y)$
Step 3: Compute change in **w** and **b**
(Take the partial derivative of the cost function with respect to Weights and bias (**dw** and **db**).
Step 4: Update **w** and **b**
w := w – α**dw**
b := b – α**db**
Step 5: Repeat from Step 2 with new values of **w** and **b** for 'n' number of iterations.

Example the loss function is:

$$\mathcal{L}(a, y) = -(y \log a + (1 - y)\log(1 - a))$$

α → Learning rate

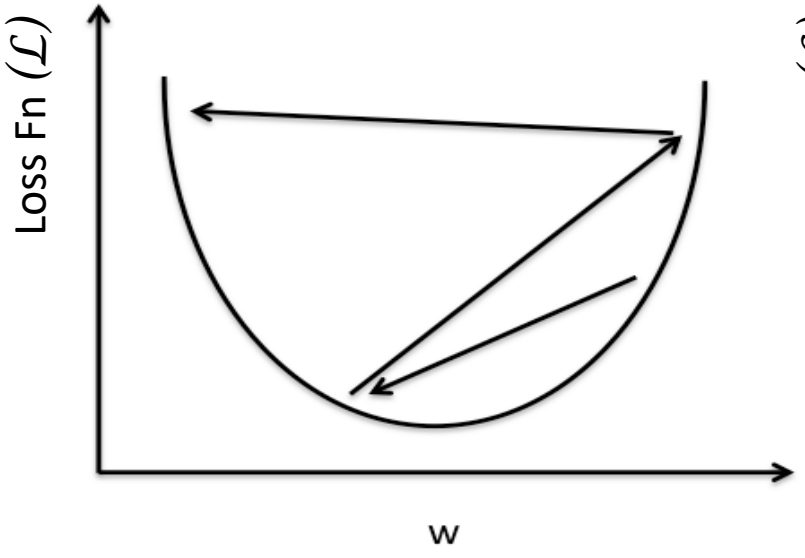
Image Source: https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781788397872/1/ch01lvl1sec22/gradient-descent
Source and Reference: http://cs230.stanford.edu/files/C1M2.pdf

Gradient Descent

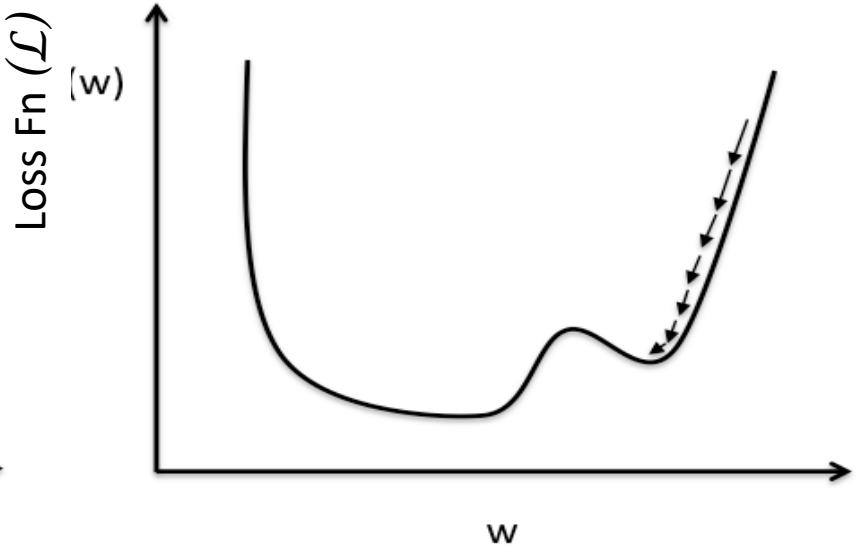
Gradient Descent for learning parameters:

Learning rate(α) issues:

- It is a ***hyper-parameter***



Large learning rate: Overshooting.



Small learning rate: Many iterations until convergence and trapping in local minima.

Image Source: https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781788397872/1/ch01lvl1sec22/gradient-descent
Source and Reference: http://cs230.stanford.edu/files/C1M2.pdf

Gradient Descent

Learning rate(α): more intuitions

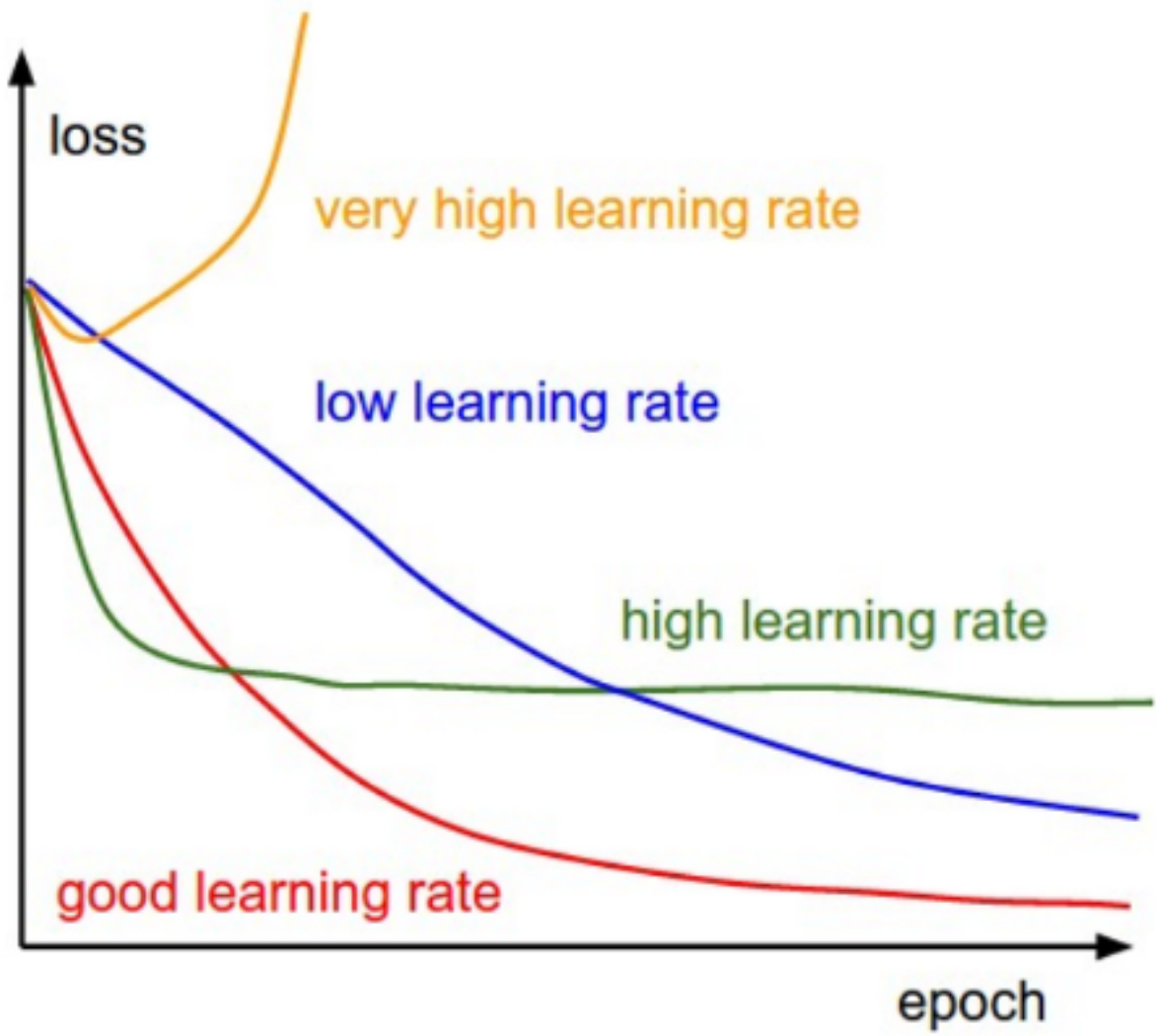


Image Source: http://cs231n.github.io/neural-networks-3/

Gradient Descent Types

There are three main types of Gradient Descent Algorithms:

1. Batch Gradient Descent (BGD)

2. Stochastic Gradient Descent (SGD)

3. Mini-Batch Gradient Descent (MBGD)

Batch Gradient Descent (BGD)

Generic steps:

-Process each input sample and find the cost

-Find the average cost over all input samples

-Update **w** and **b**, and

-repeat the steps for ‘n’ epochs(iterations)

Issues:

1. It uses the complete dataset to calculate the gradients at every steps
2. Slow when training set is large
3. Difficult to find the learning rate
4. Difficult to ascertain the number of epochs(iterations)

Stochastic Gradient Descent (SGD)

Stochastic → Random

Due to the random nature, the Algorithm is much less regular than BGD

Generic steps:

-Process a random input sample and find the cost

-Update **w** and **b**, and

-repeat the steps for ‘n’ iterations on the training samples

Advantage:

1. Computes gradient based on single input sample: memory efficient
2. Much faster compared to BGD
3. Possible to train on large dataset
4. Randomness is a good escape from local minima problem

Issues:

1. Might not reach the optimal value, but very close to it.

Stochastic Gradient Descent (SGD)

Issues: Might not reach the optimal value, but very close to it.

Possible solution: Reduce the learning rate gradually → Stimulated annealing

Create a Learning Schedule to determine the learning at each iteration.

Epoch: One round through the complete training set.

Iterations: Process in multiple subsets of the training set, say, 'm' iterations my form 1 epoch

Mini-Batch Gradient Descent (MBGD)

Generic steps:

-Divide the training set into mini-batches (set of random samples on fixed number)

-Process all the samples in a Mini-batch and find the average cost

-Update **w** and **b**, and

-repeat the steps for ‘n’ iterations/epochs on the training samples

Advantage:

1. Computes gradient based on small sets of input sample
2. Much faster compared to BGD
3. Possible to train on large dataset
4. Performance boost on matrix operations using GPUs!
5. Might not reach the optimal value, but very close to it, and possibly better than SGD

Issues:

1. It may be harder to escape the local minima.

Gradient Descent (SGD) - intuition

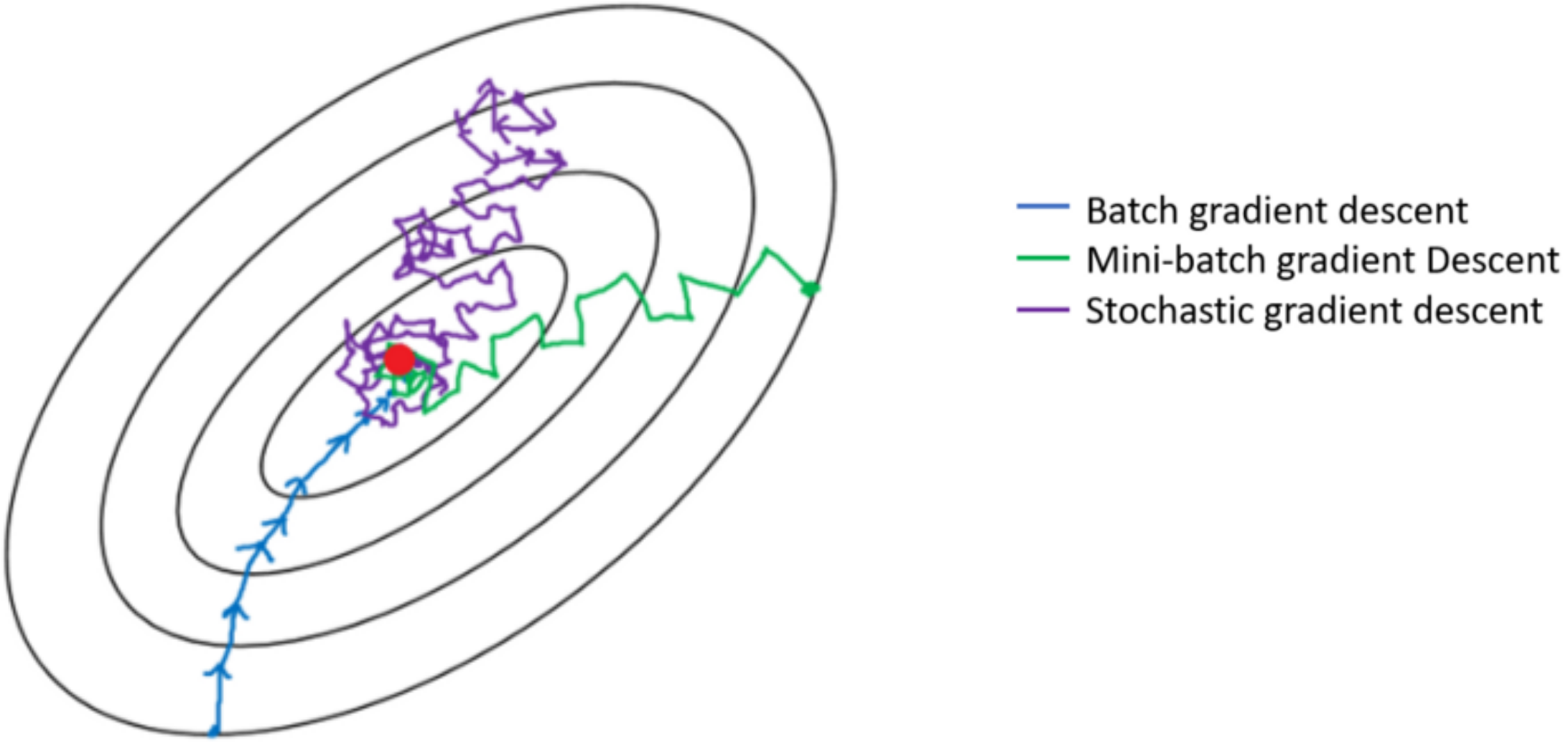


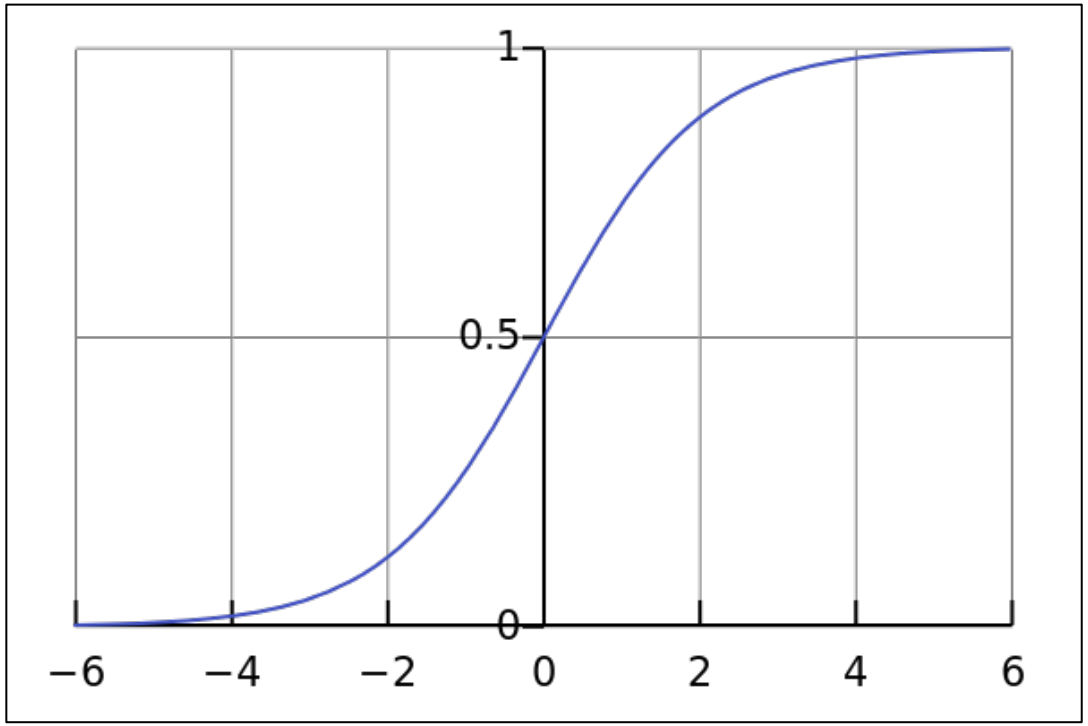
Image Source: https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3

More Activation Functions

Sigmoid function: $\sigma = \left(\frac{1}{1+e^{-x}}\right)$

Characteristics:

- **Non-linear in nature**
- **Range(0, 1)**
- **Tends to bring the activations to either side of the curve: good for a classifier**
- **Suffers from vanishing gradient problem**



Vanishing Gradient: Towards to the end of the curve, the value of Y change very less to the changes in X values. Hence gradient at the region will be very small. The network will refuse or learning extremely slowly.

Source: https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0

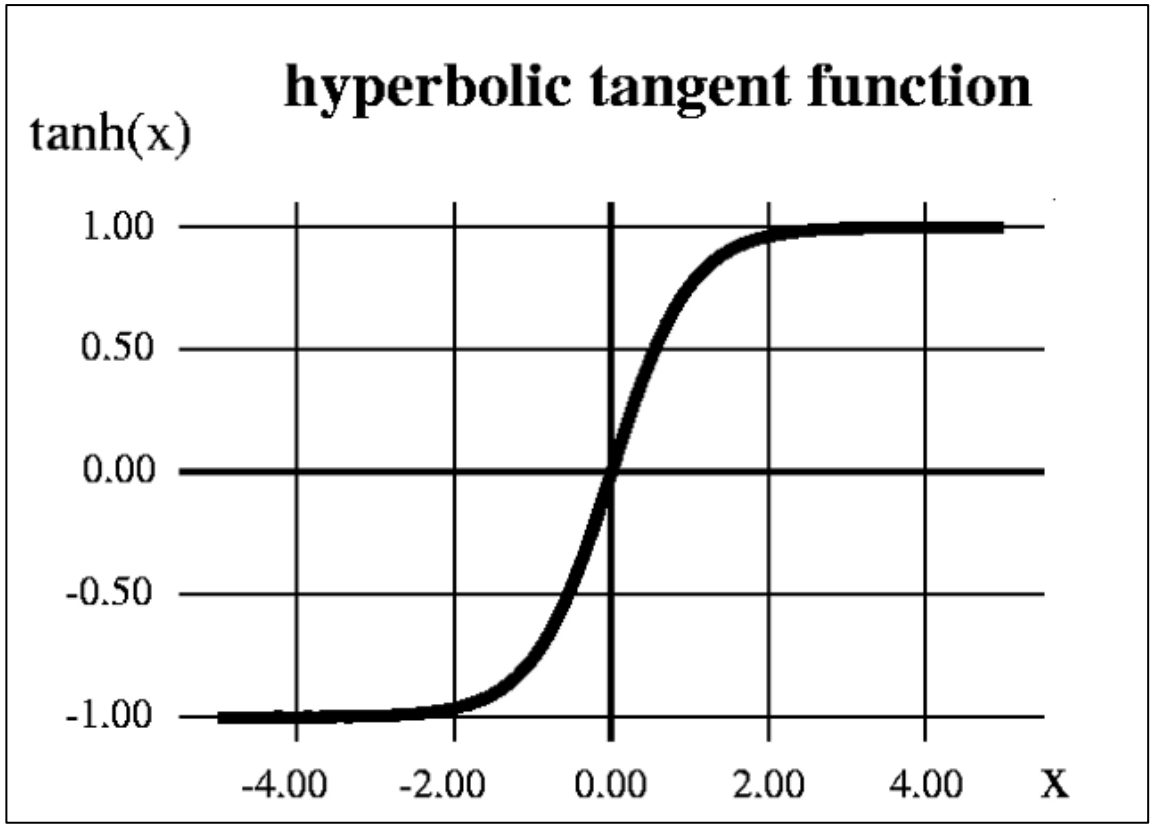
More Activation Functions

Hyperbolic tangent:

$$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$

Characteristics:

- **Non-linear in nature**
- **Range(-1, 1)**
- **Stronger gradient than sigmoid**
- **Also suffers from vanishing gradient problem**



More Activation Functions

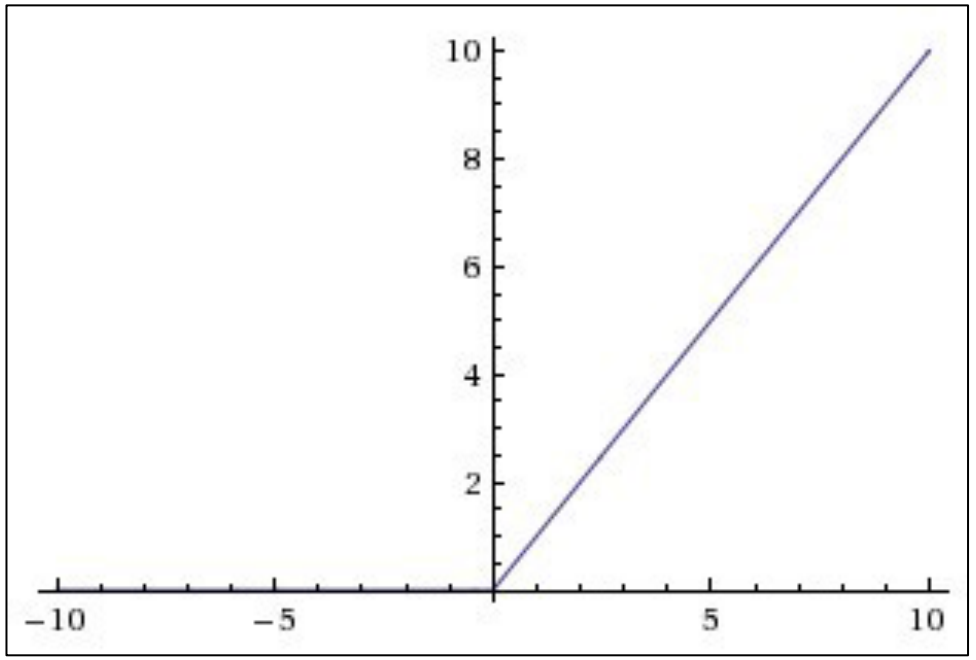
Rectified Linear Unit (ReLu)

$$A(x) = \max(0, x)$$

i.e. : if x < 0, *A(x)* = 0, if x > 0, *A(x)* = x

Characteristics:

- **Non-linear in nature**
- **Range[0, inf]**
- **Stronger gradient than sigmoid**
- **Computationally less expensive than Sigmoid and Tanh**
- **Best used in hidden layers**
- **Dying ReLu problem**



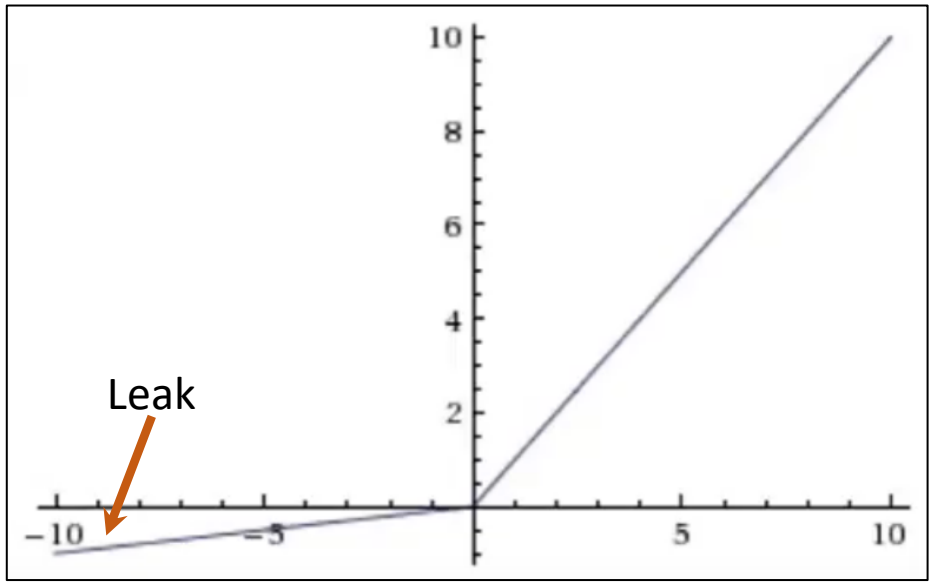
Avoids and rectifies **vanishing gradient** problem

More Activation Functions

Leaky Rectified Linear Unit (Leaky ReLu)

$$A(x) = \max(0.01x, \mathrm{x})$$

i.e. : if x < 0, *A(x)* = 0.01x, if x > 0, *A(x)* = x



Characteristics:

- **Non-linear in nature**
- **Range[0, inf]**
- **Leaky ReLUs are one attempt to fix the "dying ReLU" problem**

Source & Reference : http://cs231n.github.io/neural-networks-1/
https://towardsdatascience.com/activation-functions-in-neural-networks-58115cda9c96

More Activation Functions

Softmax $S(y_i) = \frac{e^{y_i}}{\sum_j (e^{y_j})}$ *for* $j = 1, \ldots, K.$

Characteristics:

- **Non-linear in nature**
- **Turns numbers in probabilities that sum to one.**
- **Useful when we have more than one output**
- **Used for classification in the output layer**
- **Less computationally expensive than Sigmoid and Tanh**

Illustration:

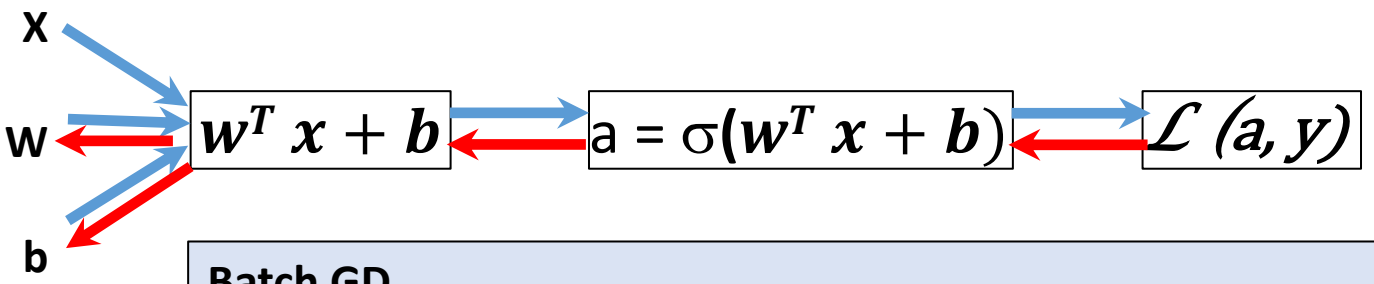
Y = [2.0, 1.0, 0.1]

Softmax(Y) = [0.7, 0.2, 0.1] (approx.)

Reference: https://medium.com/data-science-bootcamp/understand-the-softmax-function-in-minutes-f3a59641e86d

Logistic Regression with Backpropagation

Logistic Regression pipeline with the math looks like:



Batch GD

Step 1: Initialize **w** and **b**

Step 2: Perform Forward pass operation/calculations

Step 2: Compute Loss/Cost function $\mathcal{L}$ (a, y)

Step 3: Find the average cost over all input samples

(Take the partial derivative of the cost function with respect to Weights and bias (**dw** and **db**).

Step 4: Update **w** and **b**

w := w – α**dw**

b := b – α**db**

Step 5: Repeat from Step 2 with new values of **w** and **b** for 'n' number of iterations.

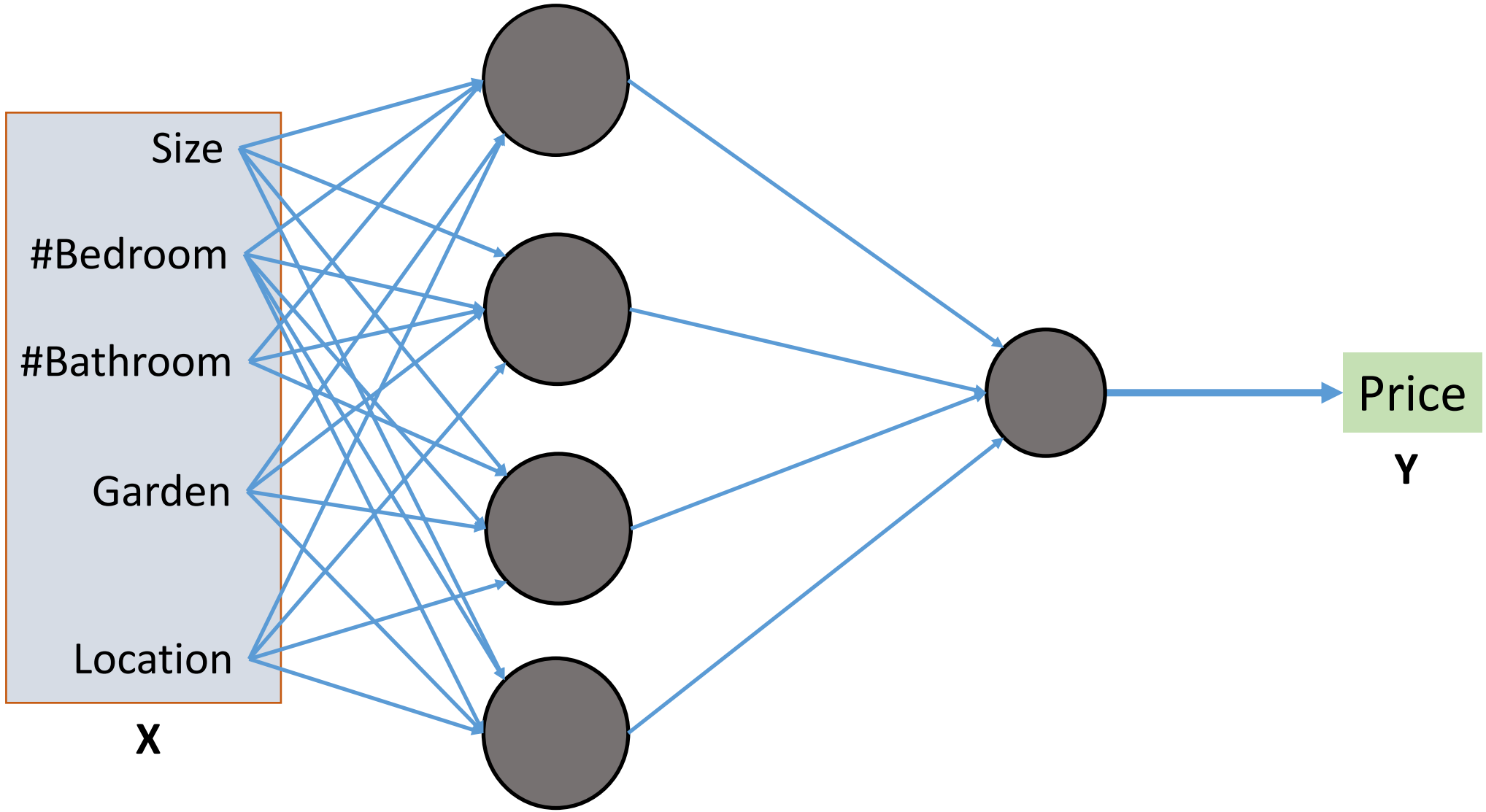
Average cost over all training 'm' samples

$$\text{Avg Loss(J)} = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(a_i, y_i)$$

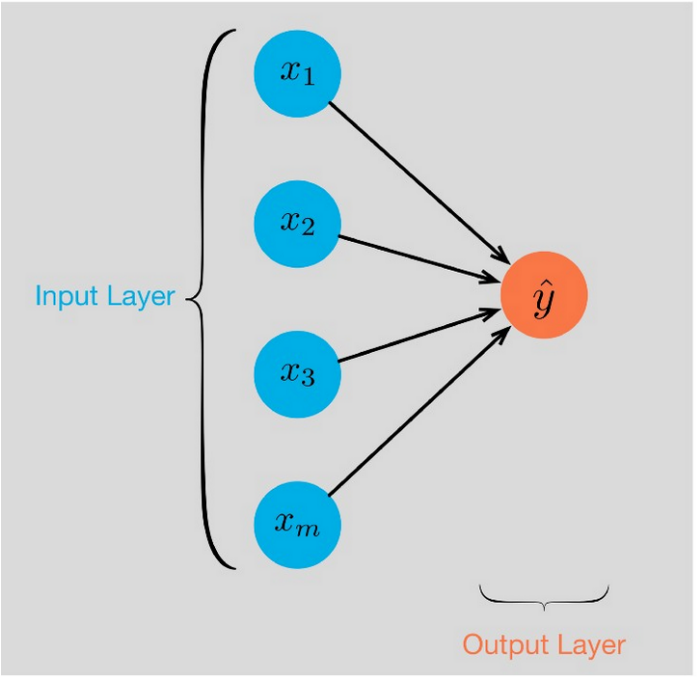
$$\text{dw} = \frac{\partial J}{\partial w}, \qquad \text{db} = \frac{\partial J}{\partial b}$$

$$w := w - \alpha \mathbf{dw}$$
$$b := b - \alpha \mathbf{db}$$

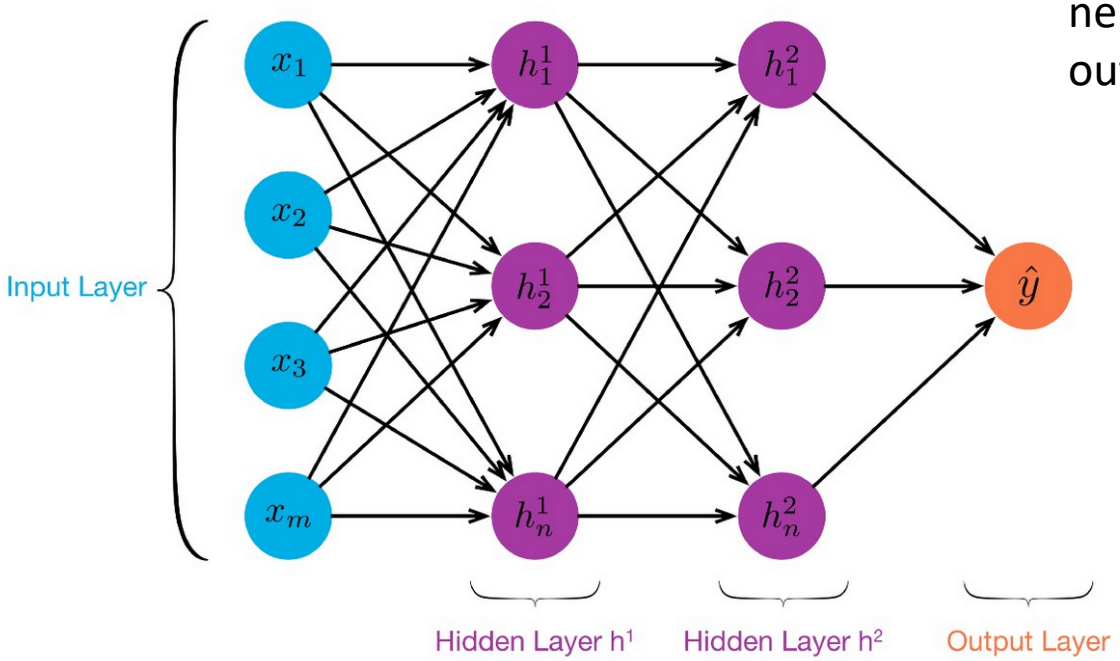
Multi-layer Neural Network



Multi-layer Neural Network



Single layer perceptron



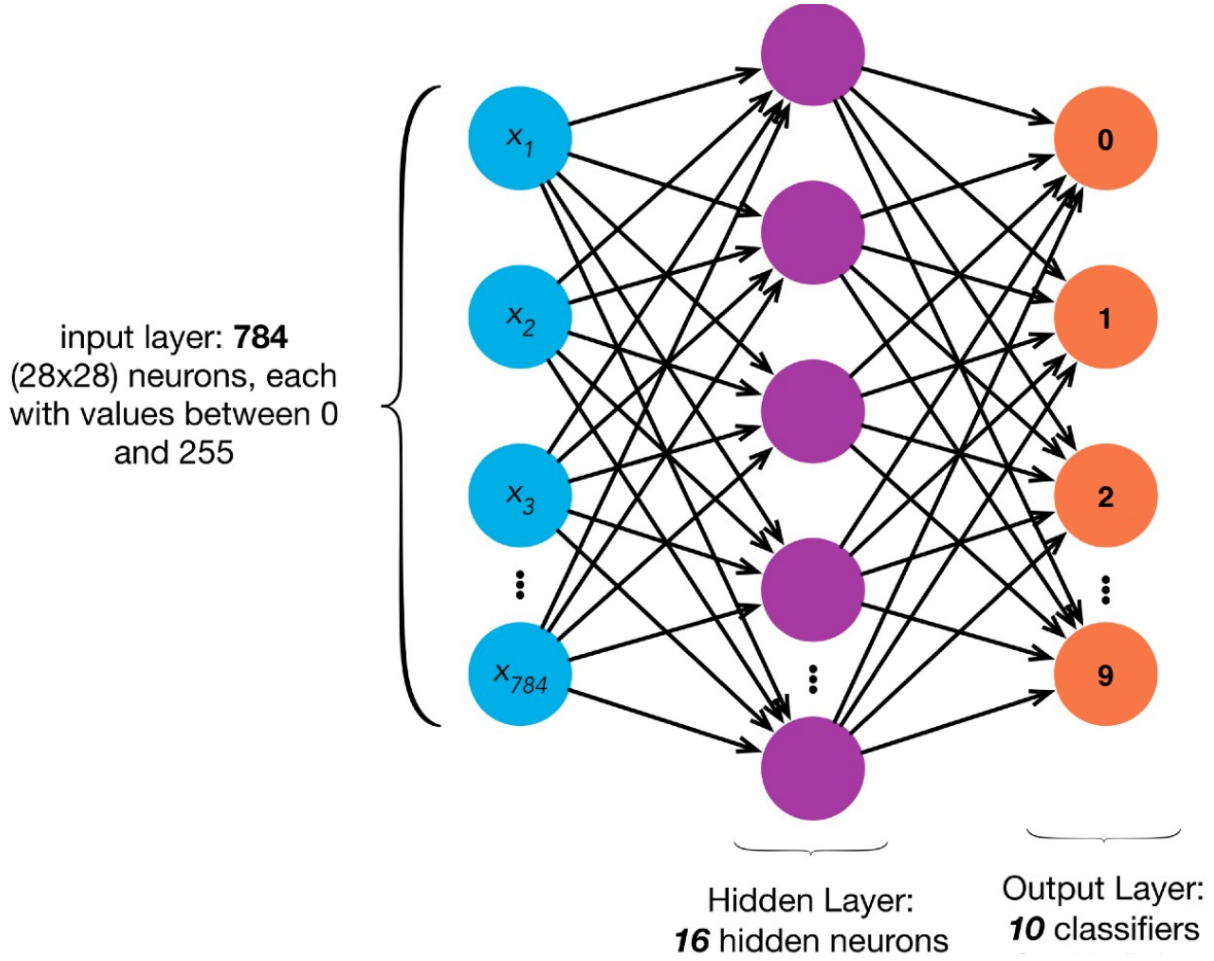
3-layered neural network with 2 hidden layers

Hidden Layer→ Adding more neurons in between input and output layer

Image Source: https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f

Multi-layer Neural Network

Example:
2-layered architecture for multi-class classification
(e.g: Fashion MNIST dataset)

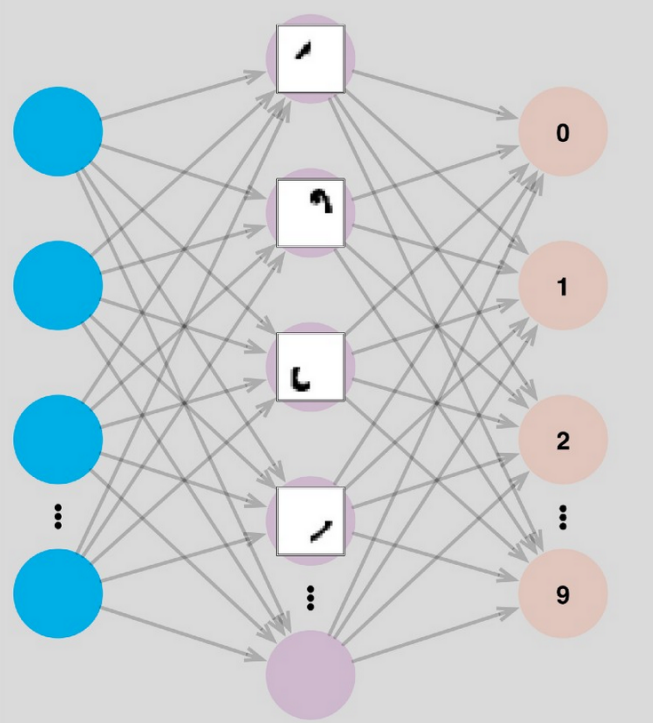


Intuition:
In a multi-layer neural network, the first hidden layer will be able to learn some very simple patterns. Each additional hidden layer will somehow be able to learn progressively more complicated patterns.

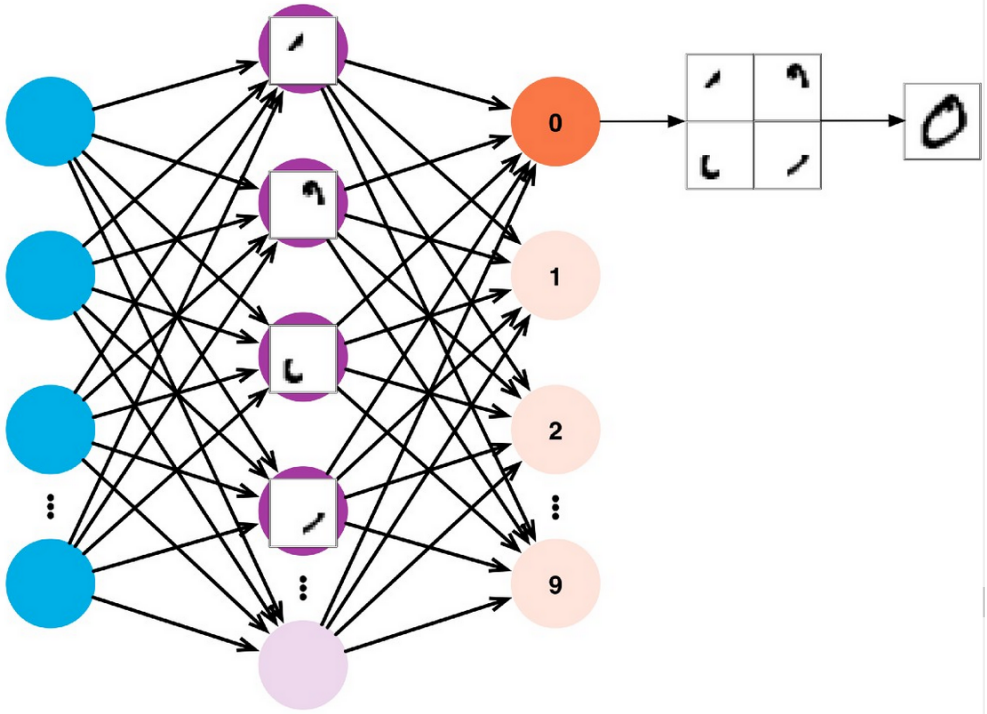
Image & content Source (Adapted) and reference to read : https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f

Multi-layer Neural Network

Example:
2-layered architecture for multi-class classification
(e.g: MNIST digit dataset) intuition



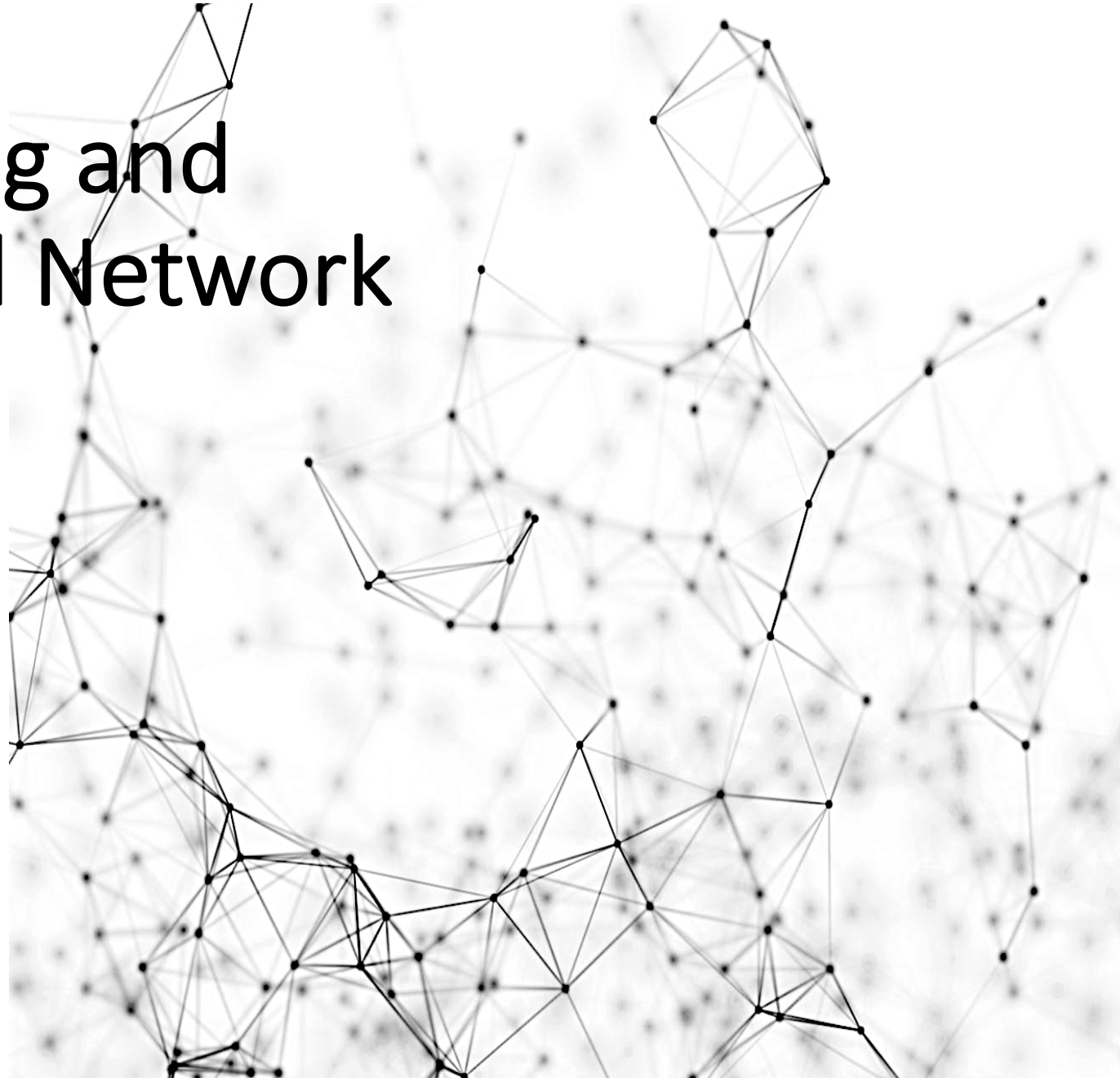
A Trained Neural Network in which each hidden neuron classifies certain pattern



Feeding a handwritten digit of 0 should trigger the 4 hidden layer neurons, and then the first output neuron

Image & content Source (Adapted) and reference to read : https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f

42028: Deep Learning and Convolutional Neural Network



Week-5 Lecture

Convolutional Neural Network (CNN) - 1

Outline

- Computer Vision tasks Recap
- CNN Layers
- Convolution layer (Padding and Stride)
- Pooling layer
- Fully Connected Layer
- CNN Layer Visualization and intuitions.

Computer Vision

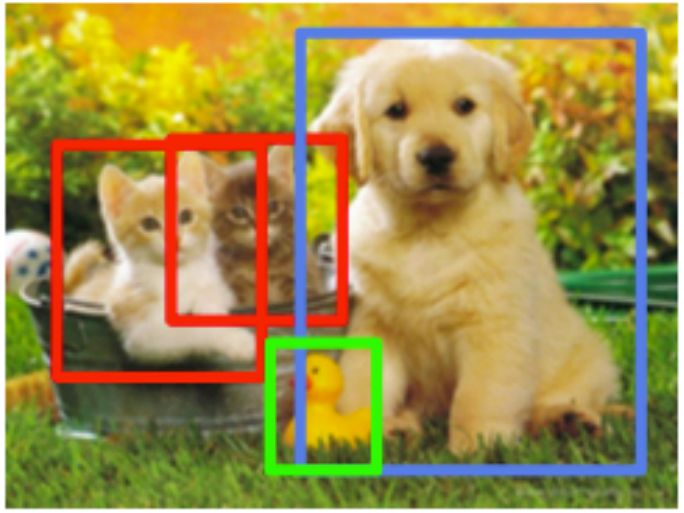
Classification

CAT

Classification + Localization

CAT

Object Detection



CAT, DOG, DUCK

Images source: [cs224d course]

Computer Vision

Classification

Cat

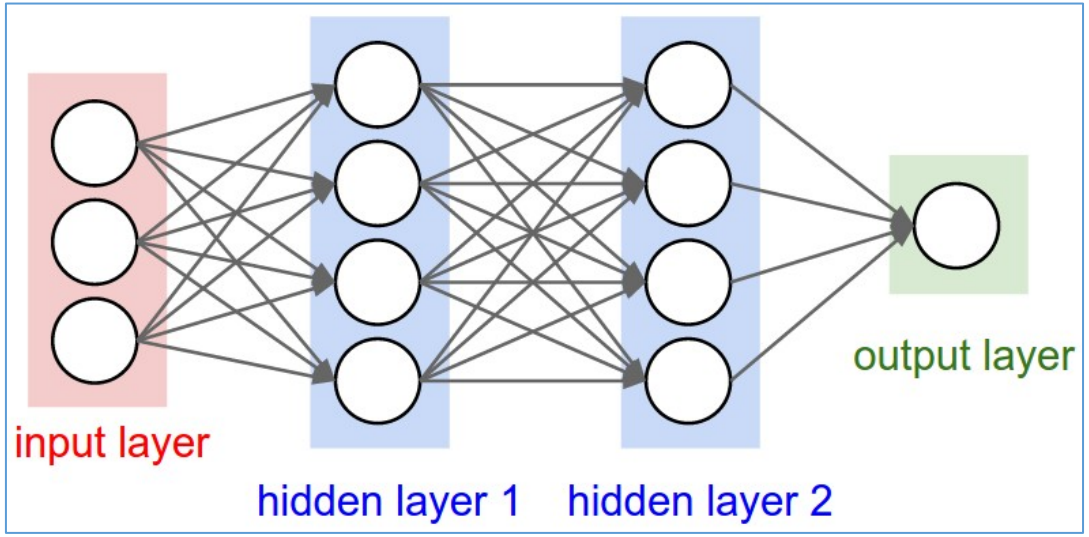
64 X 64 X 3

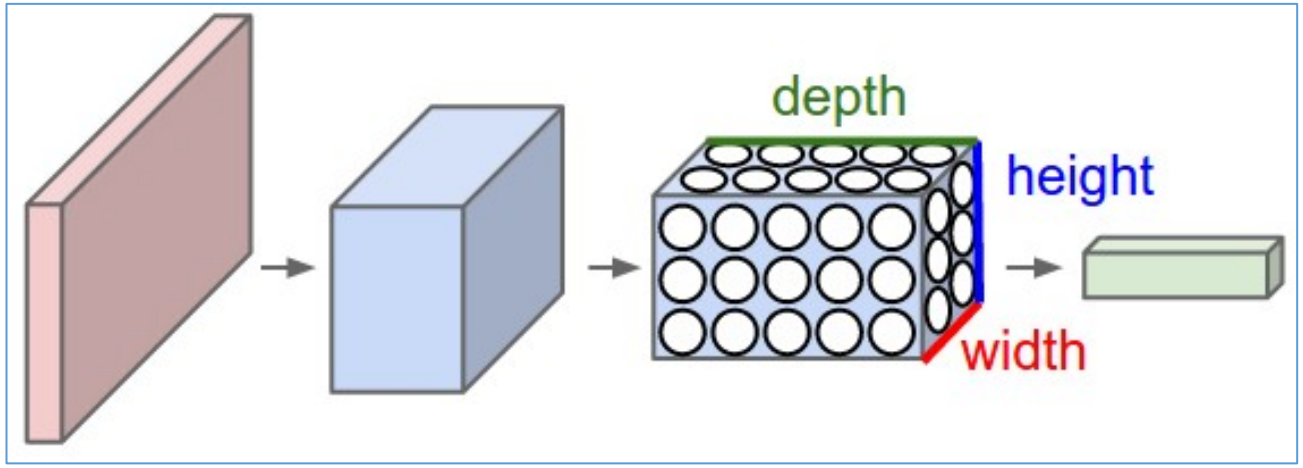
Dog

800 X 800 X 3

Images source: cs224d course

Deep-NNs Vs CNNs





ANN with 3-layers

- Fully connected
- Not great for images as input
- May lead to overfitting
- Too much of full connectivity

CNN with 3-layers

- Well suited for images with 3 dimensions
- CNNs have neurons arranged in 3D
- It is a sequence of layers which transforms input 3D volume to 3D outputs volume

Image Source and reference to read : http://cs231n.github.io/convolutional-networks/

Convolutional Neural Networks (CNN)

CNNs are the foundations of modern state-of-the-art deep learning based computer vision.

Layers in a CNN:

Three main type of layers used to build a CNN architecture

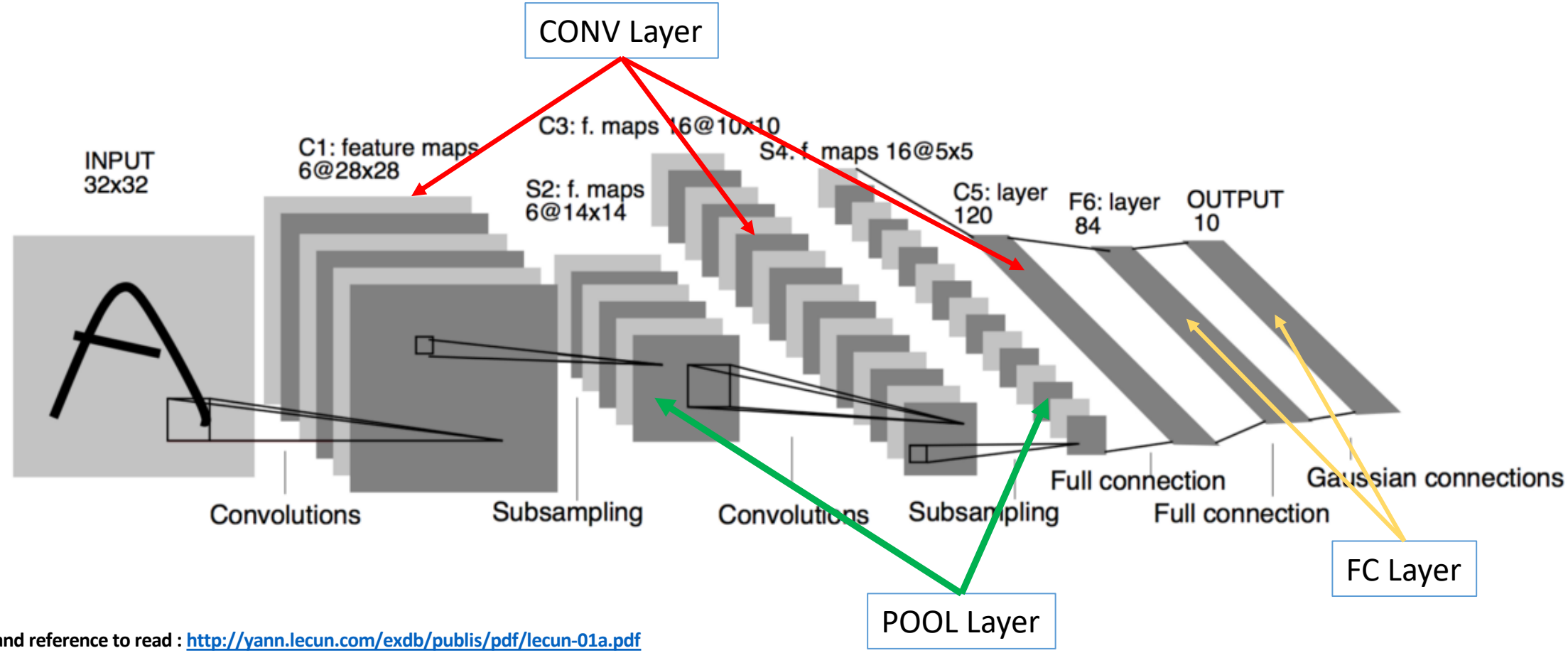
1. Convolutional Layer (**CONV**)
2. Pooling Layer (**POOL**)
3. Fully Connected layer (**FC**)

These three types of layers are stacked together to form a CNN architecture!

Source and reference to read : http://cs231n.github.io/convolutional-networks/
https://medium.com/@pechyonkin/key-deep-learning-architectures-lenet-5-6fc3c59e6f4

Convolutional Neural Networks (CNN)

Sample CNN architecture (LENET-5):



Source and reference to read : http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

Convolution Layer (CONV)

- **CONV**olution is the first layer to extract features from an input image
- Core building block of a CNN
- Convolutions are basic operation in this layer
- A number of filters (e.g. edge detectors) are applied to the input image

Source and reference to read : http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

Convolution Layer (CONV)

Convolution Operation

100	**100**	**100**	**0**	**0**	**0**
100	100	100	**0**	**0**	**0**
100	100	100	**0**	**0**	**0**
100	100	100	**0**	**0**	**0**
100	100	100	**0**	**0**	**0**
100	100	100	**0**	**0**	**0**

6 X 6 dimension image

*

Convolution operator

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

3 X 3 filter/Kernel

0	**300**	**300**	**0**
0	300	300	**0**
0	300	300	**0**
0	300	300	**0**

4 X 4 dimension matrix

Convolution Layer (CONV)

Convolution Operation

(100 X 1 + 100 X 1 + 100 X 1) +
(100 X 0 + 100 X 0 + 100 X 0) +
(100 X -1 + 100 X -1 + 100 X -1)

100 (1)	100 (0)	100 (-1)	0	0	0
100 (1)	100 (0)	100 (-1)	0	0	0
100 (1)	100 (0)	100 (-1)	0	0	0
100	100	100	0	0	0
100	100	100	0	0	0
100	100	100	0	0	0

6 X 6 dimension image

*

Convolution operator

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

3 X 3 filter/Kernel

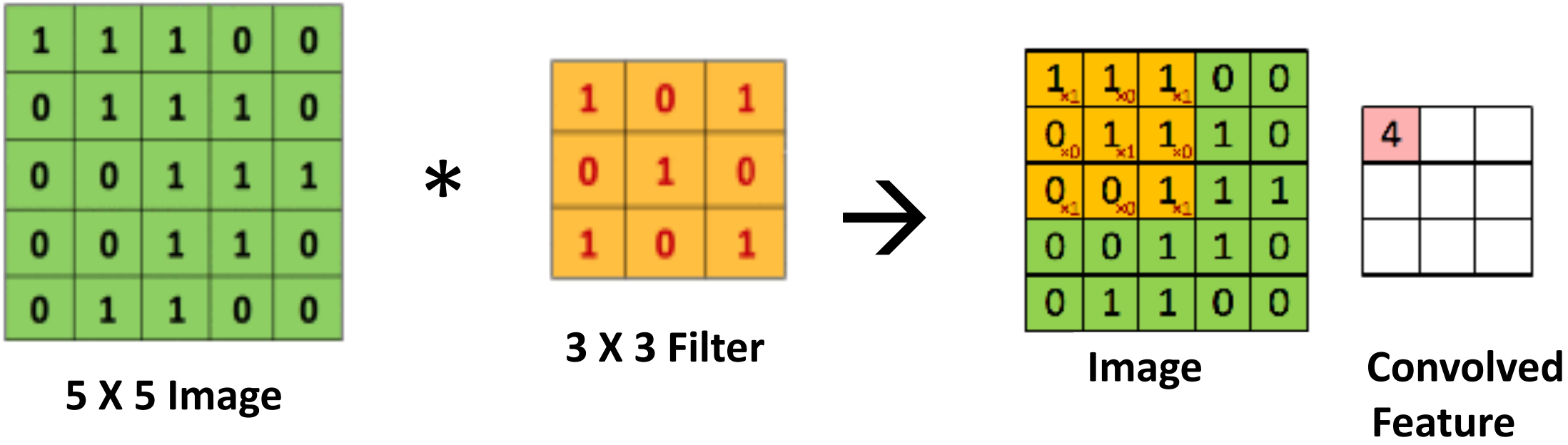
0	**300**	**300**	**0**
0	300	300	**0**
0	300	300	**0**
0	300	300	**0**

4 X 4 dimension matrix

Vertical Edge detector

Convolution Layer (CONV)

Convolution Operation



5 X 5 Image

3 X 3 Filter

Image

Convolved Feature

Image source: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

Convolution Layer (CONV)

Padding

100	**100**	**100**	**0**	**0**	**0**
100	100	100	**0**	**0**	**0**
100	100	100	**0**	**0**	**0**
100	100	100	**0**	**0**	**0**
100	100	100	**0**	**0**	**0**
100	100	100	**0**	**0**	**0**

6 X 6 dimension image without padding

Padding (p) = 1

0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	100	100	100	**0**	**0**	**0**	**0**
0	100	100	100	**0**	**0**	**0**	**0**
0	100	100	100	**0**	**0**	**0**	**0**
0	100	100	100	**0**	**0**	**0**	**0**
0	100	100	100	**0**	**0**	**0**	**0**
0	100	100	100	**0**	**0**	**0**	**0**
0	0	0	**0**	**0**	**0**	**0**	**0**

8 X 8 dimension matrix with padding

Convolution Layer (CONV)

Padding

0	0	0	0	0	0	0	0
0	100	100	100	0	0	0	0
0	100	100	100	0	0	0	0
0	100	100	100	0	0	0	0
0	100	100	100	0	0	0	0
0	100	100	100	0	0	0	0
0	100	100	100	0	0	0	0
0	0	0	0	0	0	0	0

8 X 8 dimension matrix

*

Convolution operator

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

3 X 3 filter/Kernel

-200	0	200	200	0	0
-300	0	300	300	0	0
-300	0	300	300	0	0
-300	0	300	300	0	0
-300	0	300	300	0	0
-200	0	200	200	0	0

6 X 6 dimension matrix
== Input Matrix dimension

Convolution Layer (CONV)

Padding

Input Matrix Dimension : $n\ x\ n$

Filter size: $f\ x\ f$

Padding (p) : 1

So, $(\boldsymbol{n\ X\ n}) * (\boldsymbol{f\ X\ f})$ will produce $(\boldsymbol{n + 2p - f + 1})\ \boldsymbol{X}\ (\boldsymbol{n + 2p - f + 1})$

Input Matrix

Output Matrix

e.g.: $(6\ X\ 6)\ * (3\ X\ 3) \rightarrow\ \ 6\ X\ 6$ Output matrix

Reference and Source: http://cs230.stanford.edu/files/C4M1.pdf

Convolution Layer (CONV)

Padding

Given: **Input Matrix Dimension** : $n\ x\ n$

Filter size: $f\ x\ f$

Required Output Size = $(\boldsymbol{n + 2p - f + 1})\boldsymbol{X}\ (\boldsymbol{n + 2p - f + 1})$

Question: What is pad size ($\boldsymbol{p}$) so that the input and output matrix are of same sizes?

So, $(n + 2p - f + 1) = n$

$$p = \frac{(f\ -1)}{2}$$

Reference and Source: http://cs230.stanford.edu/files/C4M1.pdf

Convolution Layer (CONV)

Padding (Same and Valid)

Valid Padding: ≈ *No Padding* (Padding $\boldsymbol{p}$ = 0)

So, Output size will be → $(\boldsymbol{n} - \boldsymbol{f} + \mathbf{1})\boldsymbol{X}\ (\boldsymbol{n} - \boldsymbol{f} + \mathbf{1})$

Same Padding: ≈ *Output size and input size is same*, this requires appropriate padding. Hence use $p = \frac{(f\ -1)}{2}$, for calculate the required padding.

Convolution Layer (CONV)

Stride

It is the number of pixels by which we slide the filter over the input matrix

Example:

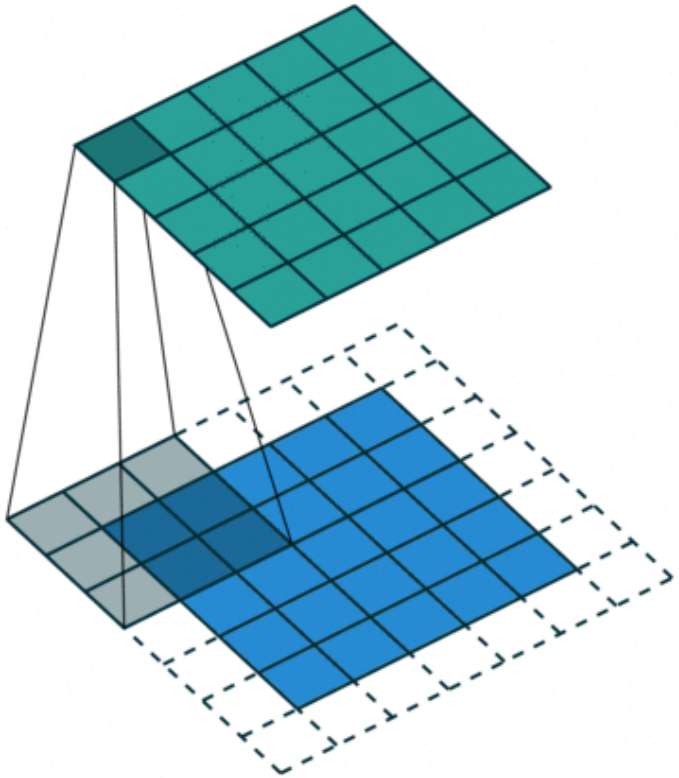
1. Stride(s) = 1: Move the filter by one pixel horizontally and vertically
2. Stride(s) = 2: Move the filter by two pixels horizontally and vertically

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

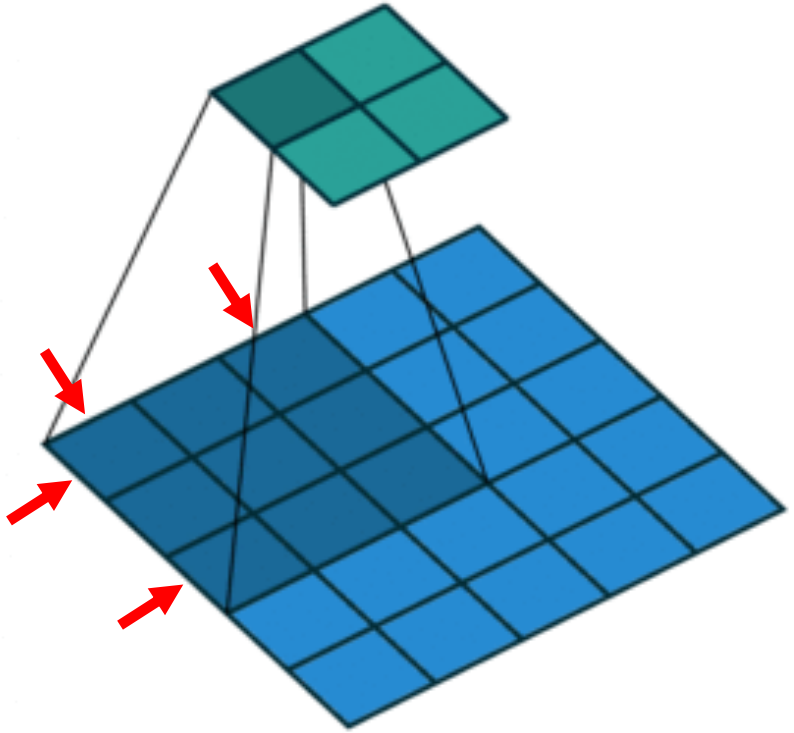
Convolution Layer (CONV)

Stride and Padding illustration

Convolution with stride (s) =1
padding (p) = 1



Convolution with stride (s) =2
padding (p) = 0



Convolution with stride (s)=2,
padding (p) = 1

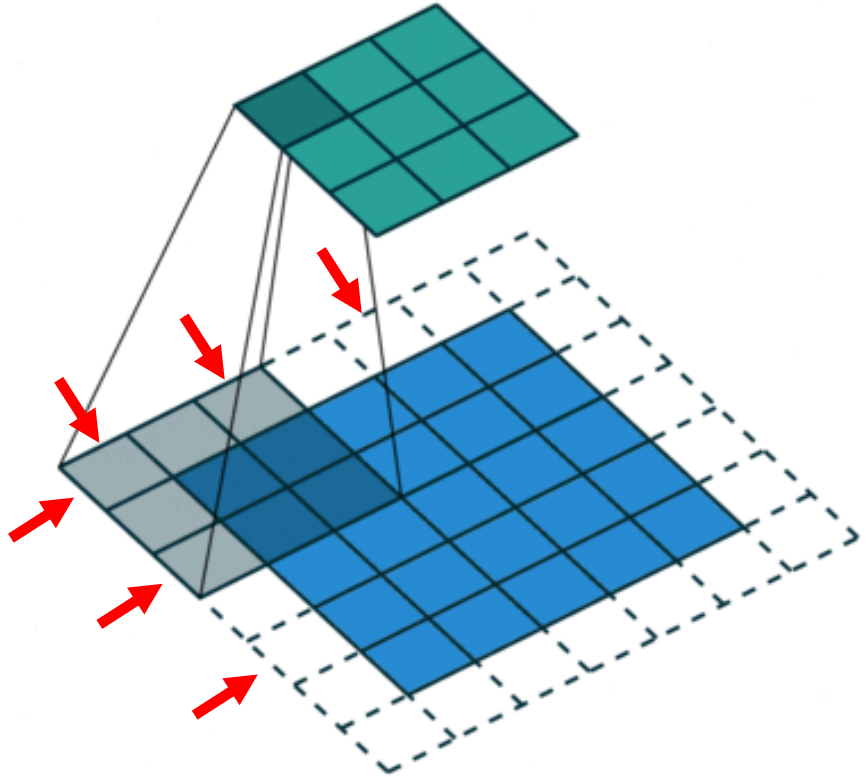


Image source: https://github.com/vdumoulin/conv_arithmetic

Convolution Layer (CONV)

Output size with Stride and padding

Given:

Input Matrix Dimension : $n \; x \; n$

Filter size: $f \; x \; f$

Padding: p

Stride : s

$$\textbf{Output Size} = \left(\frac{n + 2p - f}{s} + 1 \right) \boldsymbol{X} \left(\frac{n + 2p - f}{s} + 1 \right)$$

Example:
Input Matrix Dimension : $7 \; x \; 7$, **Filter size**: $3 \; x \; 3$
Padding: 0, Stride : 2

$$\textit{Output Size} = 3 \; X \; 3$$

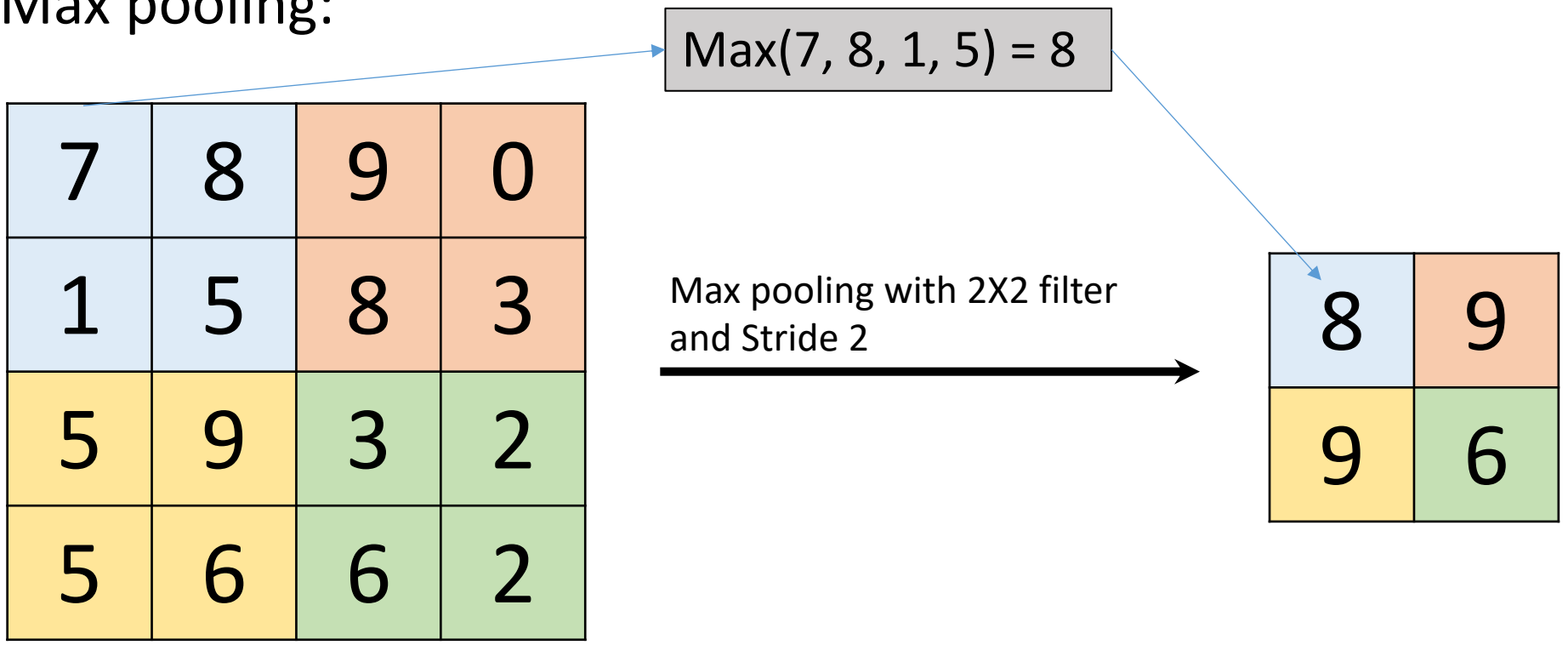
Reference and Source: http://cs230.stanford.edu/files/C4M1.pdf

Pooling Layer (POOL)

- Pooling layer is a down sampling operation which reduces the dimensionality of a matrix.
- In other words, it reduces the number of parameters for large image, but retain the valuable information.
- 3 types:
  - Max pooling
  - Average pooling
  - Sum pooling

Pooling Layer (POOL)

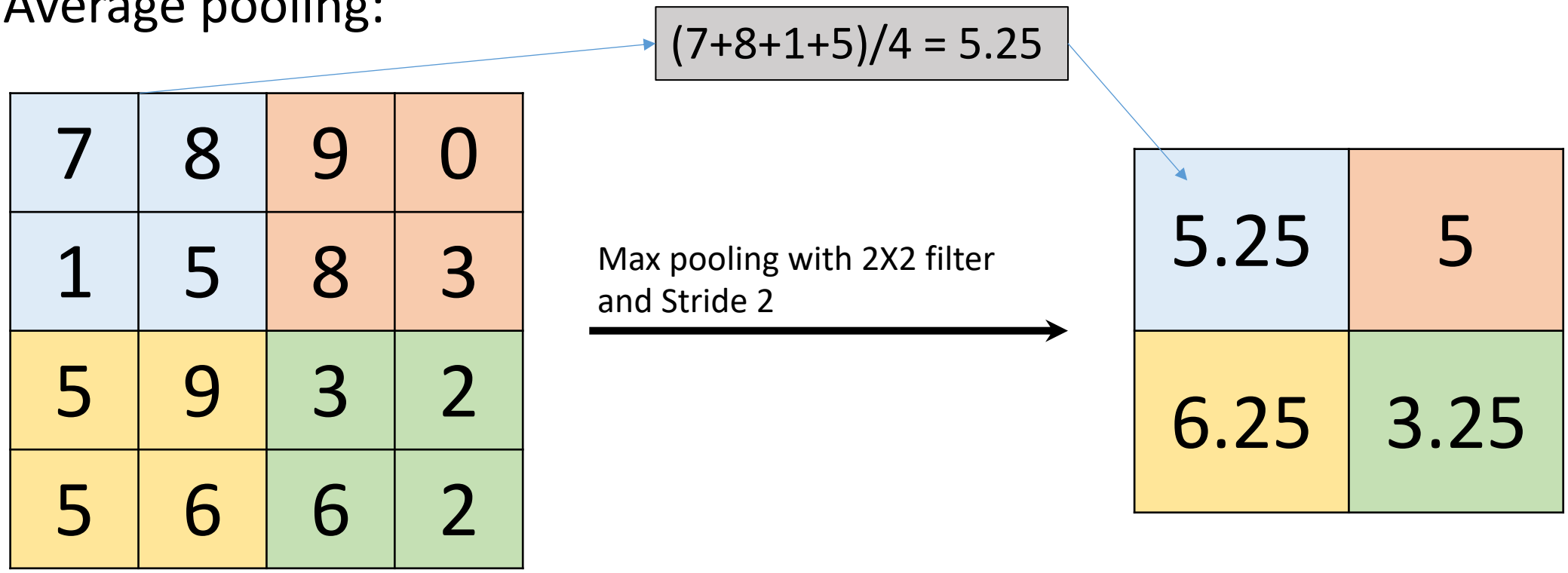
- Max pooling:



Source: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

Pooling Layer (POOL)

- Average pooling:



Source: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

Fully Connected Layer (FC)

- In FC layer, the output matrix after convolution layer is flattened and feed into a fully connected layer similar to ANN

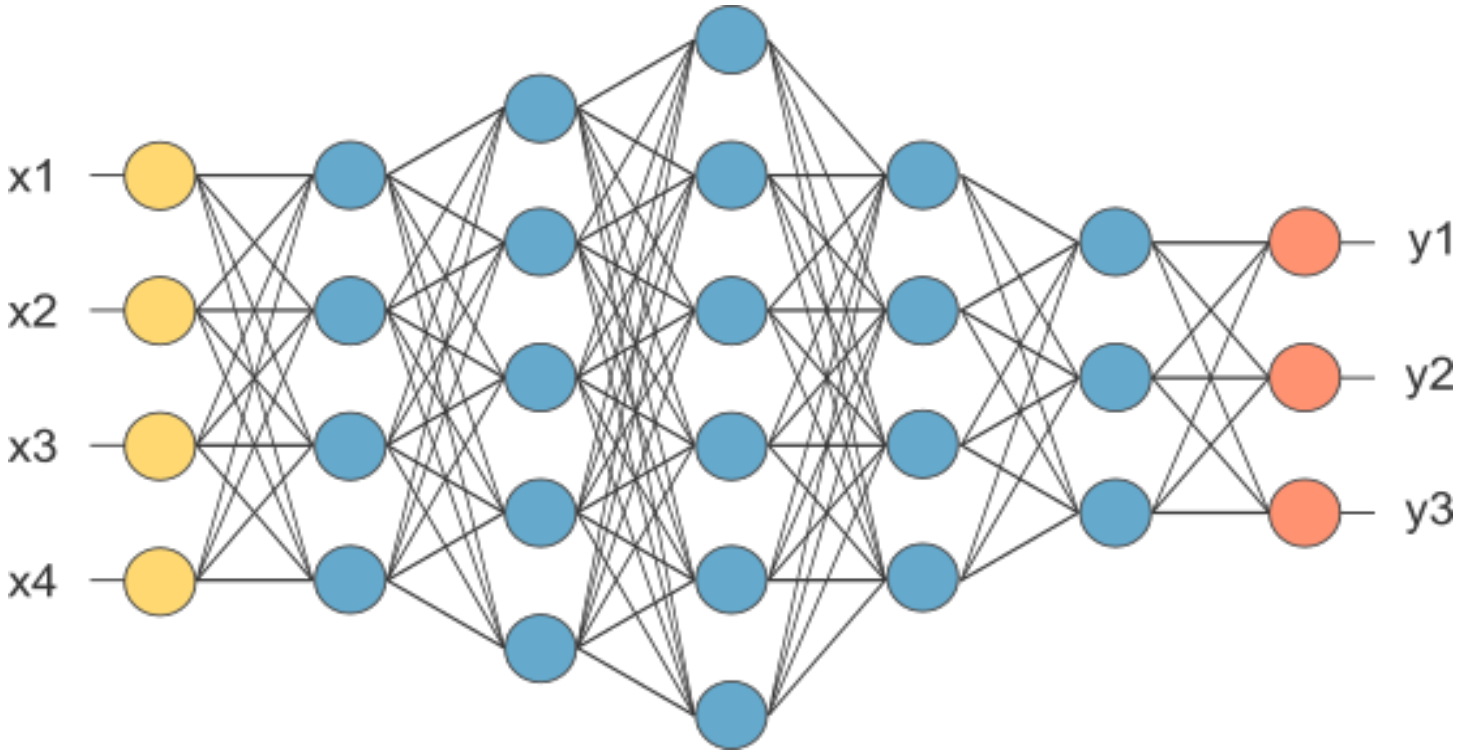


Image source ad reference: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

Fully Connected Layer (FC)

- It is a traditional Multi-layer Perception/Neural Network
- For multi-class classification, usually Softmax activation is used.
- Softmax ensures the output
- Output of the CONV and POOL layers represent a high level features of the Input image
- FC layer uses this feature to classify the input image into the desired class.

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

CNN layers visualization and intuition

Example: Face recognition using CNNs

Uses simple shapes to form higher level features like facial shapes!

Uses edges to detect simple shapes

Low level feature like edges from raw pixels

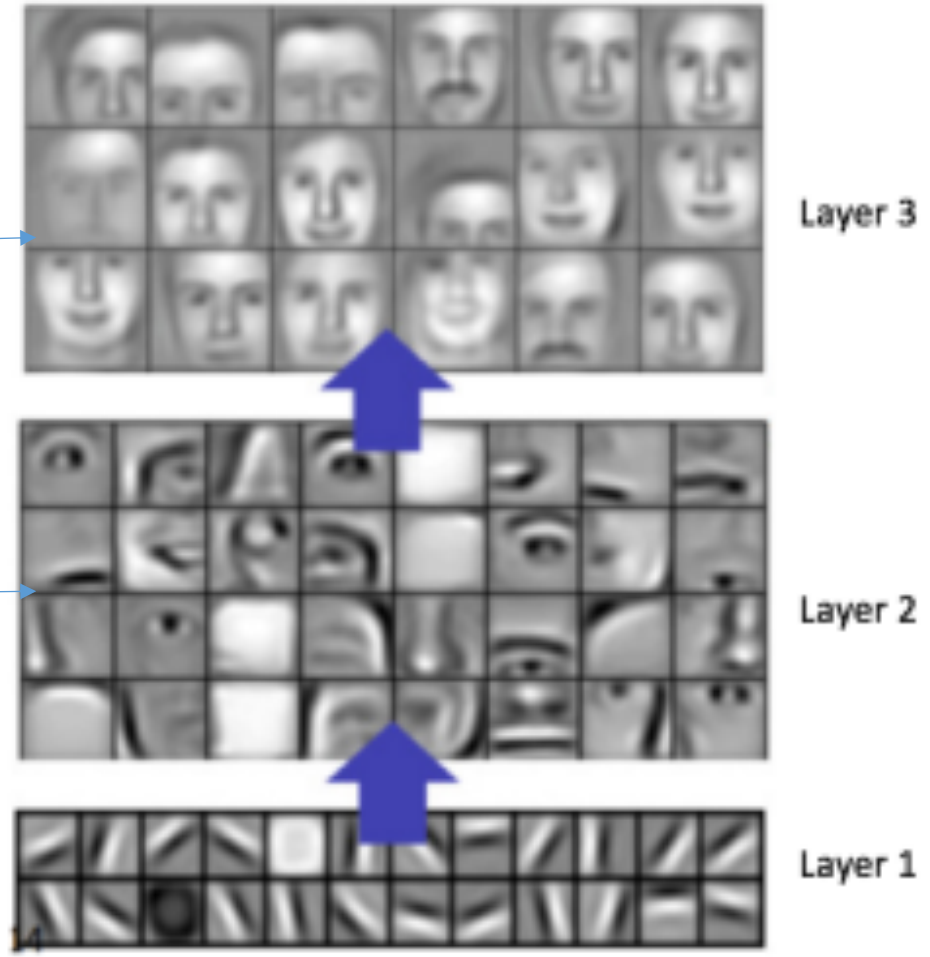
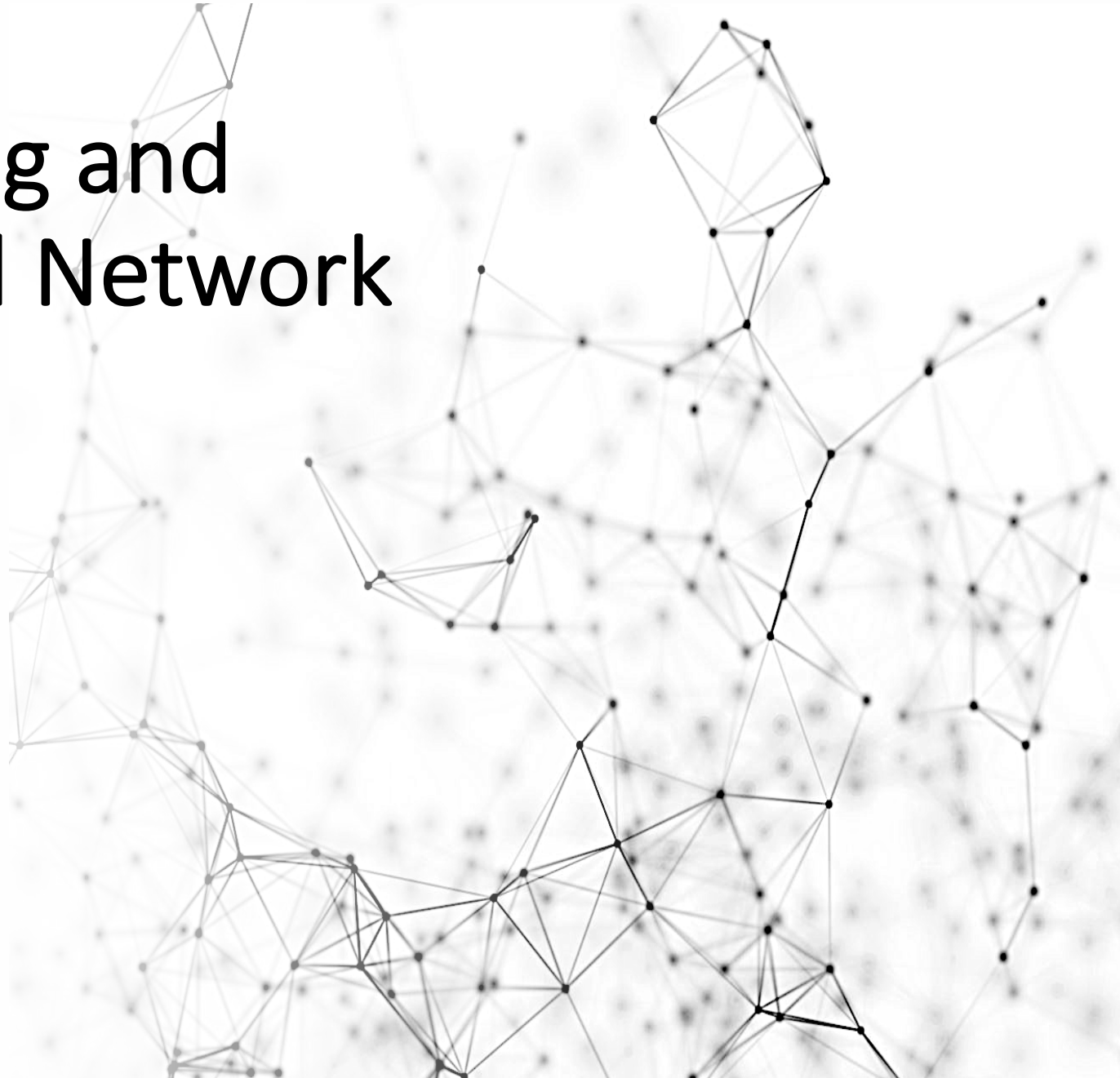


Image Source: http://web.eecs.umich.edu/~honglak/icml09-ConvolutionalDeepBeliefNetworks.pdf

42028: Deep Learning and Convolutional Neural Network



Week-6 Lecture

Convolutional Neural Network (CNN) - 2

Outline

- Dataset preparation
- Bias and Variance
- Fixing Bias and Variance issues
- Regularization
- L1 and L2 Regularization
- Dropouts
- Data Augmentation

Dataset preparation

- In case of small dataset (Range : 100 - <100k)
  - Train set: 60%
  - Validation set: 20%
  - Test set: 20%

Or,

  - Train set: 70%
  - Test set: 30%

Popular dataset spit choice in non-DL era!
Or Small Data era!

Source and reference: https://cs230-stanford.github.io/train-dev-test-split.html

Dataset preparation

- In case of Large dataset (Range : 500K - 1M+)

Example: Total data sample : 1M+

Train: 98% !

Validation: 10,000 samples

Test: 10,000 samples

Popular dataset spit choice in DL era!
Or BIG Data era!

Source and reference: https://cs230-stanford.github.io/train-dev-test-split.html

Dataset preparation

Train, validation and test set distribution:

Rule of Thumb:

Validation and Test set should come from the same distribution

Source and reference: https://cs230-stanford.github.io/train-dev-test-split.html

Bias and Variance

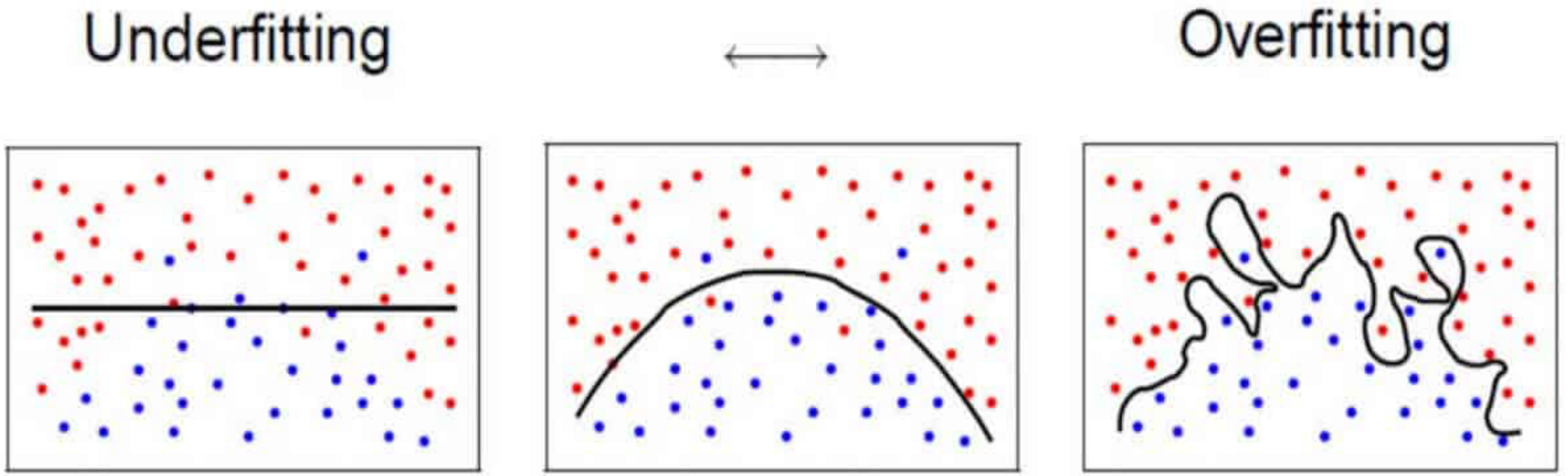
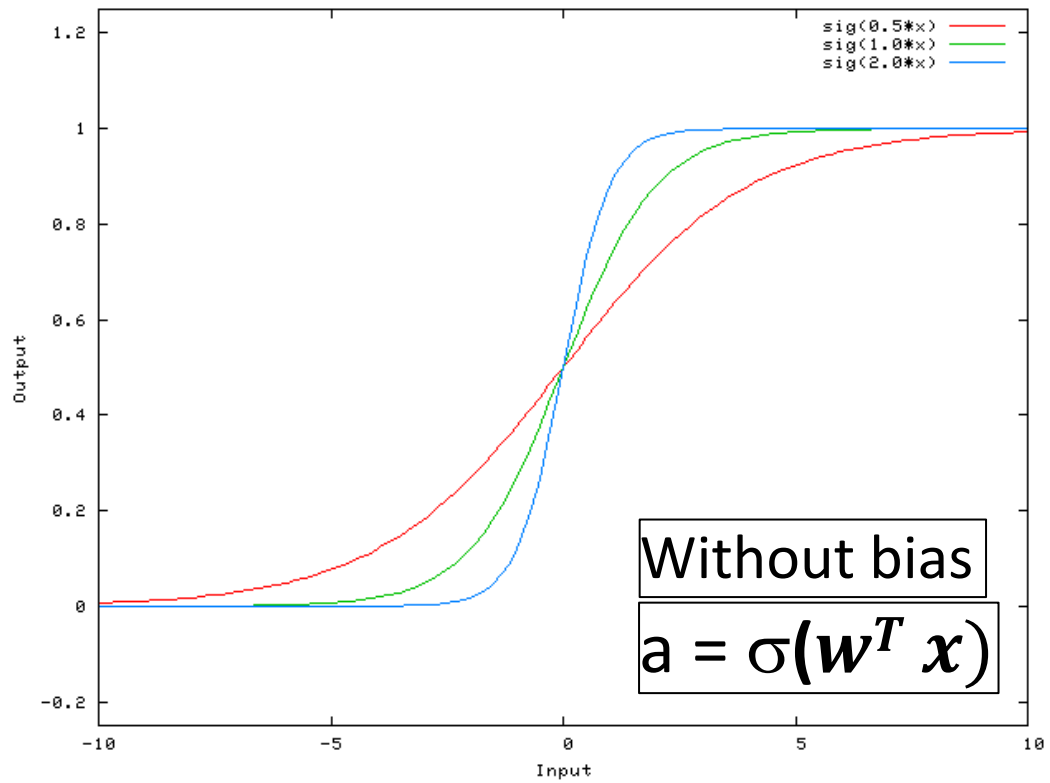
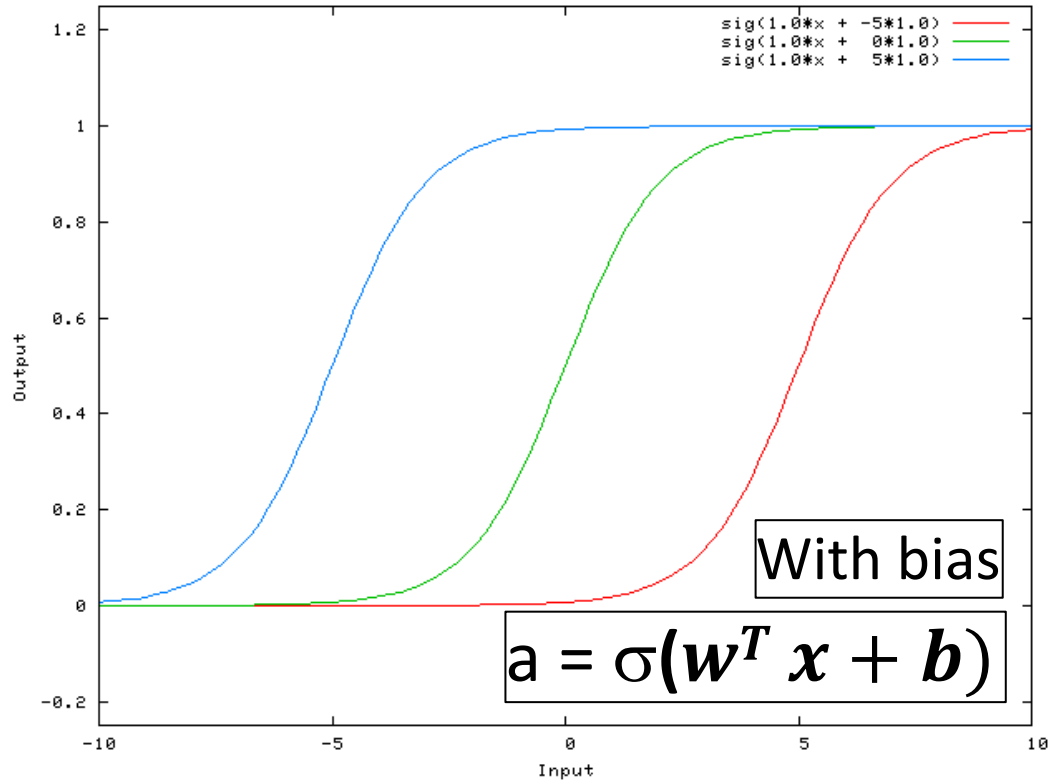


Image Source: https://cv-tricks.com/machine-learning/bias-variance-trade-off/

Bias

- It is a value that allows to shift the activation function to left or right, to better fit the data



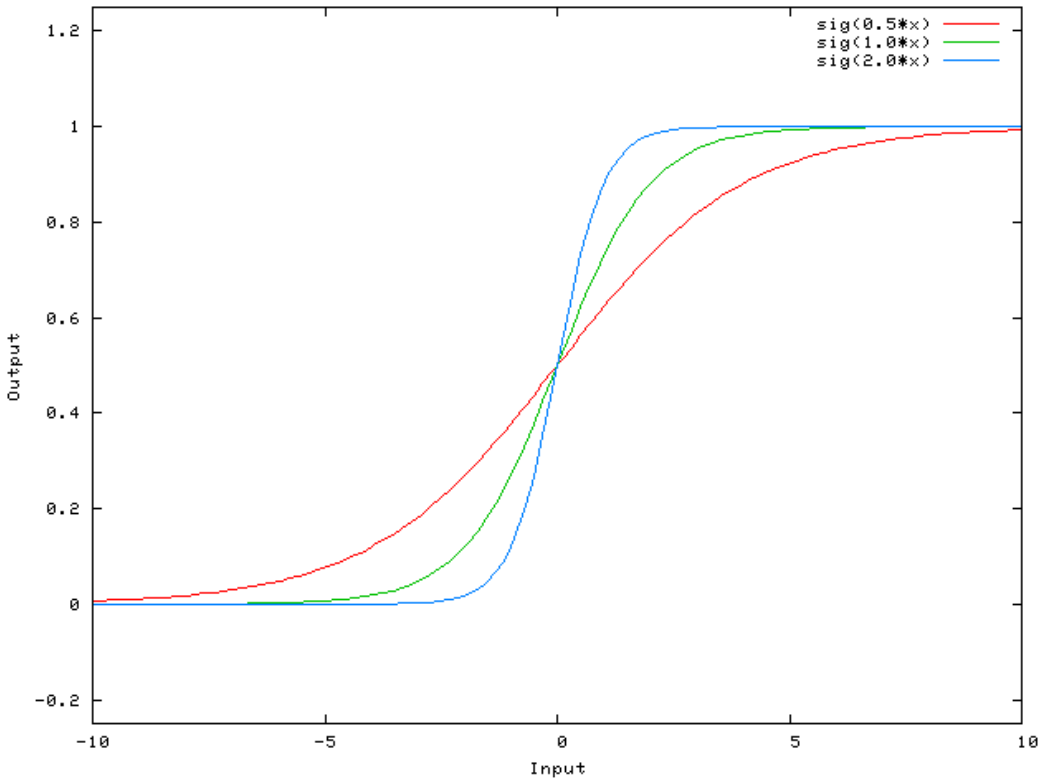


Reference and (image) Source: https://medium.com/deeper-learning/glossary-of-deep-learning-bias-cf49d9c895e2

Bias

- Changes in '**w**' alters the steepness of the curve, keeping the origin at (0,0) or same/unchanged

- Without bias we may get a poor fit to training data

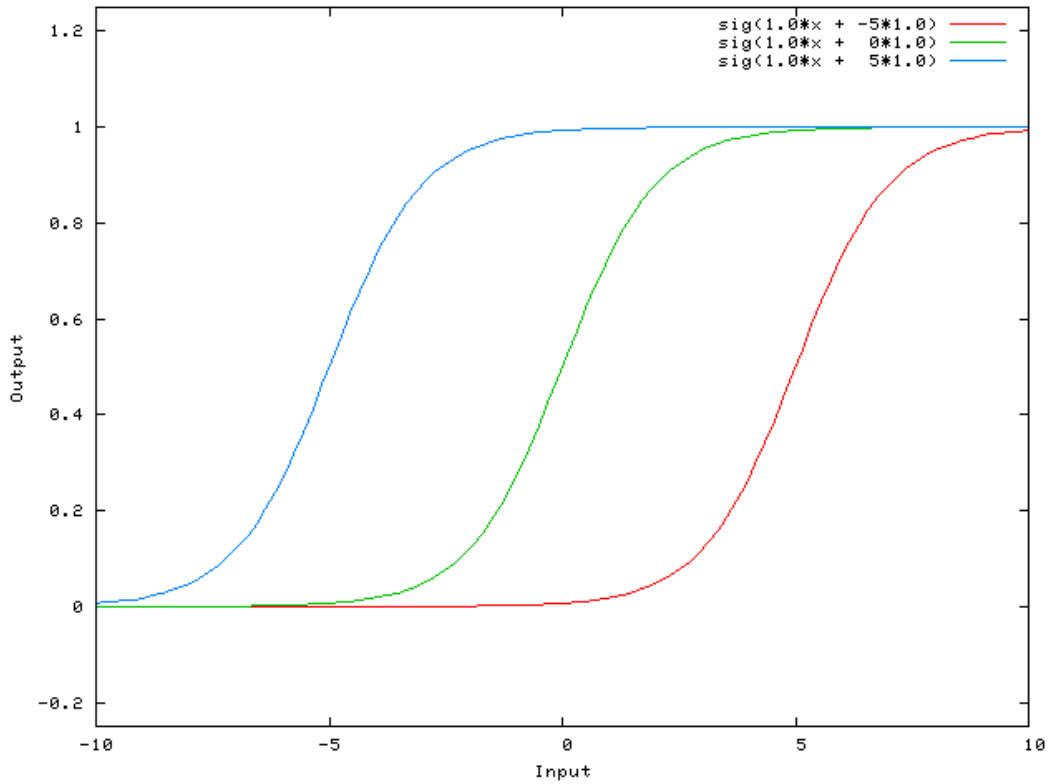


Without bias $a = \sigma(w^T x)$

Reference and (image) Source: https://medium.com/deeper-learning/glossary-of-deep-learning-bias-cf49d9c895e2

Bias

- Changes in '**b**' shifts the curve to left or right
- With bias the curve/line will not always pass through origin
- We get a better fit to training data



With bias $a = \sigma(\boldsymbol{w}^T \boldsymbol{x} + \boldsymbol{b})$

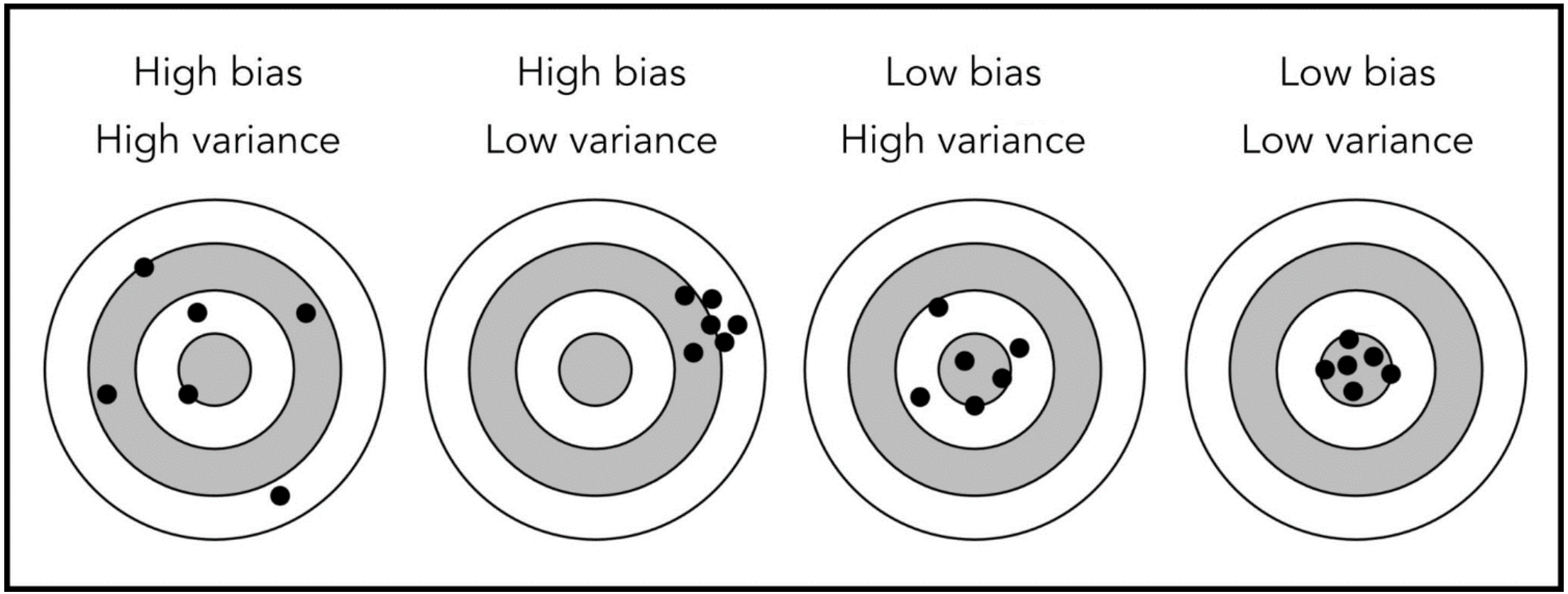
Reference and (image) Source: https://medium.com/deeper-learning/glossary-of-deep-learning-bias-cf49d9c895e2

Variance

- It is the change in prediction accuracy of Machine Learning model between training data and test data.
- Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn’t seen before.
- With high variance, models perform very well on training data but has high error rates on test data.

Reference and Source: https://medium.com/datadriveninvestor/bias-and-variance-in-machine-learning-51fdd38d1f86

Bias and Variance effect



Reference and Source: http://snoek.ddns.net/~oliver/mysite/the-bias-variance-tradeoff.html

Bias and Variance effect

- **Identify High Bias:**
  - High training error
  - Validation/test error nearly same as train error
- **Identify High Variance:**
  - Low training error
  - High validation/test error

Reference and Source: https://medium.com/datadriveninvestor/bias-and-variance-in-machine-learning-51fdd38d1f86

Bias and Variance effect

- **High Bias Low Variance:** Models are consistent but inaccurate
- **High Bias High Variance:** Models are inconsistent and inaccurate
- **Low Bias and Low Variance**: Models are consistent and accurate
- **Low Bias and High Variance:** Models are somewhat accurate but inconsistent on average

Reference and Source: https://medium.com/datadriveninvestor/bias-and-variance-in-machine-learning-51fdd38d1f86

Fixing Bias and Variance issues

- **High Bias:** Due to simple ML model and high training error.
- **Potential things to try :**
  - Increase features: this will help in generalizing dataset
  - Make ML model more complicated
  - Decrease *Regularization* parameter

Reference and Source: https://medium.com/datadriveninvestor/bias-and-variance-in-machine-learning-51fdd38d1f86

Fixing Bias and Variance issues

- **High Variance:** Due to a ML model which is fitting most of the training dataset - overfitting.
- **Potential things to try :**
  - Increase dataset size
  - Reduce input features
  - Increasing *Regularization* parameter

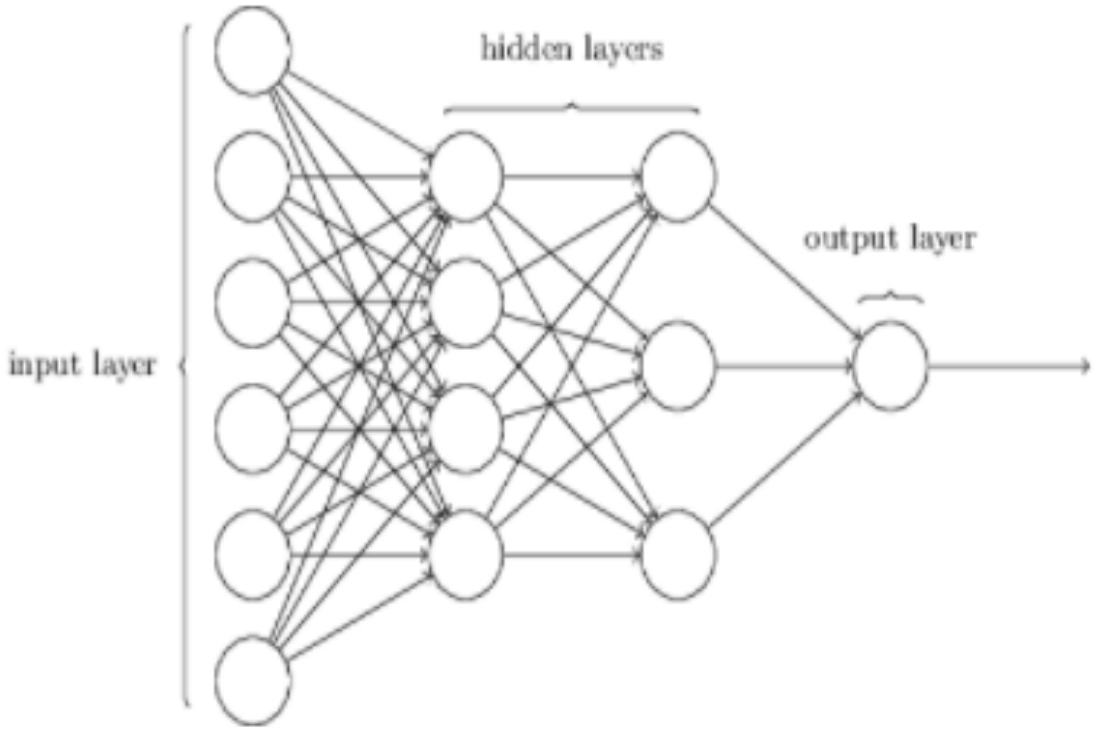
Reference and Source: https://medium.com/datadriveninvestor/bias-and-variance-in-machine-learning-51fdd38d1f86

Regularization

- Regularization is a technique which makes slight modifications to the learning algorithm such that the model generalizes better.
- Improves the model's performance on the unseen data as well.
- Popular techniques:
  - L2 and L1 regularization
  - Dropout

Source and reference: https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/

Regularization



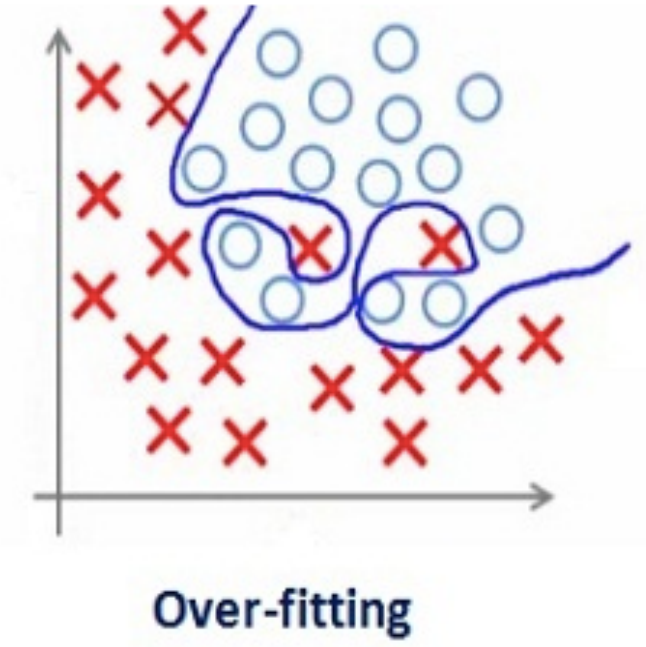
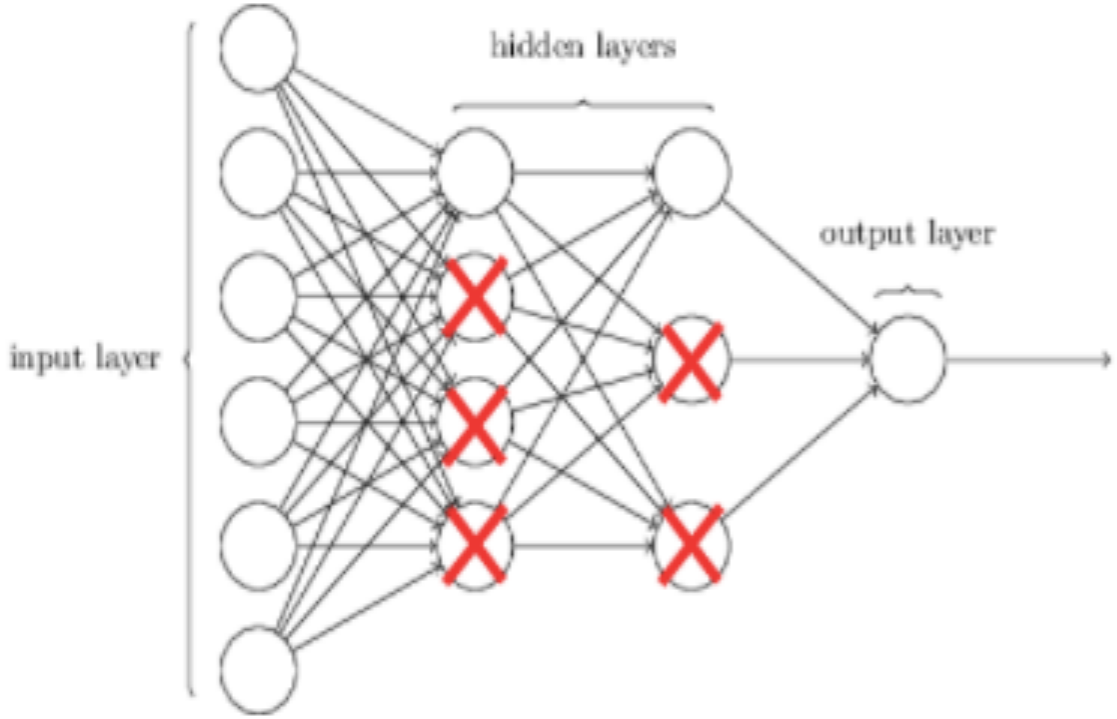


Image Source and reference: https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/

Regularization



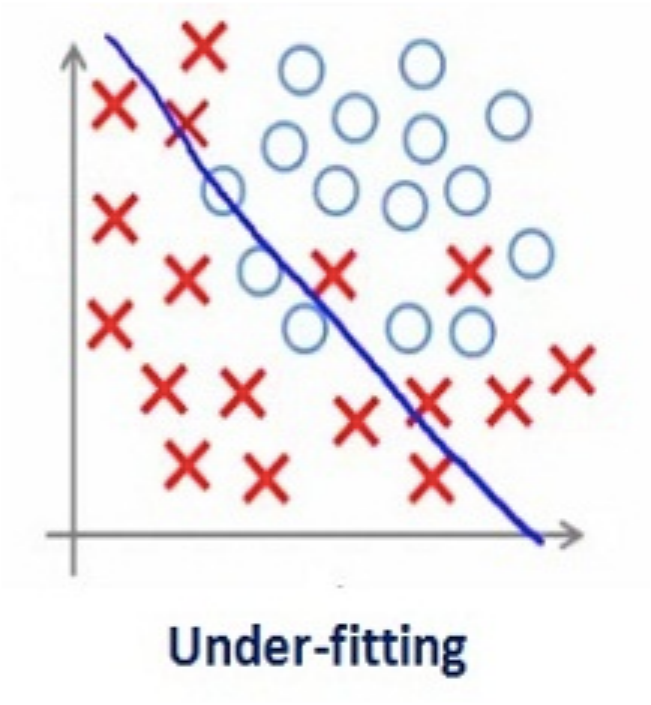


Image Source and reference: https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/

Regularization- L1 and L2

- L2 and L1 regularization are common types and help in reducing the overfitting issue
- Idea: Update the loss/cost function by adding a regularization term

Loss function = Loss + Regularization term (λ)

- Due to λ, the weight matrices will decrease, assuming a neural network with smaller weight matrices leads to simpler model
- In Deep Learning, Regularization penalizes the weight matrices of the nodes

Source and reference: https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/

Regularization- L1 and L2

- L2 regularization:

$$\text{Cost function} = \text{Loss} + \frac{\lambda}{2m} * \sum ||w||^2$$

λ is a hyper-parameter

Also know as weight decay, as it forces the weight to decay towards zero, but not exactly zero.

Source and reference: https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/

Regularization- L1 and L2

- L1 regularization:

$$\text{Cost function} = \text{Loss} + \frac{\lambda}{2m} * \sum ||w||$$

- Penalize the absolute value of the ‘w’
- Weight may reduce to zero
- Useful in compressing a model

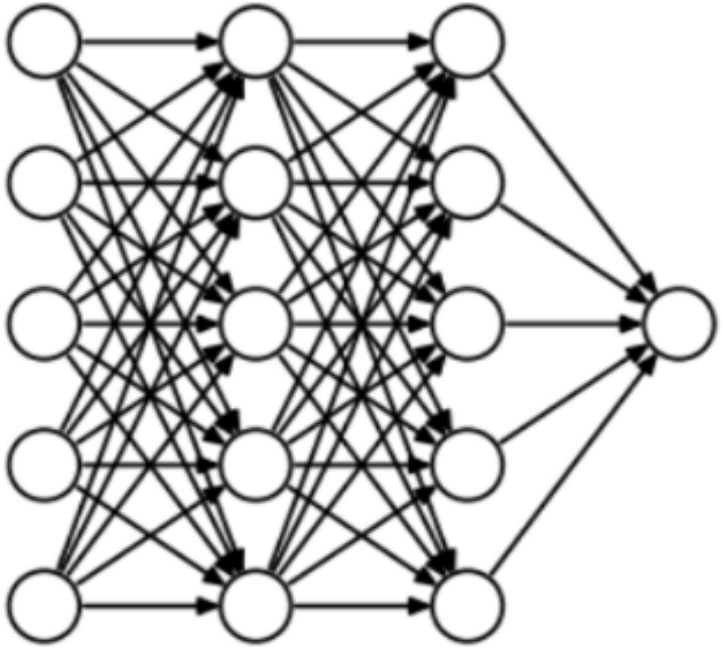
Source and reference: https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/

Regularization- Dropout

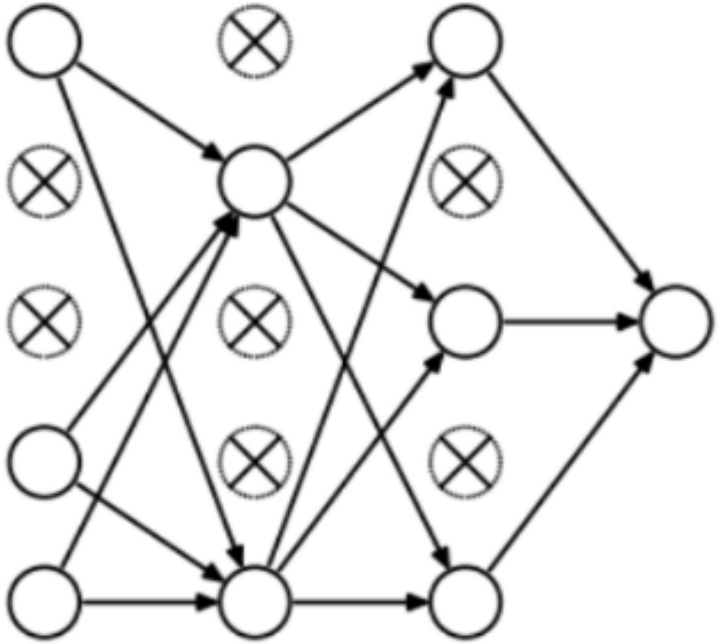
- It produces good results and most popular regularization technique
- At every iteration it randomly selects and drops some nodes and remove all the connections to and from them
- Each iteration has a different set of nodes

Source and reference: https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/

Regularization- Dropout



Example Deep NN

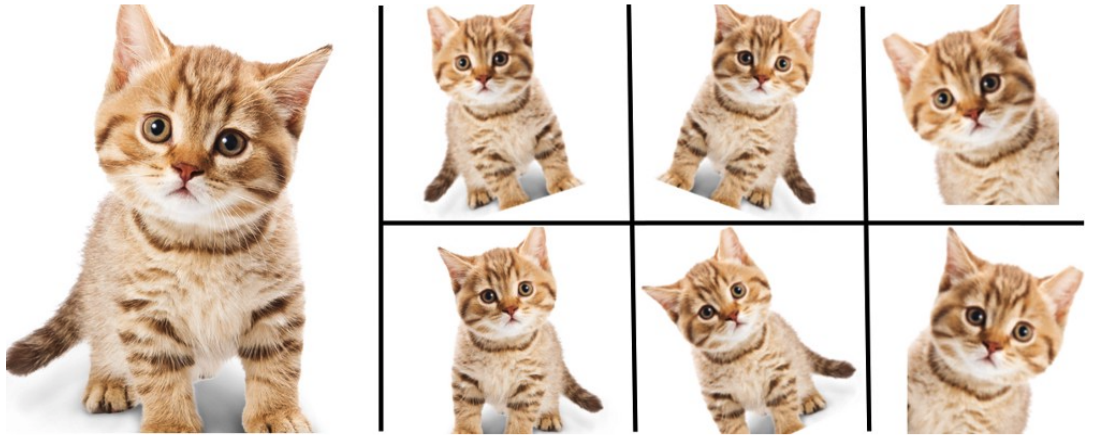


Example Deep NN with Dropout

Source and reference: https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/

Data Augmentation

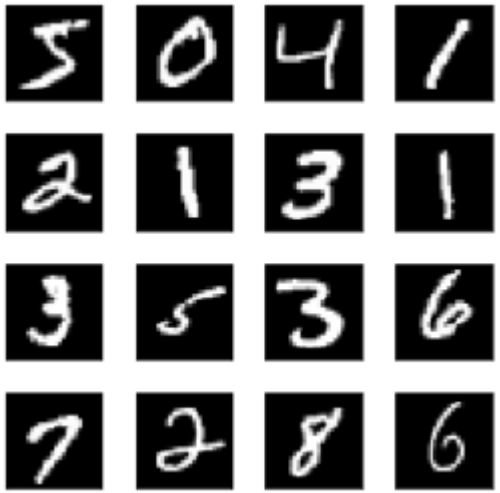
- Another simple way to reduce overfitting is to increase size of training dataset!
- Increase the size of training data by creating more sample using the existing training set and applying the following simple operations:
  - Flip
  - Rotate
  - Scale
  - Crop
  - Translate
  - Gaussian Noise



Source and reference: https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/

Data Augmentation

- Advanced data augmentation techniques:
  - Generative Adversarial Networks (GANs):
    - Among the hottest topic is DL
    - Able to generate images which look similar to the original ones
    - Proven to be very effective



Original image from MNIST

GAN generated

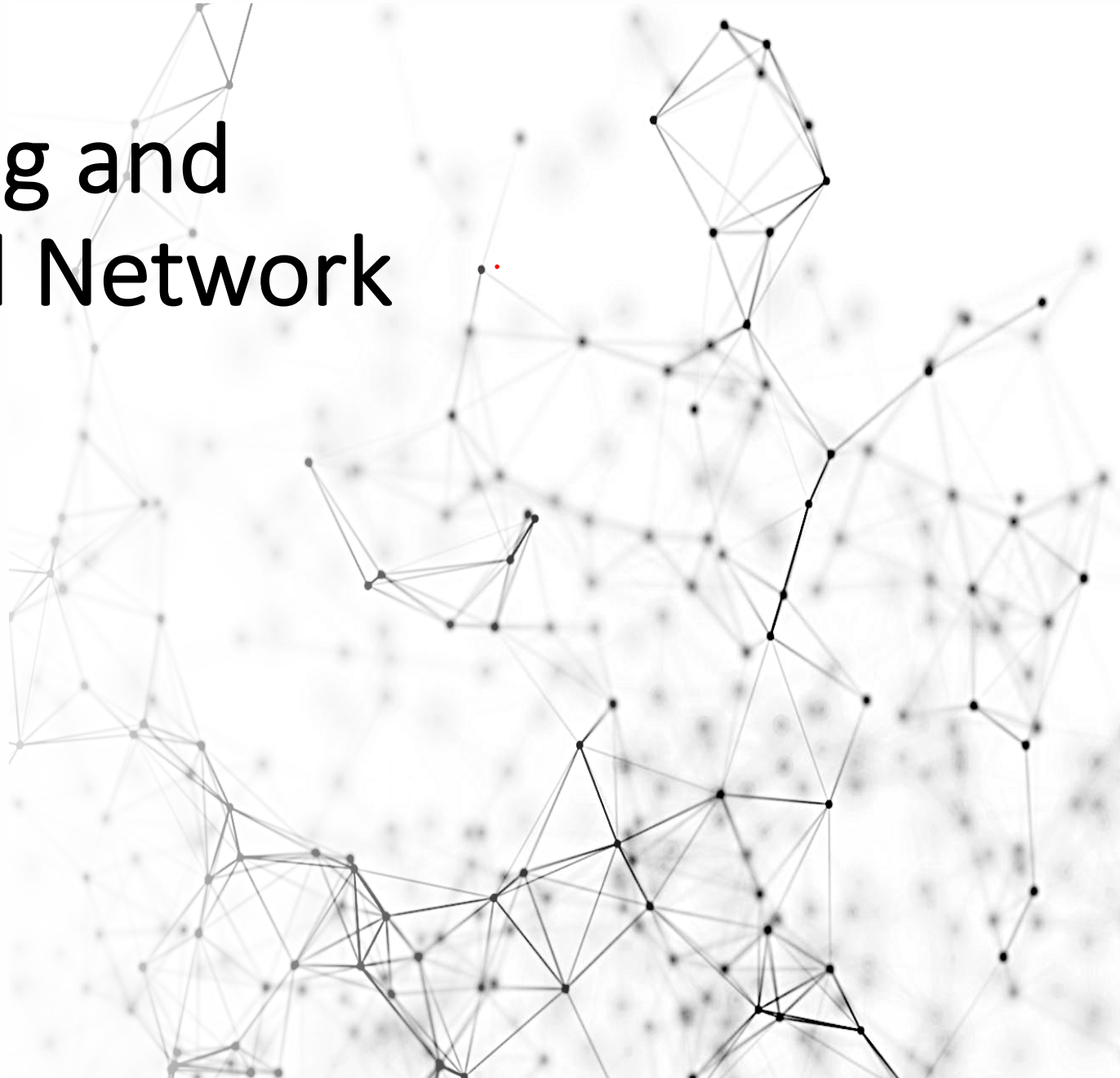
Source and reference: https://towardsdatascience.com/advanced-data-augmentation-strategies-383226cd11ba
Image Source: https://towardsdatascience.com/having-fun-with-deep-convolutional-gans-f4f8393686ed

Data Augmentation

- Advanced data augmentation techniques:
  - Neural Style transfer:
    - Using CNN to separate style
    - transfer style to different image

Source and reference: https://towardsdatascience.com/advanced-data-augmentation-strategies-383226cd11ba

42028: Deep Learning and Convolutional Neural Network



Week-7 Lecture

Convolutional Neural Network (CNN) - 3

Outline

- Exponentially Weighted Averages
- Optimizers
  - SGD with Momentum
  - RMSProp
  - Adam
- Transfer Learning strategies
- Classic CNN architectures case studies
  - AlexNet
  - Inception/GoogleNet

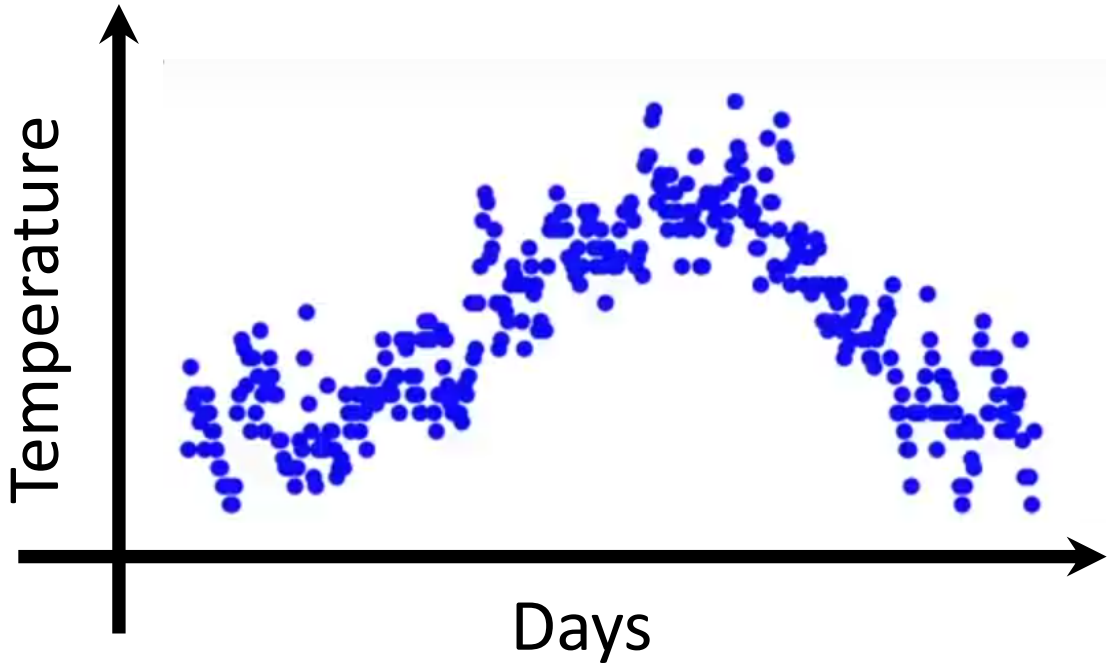
Optimization techniques

Exponentially Weighted Averages

- One of the popular algorithm for smoothing sequential data
- Also called Moving Average
- Weight the number of observations and using their average

- Example:

  Temperature θ over ‘n’ days



Source and reference https://towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d
https://medium.com/datadriveninvestor/exponentially-weighted-average-for-deep-neural-networks-39873b8230e9

Exponentially Weighted Averages

V_t : Moving average on day ‘t’

So, let

$V_0 = 0$

$V_1 = 0.9\ V_0 + 0.1\ \theta_1$

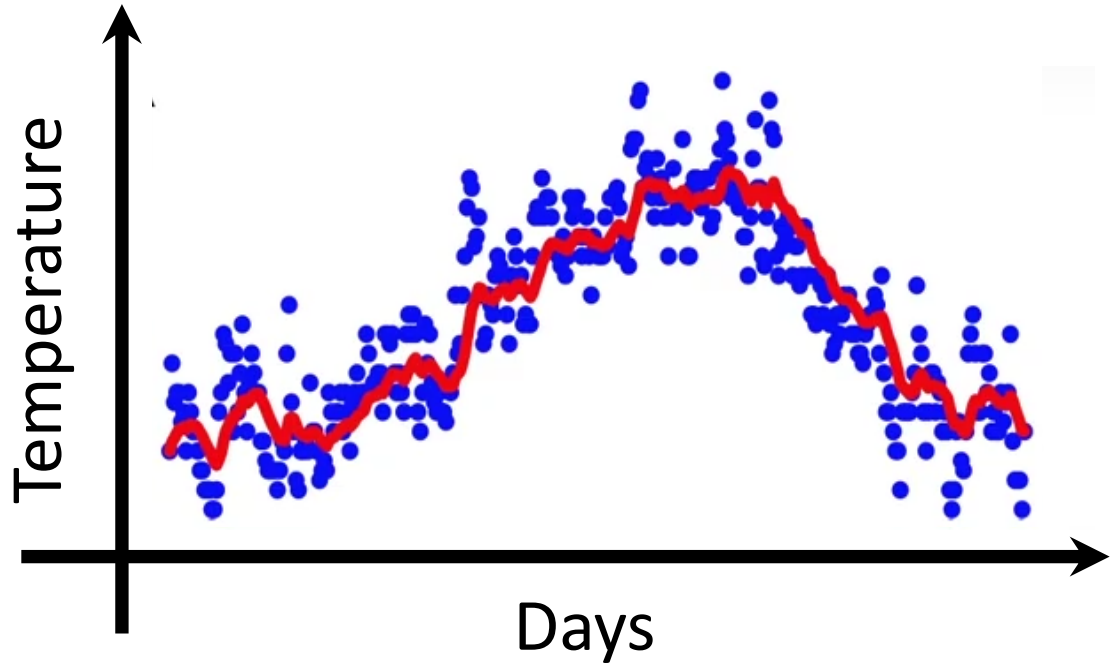
$V_2 = 0.9\ V_1 + 0.1\ \theta_2$

$V_3 = 0.9\ V_2 + 0.1\ \theta_3$

:

:

$V_t = 0.9\ V_{t-1} + 0.1\ \theta_t$



Source and reference https://towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d
https://medium.com/datadriveninvestor/exponentially-weighted-average-for-deep-neural-networks-39873b8230e9

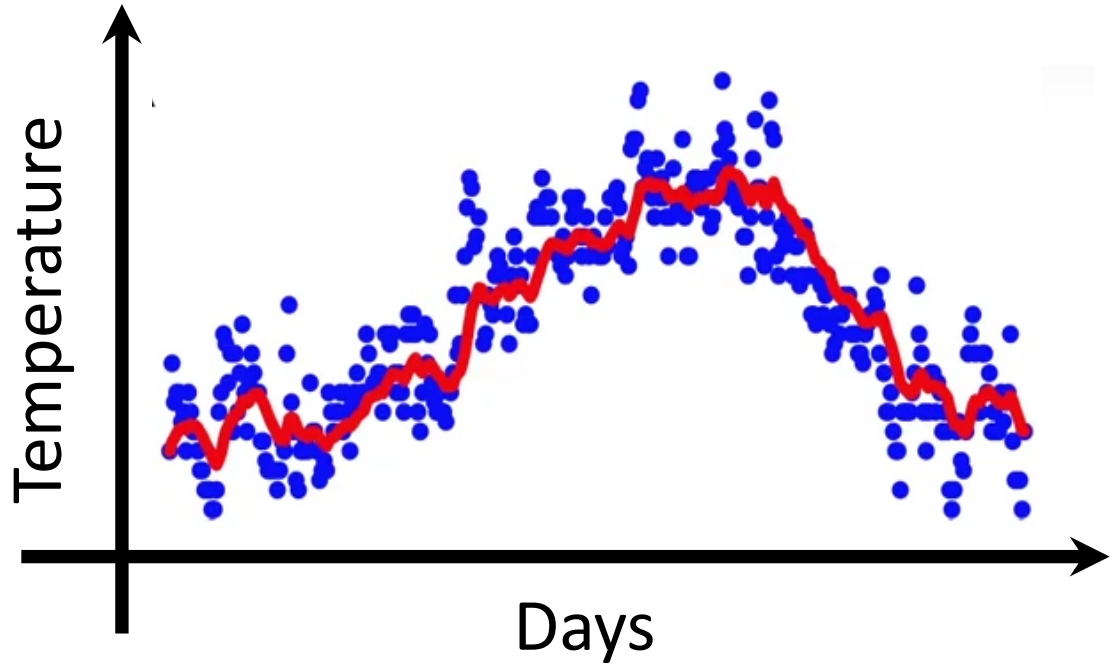
Exponentially Weighted Averages

$V_t = 0.9\ V_{t-1} + 0.1\ \theta_t$

If $\beta = 0.9$,

$$V_t = \beta\ V_{t-1} + (1-\beta)\ \theta_t$$

This equation gives the moving average shown by the red line.



Source and reference https://towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d
https://medium.com/datadriveninvestor/exponentially-weighted-average-for-deep-neural-networks-39873b8230e9

Exponentially Weighted Averages

$$V_t = \beta V_{t-1} + (1-\beta)\ \theta_t$$

V_t is approximate average over

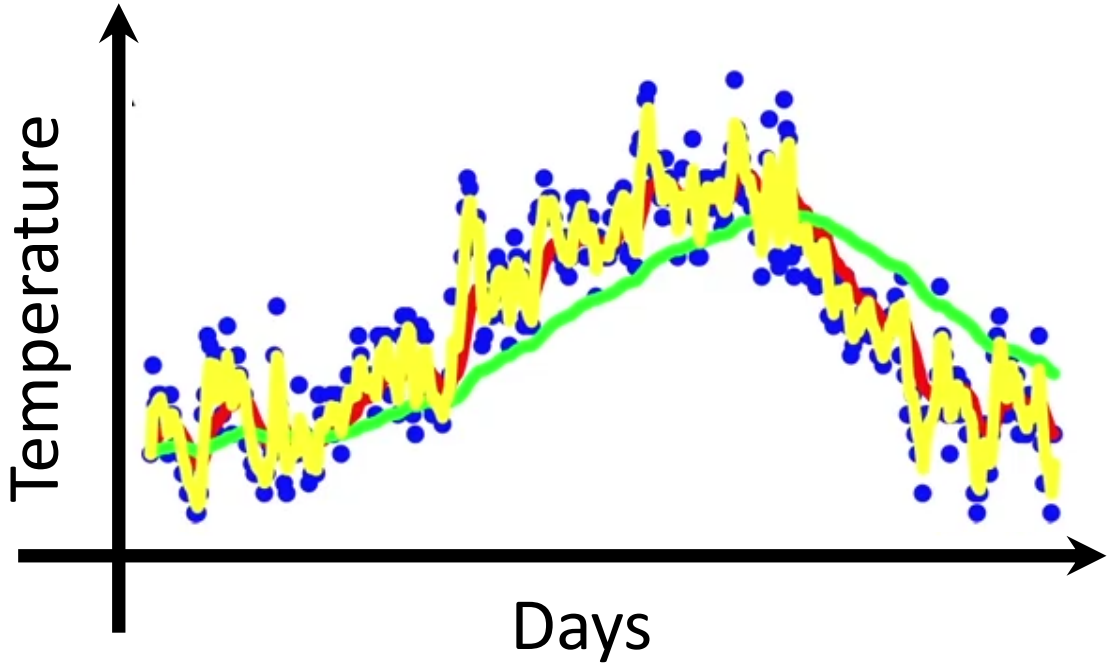
$\approx \frac{1}{1-\beta}$ days

So,

$\beta = 0.9$ is closer to 10 days temperature

$\beta = 0.98$ is closer to 50 days temperature

$\beta = 0.5$ is closer to 2 days temperature



Source and reference https://towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d
https://medium.com/datadriveninvestor/exponentially-weighted-average-for-deep-neural-networks-39873b8230e9

Exponentially Weighted Averages

What is Exponentially Weighted Averages doing?

$$V_t = \beta V_{t-1} + (1-\beta)\ \theta_t$$

For,

$$V_{100} = 0.9\ V_{99} + 0.1\ \theta_{100}$$
$$V_{99} = 0.9\ V_{98} + 0.1\ \theta_{99}$$

Substituting, V_{99}

$$V_{100} = 0.1\ \theta_{100} + 0.9\ (0.9\ V_{98} + 0.1\ \theta_{99})$$
$$V_{100} = 0.1\ \theta_{100} + 0.9\ (\ 0.1\ \theta_{99} + 0.9\ (0.9\ V_{97} + 0.1\ V_{98}))\ ..$$

Source and reference https://towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d
https://medium.com/datadriveninvestor/exponentially-weighted-average-for-deep-neural-networks-39873b8230e9

Optimizers – SGD with Momentum

- *"Compute the Exponentially weighted average of the gradients and use that gradient to update weights"* - ***Andrew NG***
- One of the most popular algorithms
- Helps to accelerate the gradient vectors in right direction and reduces oscillation
- Always faster than the SGD

Source and reference https://www.youtube.com/watch?v=k8fTYJPd3_I

Optimizers – SGD with Momentum

Algorithm:

At iteration t:

Calculate $dw\ and\ db$ on the current mini-batch

$V_{dw} = \beta\ V_{dw} + (1 - \beta)\ dw$ ➔ $V_t = \beta\ V_{t-1} + (1- \beta)\ \theta_t$

$V_{db} = \beta\ V_{db} + (1 - \beta)\ db$

Update w and b:

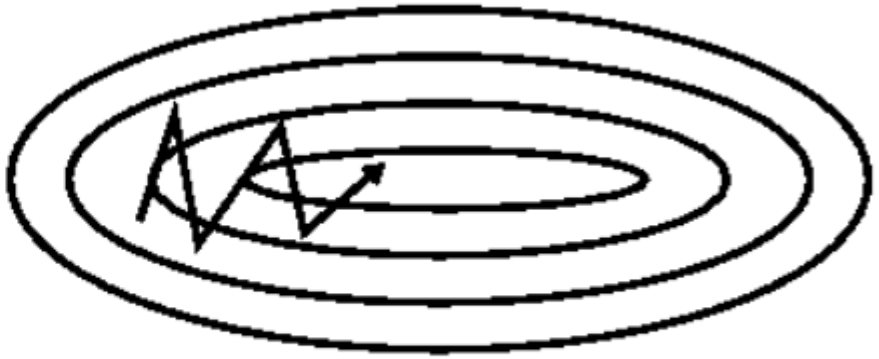
$w = w - \alpha\ V_{dw}$, $b = b - \alpha\ V_{db}$

Hyper-parameters: α, β

Source and reference https://www.youtube.com/watch?v=k8fTYJPd3_I

Optimizers – SGD with Momentum

SGD Without Momentum



SGD With Momentum

Faster convergence and reduced oscillation

Image Source and reference http://ruder.io/optimizing-gradient-descent/index.html#momentum

Optimizers – RMSProp

- Root Mean Square Propagation
- Unpublished adaptive learning method by Geoffery Hinton
- RMSProp also reduces oscillation but in a different way than Momentum
- RMSprop as well divides the learning rate by an exponentially decaying average of squared gradients.

Source and reference https://www.youtube.com/watch?v=k8fTYJPd3_I

Optimizers – RMSProp

Algorithm:

At iteration t:

Calculate $dw\ and\ db$ on the current mini-batch

$$S_{dw} = \beta_2 S_{dw} + (1 - \beta_2) dw^2$$

$$S_{db} = \beta_2 S_{db} + (1 - \beta_2) db^2$$

Squaring the derivatives

Update w and b:

$$w = w - \alpha \frac{dw}{\sqrt{S_{dw}}} \quad b = b - \alpha \frac{db}{\sqrt{S_{db}}}$$

Square root of derivatives

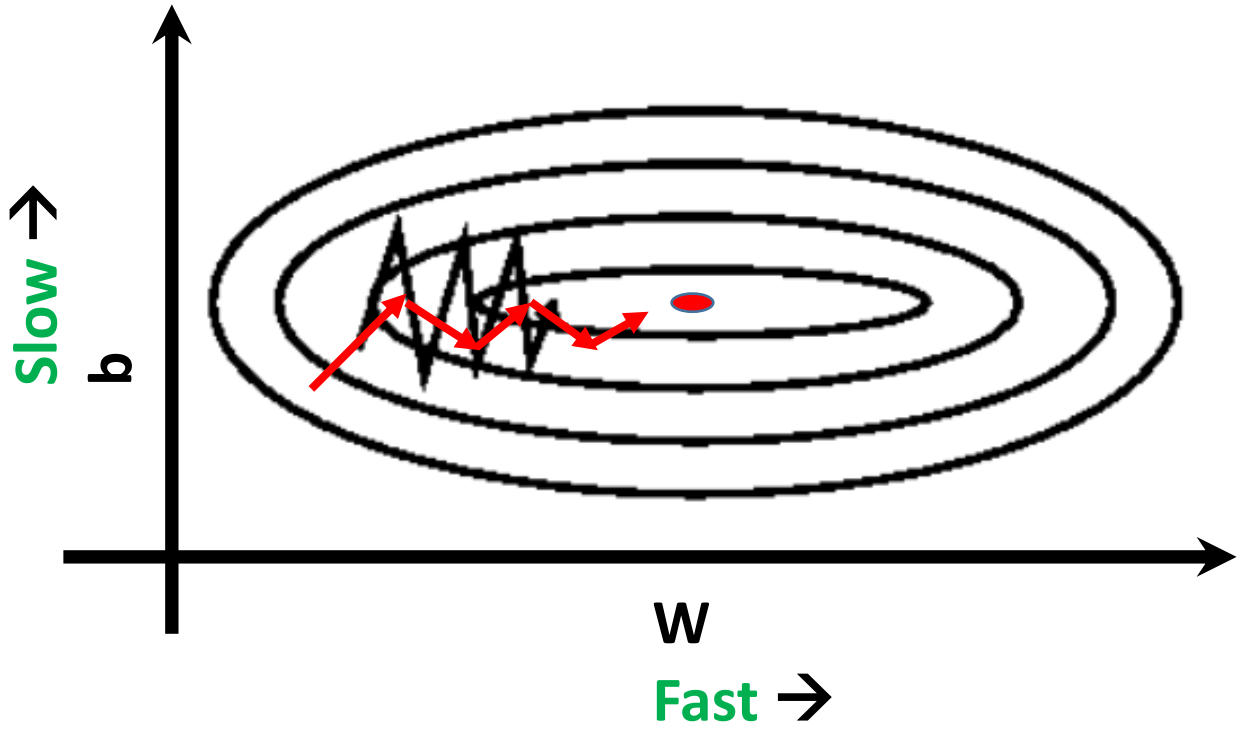
Source and reference https://www.youtube.com/watch?v=k8fTYJPd3_I

Optimizers – RMSProp

Intuition:

S_{dw} → Smaller number expected

S_{db} → Larger number expected



So,

$$w = w - \alpha \frac{dw}{\sqrt{S_{dw}}}, \quad b = b - \alpha \frac{db}{\sqrt{S_{db}}}$$

Smaller number
So, w is larger

Larger number
So, b is small

In Practice add ε :

$$w = w - \alpha \frac{dw}{\sqrt{S_{dw}} + \varepsilon}, \quad b = b - \alpha \frac{db}{\sqrt{S_{db}} + \varepsilon}$$

ε → small number, 10^{-8}

Source and reference https://www.youtube.com/watch?v=k8fTYJPd3_I

Optimizers – Adam

- Adam → Adaptive Moment Estimation
- Combination of RMSProp and Momentum
- Work well for a wide range of deep learning architecture

Source and reference: https://www.youtube.com/watch?v=JXQT_vxqwIs

Optimizers – Adam

Algorithm:

Initialize $V_{dw} = 0$, $V_{db} = 0$, $S_{dw} = 0$, $S_{db} = 0$

At iteration t:

Calculate dw and db on the current mini-batch

$V_{dw} = \beta_1 V_{dw} + (1 - \beta_1)\, dw$, $V_{db} = \beta_1 V_{db} + (1 - \beta_1)\, db$ ← From Momentum, β_1

$S_{dw} = \beta_2 S_{dw} + (1 - \beta_2)\, dw^2$, $S_{db} = \beta_2 S_{db} + (1 - \beta_2)\, db^2$ ← From RMSProp, β_2

Update w and b:

$$w = w - \alpha \frac{V_{dw}}{\sqrt{S_{dw}} + \varepsilon}, \quad b = b - \alpha \frac{V_{db}}{\sqrt{S_{db}} + \varepsilon}$$

Source and reference: https://www.youtube.com/watch?v=JXQT_vxqwIs

Optimizers – Adam

In practice: Bias correction is required as V_{dw}, V_{db}, S_{dw}, S_{db} are initialized to 0 and are biased towards zero. Hence, a bias correction is required as follows:

$$V'_{dw} = \frac{Vdw}{(1-\beta_1)}, \quad V'_{db} = \frac{V_{db}}{(1-\beta_1)}$$

$$S'_{dw} = \frac{S_{dw}}{(1-\beta_2)}, \quad S'_{db} = \frac{S_{db}}{(1-\beta_2)}$$

Update w and b:

$$w = w - \alpha \frac{V'_{dw}}{\sqrt{S'_{dw}} + \varepsilon}, \quad b = b - \alpha \frac{V'_{db}}{\sqrt{S'_{db}} + \varepsilon}$$

Source and reference: https://www.youtube.com/watch?v=JXQT_vxqwIs

Optimizers – Adam

Hyper parameter guide:

α (Learning rate)→ should be tunned, start with 0.001

β_1 (Momentum term) → 0.9 (dw)

β_2 (moving weighted average) → 0.999 (dw^2)

ε → 10^{-8}

Optimization Demo:

https://vis.ensmallen.org/

Source and reference: https://www.youtube.com/watch?v=JXQT_vxqwIs

ImageNet Dataset:

- 15+ million labelled high-resolution images
- 22000 categories
- ILSVRC (**Large Scale Visual Recognition Challenge**) used a subset of ImageNet:
  - ~1000 images per category
  - 1000 categories
  - Train: 1.2 million images
  - Validation: 50k images
  - Test : 150k images

Source and reference: http://www.image-net.org/
http://cvml.ist.ac.at/courses/DLWT_W17/material/AlexNet.pdf

ImageNet Dataset:

Source and reference https://cs.stanford.edu/people/karpathy/cnnembed/cnn_embed_full_1k.jpg

ImageNet Dataset Results:

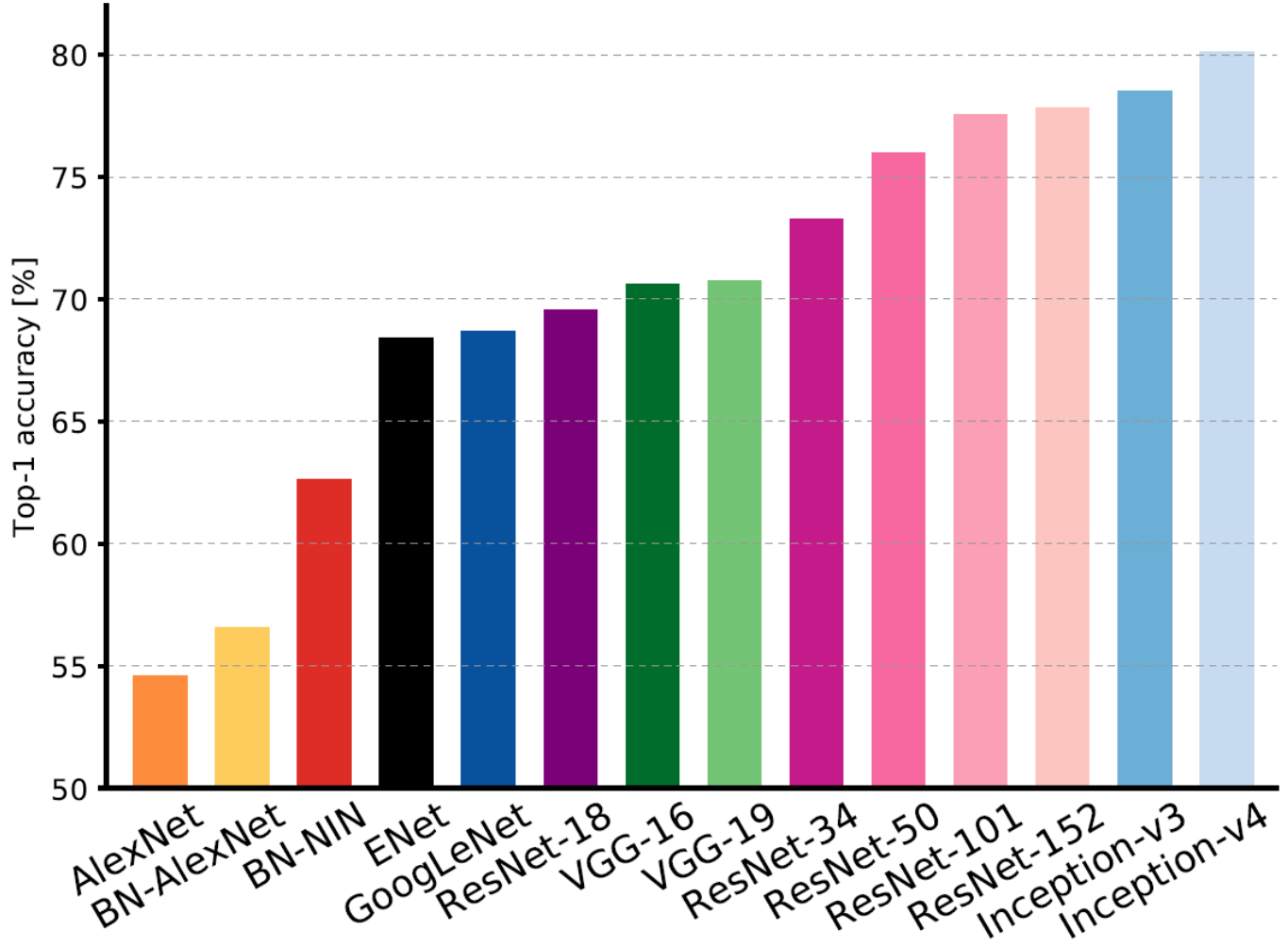


Image Source and reference: An Analysis of Deep Neural Network Models for Practical Applications, 2017, https://arxiv.org/abs/1605.07678

Transfer Learning

- Knowledge acquired while solving one task, can be used to solve related tasks.
- Example:
  - You know how to ride a Bi-cycle → You can learn how to ride a Motorbike
  - You know how to use a Tablet → You can easily learn how to use a Laptop/desktop

- Similar to the way humans apply knowledge acquired from one task to solve a new but similar/related task.
- We learned how to read in Year-1 in literacy class. Reading skills acquired in the literacy classes made it easy to understand Physics in Year-9.

Source and reference: https://cs231n.github.io/transfer-learning/
https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a

Transfer Learning Benefits

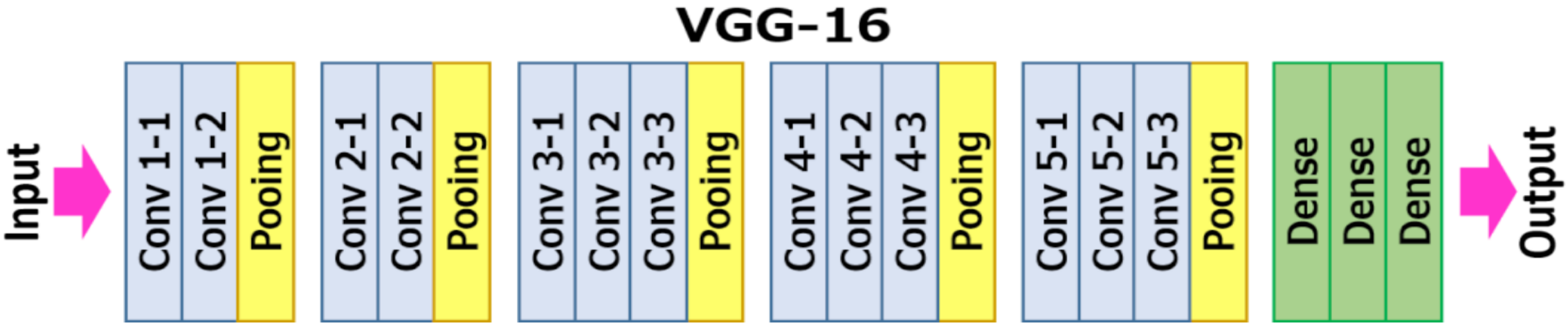
1. **Less training data required:** Don't have enough data to train a Deep Learning model from scratch. Model trained using a large (similar) dataset can be used.

2. **Faster training** : Training can converge faster, due the use to existing knowledge (weights) to start with rather than from scratch.

3. **Better model generalization**: Model is trained to identify features which can be applied to new contexts.

Source and reference: https://cs231n.github.io/transfer-learning/
https://missinglink.ai/guides/neural-network-concepts/transfer-learning-overview/

Transfer Learning Strategies



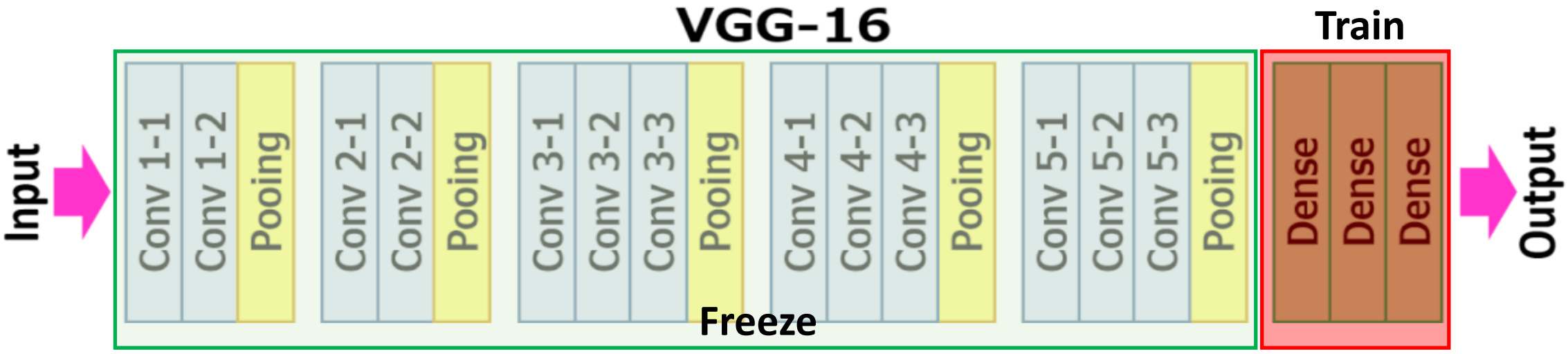
<u>Option-1:</u> (VGG-16 considered as an example)

Use pre-trained (ImageNet) model for prediction, without any training.

→Useful when your dataset distribution is similar to ImageNet, with small number of samples.

Source and reference: https://cs231n.github.io/transfer-learning/
https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a

Transfer Learning Strategies



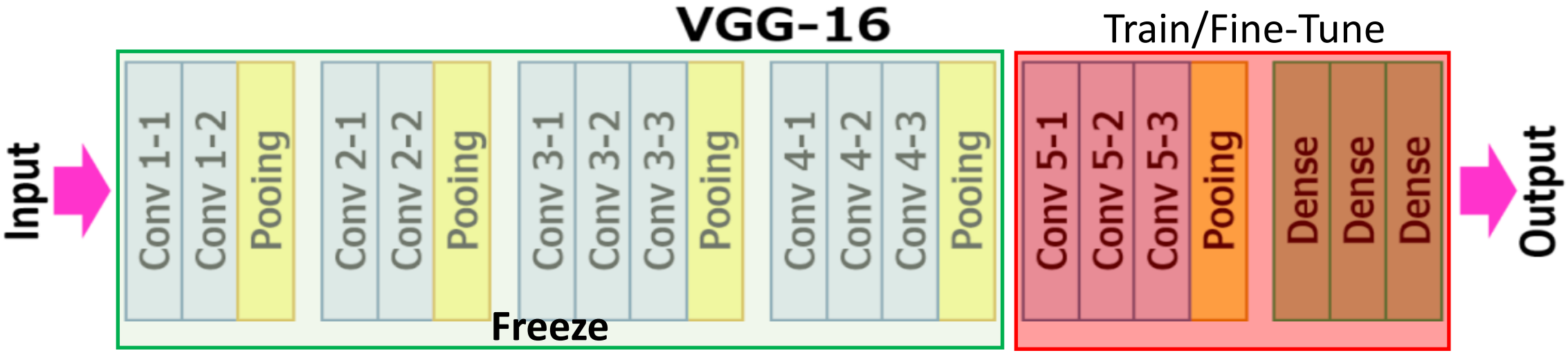
<u>Option-2:</u> (VGG-16 considered as an example)

Train Full-Connected layer, Use CONV layers for feature extraction

→Useful when your dataset distribution is similar to ImageNet (or original dataset), but number of classes are different and your dataset is small.

Source and reference: https://cs231n.github.io/transfer-learning/
https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a

Transfer Learning Strategies



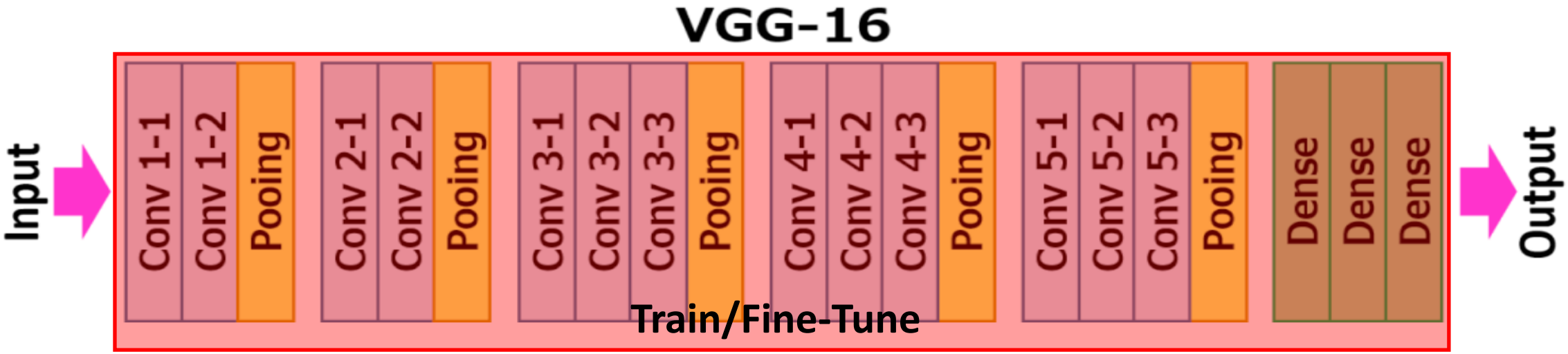
<u>Option-3:</u> (VGG-16 considered as an example)

Partially Train CONV layers (usually last layer(s) which have specialised features) + Full Connection (FC) layer (with modifications)

→Useful when your dataset distribution is not similar to ImageNet (or original dataset), number of classes are different and your dataset is small.

Source and reference: https://cs231n.github.io/transfer-learning/
https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a

Transfer Learning Strategies



<u>Option-4:</u> (VGG-16 considered as an example)

Train all the CONV layers + Full Connection (FC) layer (with modifications)

→Useful when your dataset distribution is not similar to ImageNet, number of classes are different, your is dataset large and the task is complex.

Source and reference: https://cs231n.github.io/transfer-learning/
https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a

Classic CNN Architectures

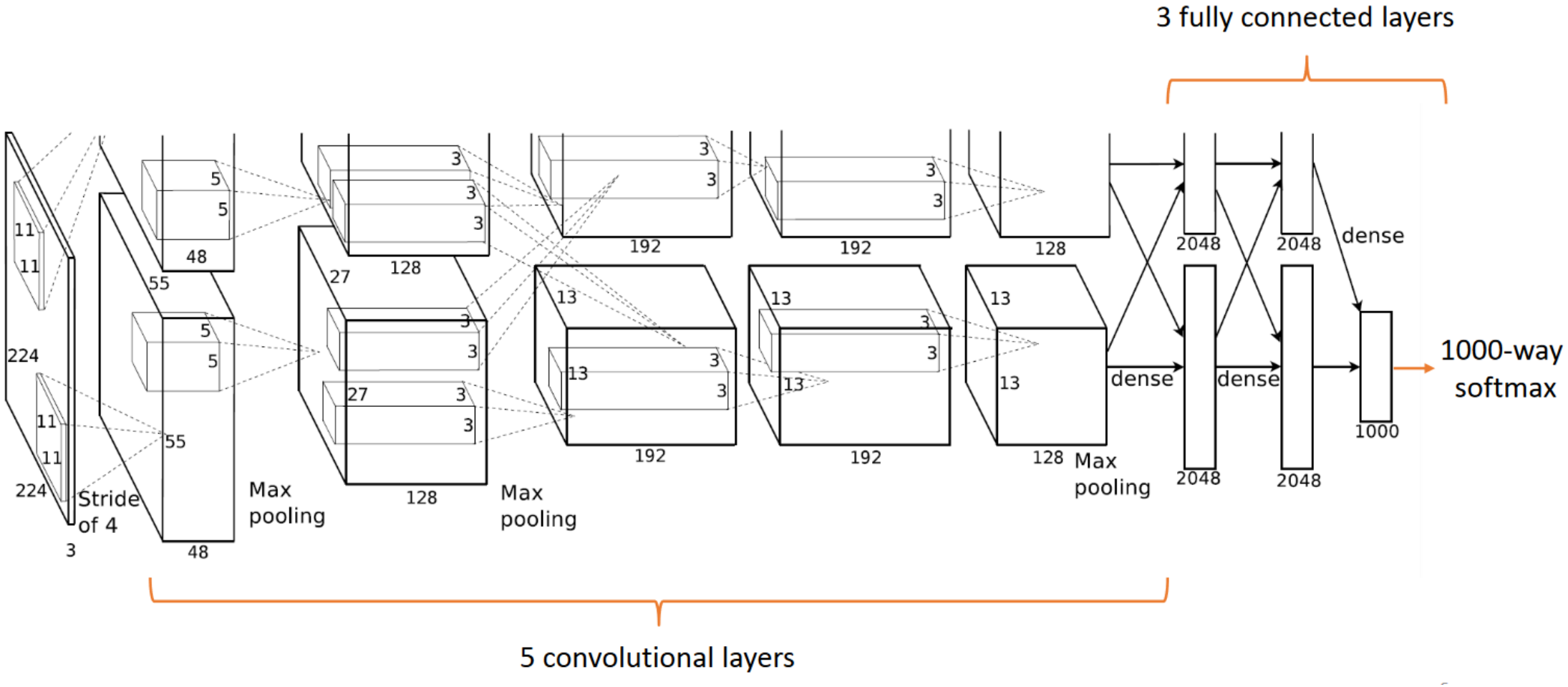
Case Study: AlexNet

- Similar architecture as LeNet by Yann LeCunn et al. but deeper with more layers
- Simple architecture:
  - CONV : 5 layers
  - FC: 3 layer
  - Max pooling
  - Dropout

- Accuracy: top-5 test error rate of 15.3%
- Winner of ILSVRC 2012!
- First CNN to be successful on a very big dataset!

Source and reference: http://cvml.ist.ac.at/courses/DLWT_W17/material/AlexNet.pdf

Case Study: AlexNet



Source and reference: http://cvml.ist.ac.at/courses/DLWT_W17/material/AlexNet.pdf

Case Study: AlexNet

Input: 224x224x3 image

CONV1 → CONV2 → CONV3 → CONV4 → CONV5 → FC1 → FC2 → FC3

Filters: 96 **Dim**: 11x11 **Stride**: **4** **Pad:** 0	**Filters**: 256 **Dim**: 5x5 **Stride**: **1** **Pad:** 2	**Filters**: 384 **Dim**: 3x3 **Stride**: **1** **Pad: 1**	**Filters**: 384 **Dim**: 3x3 **Stride**: **1** **Pad: 1**	**Filters**: 256 **Dim**: 3x3 **Stride**: **1** **Pad: 1**	**4096 Neuron**	**4096 Neuron**	**1000 Neuron**

Activations: Relu after each CONV and FC layer

Optimizer: SGD with Momentum

Regularization: Dropout in FC1 and FC2

Total Trainable parameter: ~60Million

Training settings: 2 X Nvidia GTX 580 3GB GPUs for 5-6days!

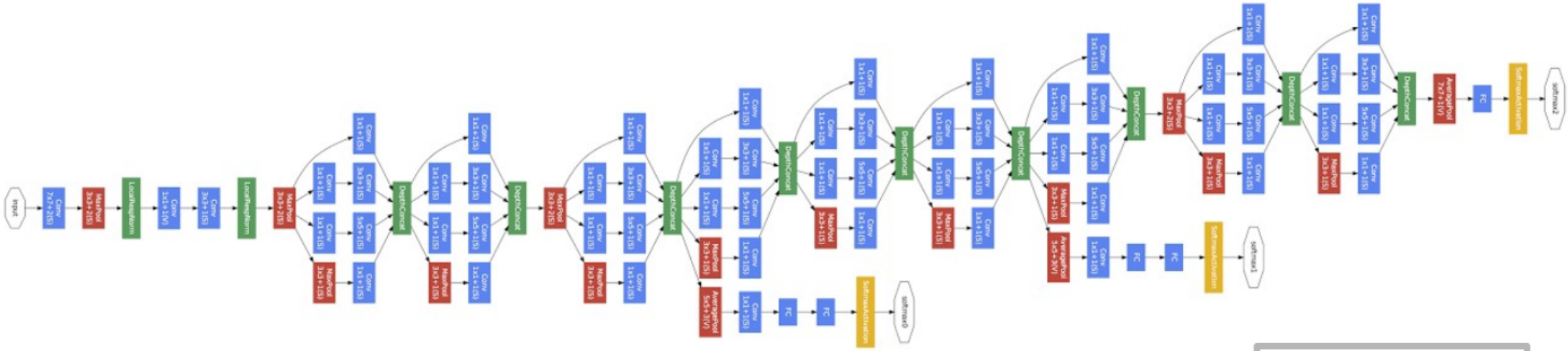
Source and reference: http://cvml.ist.ac.at/courses/DLWT_W17/material/AlexNet.pdf

Case Study: GoogleNet/Inception(2014)

- Accuracy: top-5 test error rate of 6.7%
- Close to human level performance
- Winner of ILSVRC 2014!
- 22 layer Deep CNN
- Number of trainable parameters: 4 Million (Alexnet ~ 60M), Significantly reduced
- A novel inception module was introduced.
- Optimizer: RMSProp

Source and reference: https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

Case Study: GoogleNet/Inception(2014)

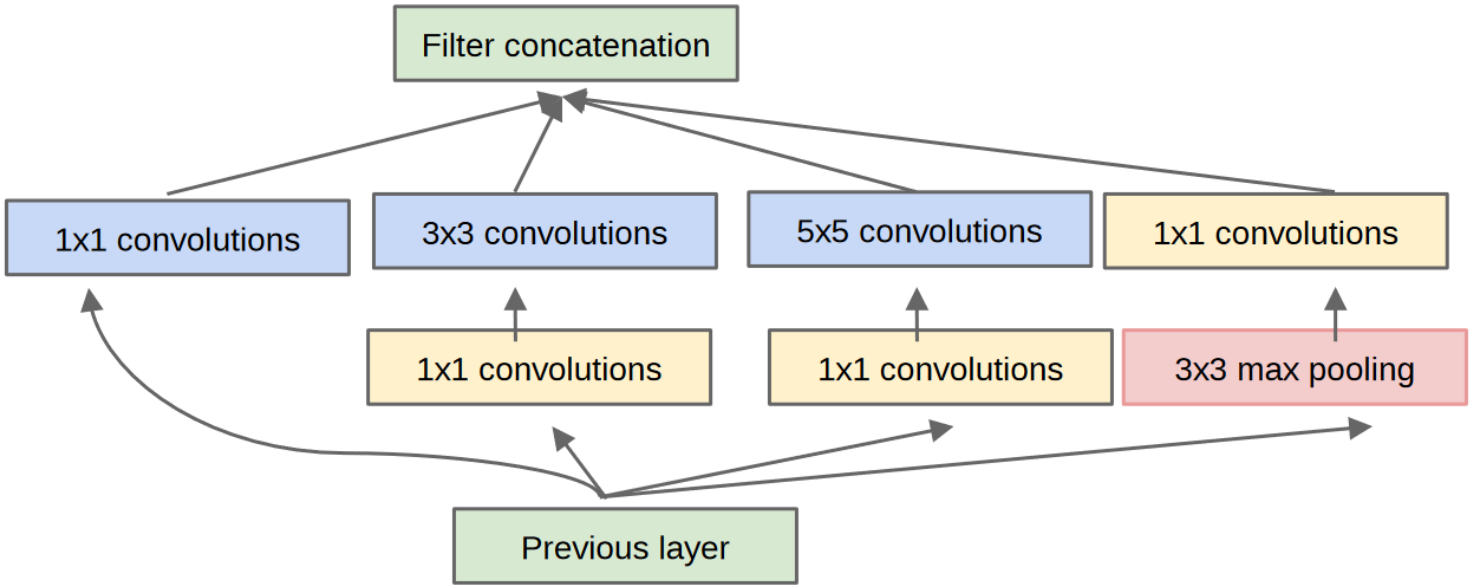


Convolution
Pooling
Softmax
Other

Image Source and reference: https://www.cs.unc.edu/~wliu/papers/GoogLeNet.pdf

Case Study: GoogleNet/Inception(2014)

Inception Module



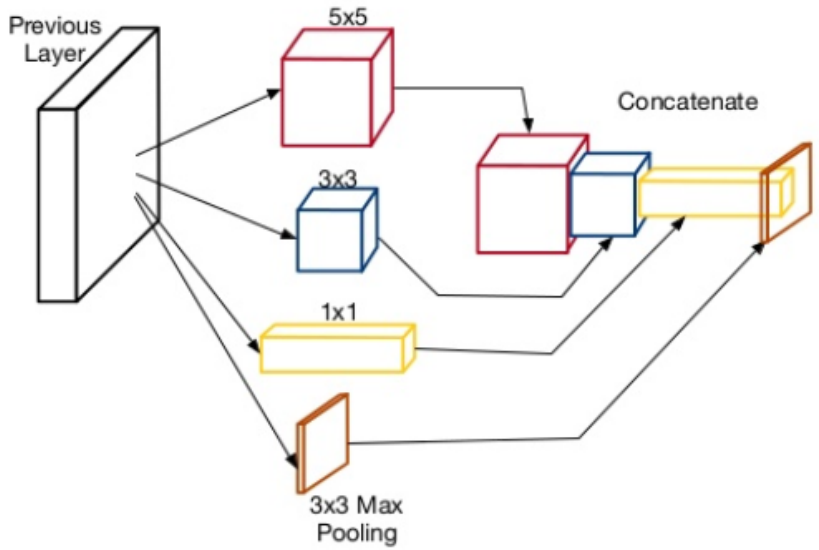
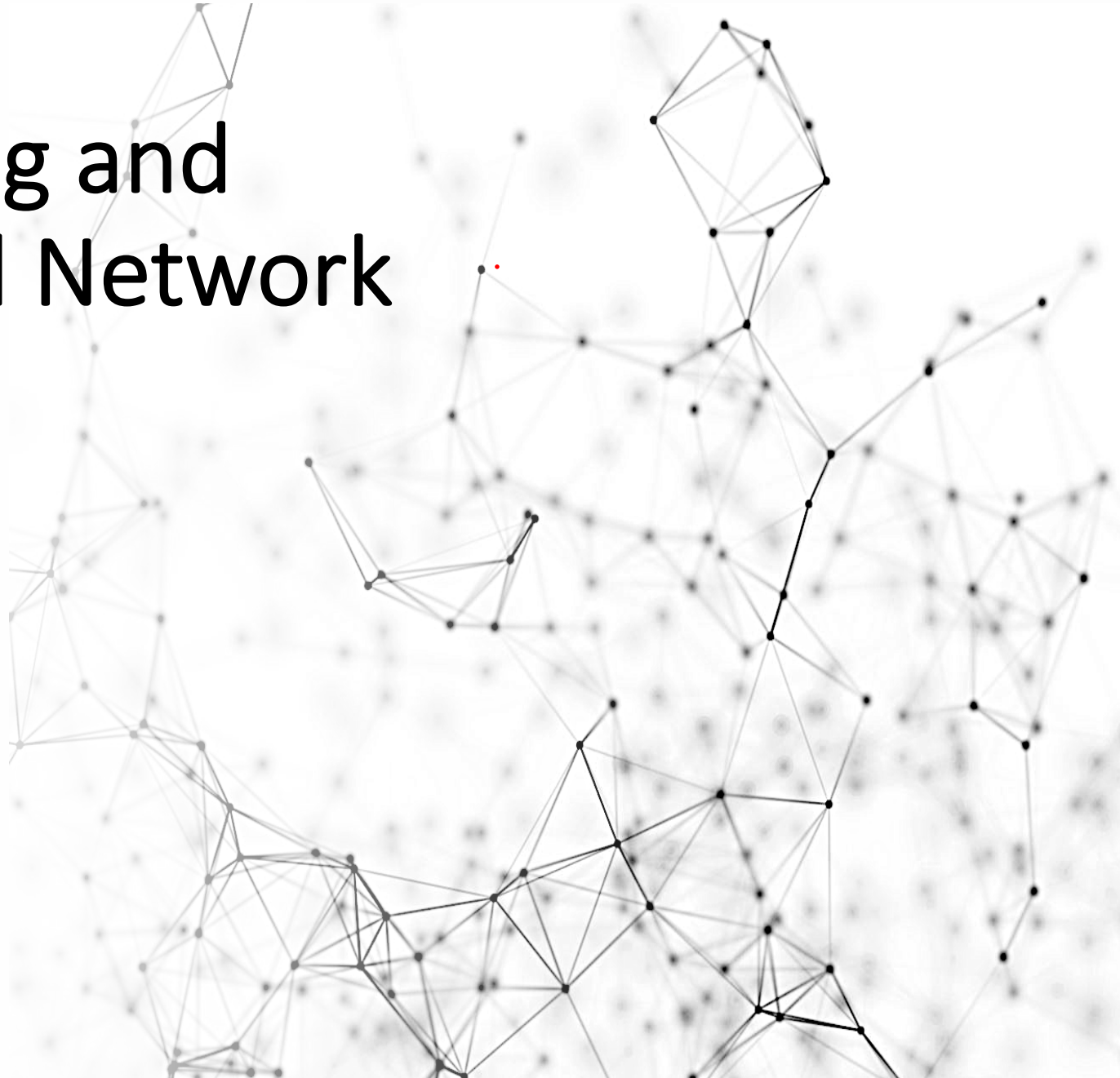


Image Source and reference: https://www.cs.unc.edu/~wliu/papers/GoogLeNet.pdf

42028: Deep Learning and Convolutional Neural Network



Week-8 Lecture

Object Detection -1

Outline

- Introduction to Object Detection
- Datasets and Performance metrics
- Different frameworks
- Classification and Localization task
- Object detection as a regression problem
- Object detection as a classification problem
- Case study: RCNN family
- Image annotation

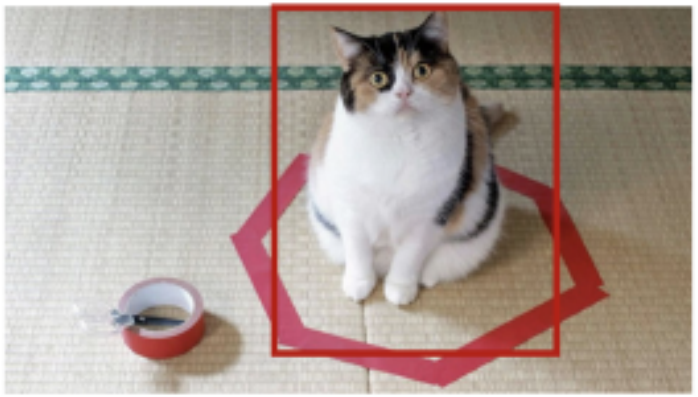
Introduction

1

Is this image of Cat or not?

Image classification problem

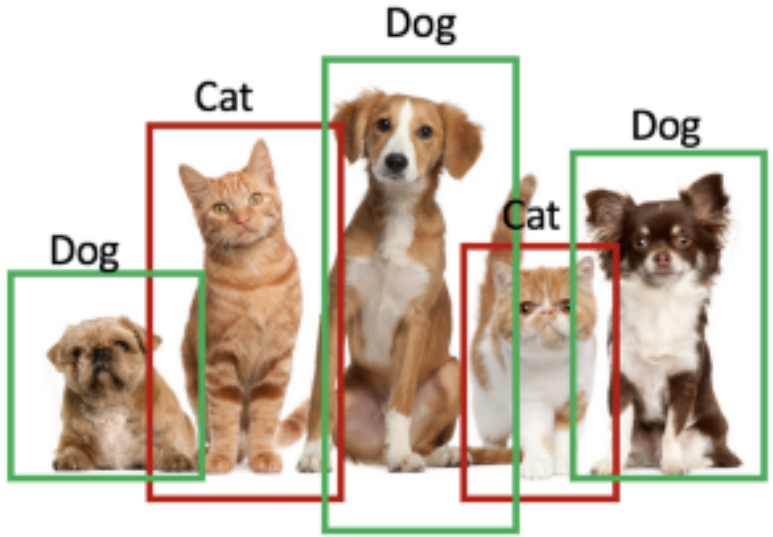
2



Where is Cat?

Classification with localization problem

3



Which animals are there in image and where?

Object detection problem

Images source: https://towardsdatascience.com/evolution-of-object-detection-and-localization-algorithms-e241021d8bad

Datasets and Performance Metrics

- The **PASCAL Visual Object Classification (PASCAL VOC)** is a popular dataset for object detection, classification and segmentation.
- 20 categories
- Link: http://host.robots.ox.ac.uk/pascal/VOC/

- **ImageNet** has released an object detection dataset in 2013
- Train set: 500,000 images, 200 categories.
- Not very popular due to large number of classes and dataset size!
- Large number classes complicates the task

Images source: https://towardsdatascience.com/evolution-of-object-detection-and-localization-algorithms-e241021d8bad

Datasets and Performance Metrics

Dataset Comparison

		PASCAL VOC 2012	ILSVRC 2014
Number of object classes		20	**200**
Training	Num images	5717	456567
	Num objects	13609	478807
Validation	Num images	5823	20121
	Num objects	13841	55502
Testing	Num images	10991	40152
	Num objects	---	---

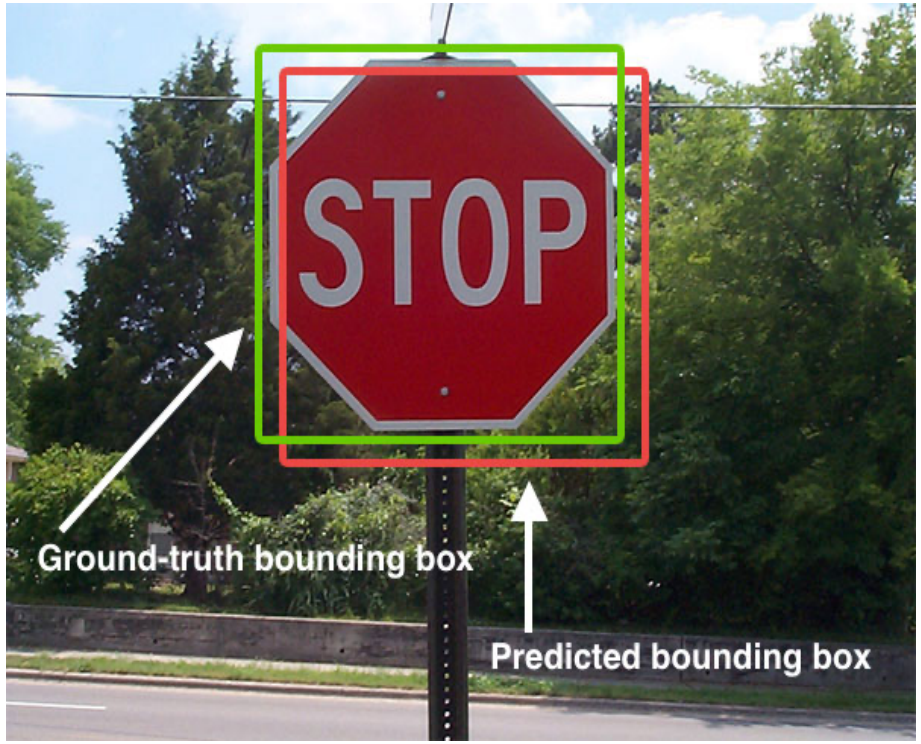
Images source: http://image-net.org/challenges/LSVRC/2014/

Datasets and Performance Metrics

- **Intersection over Union (IoU):**

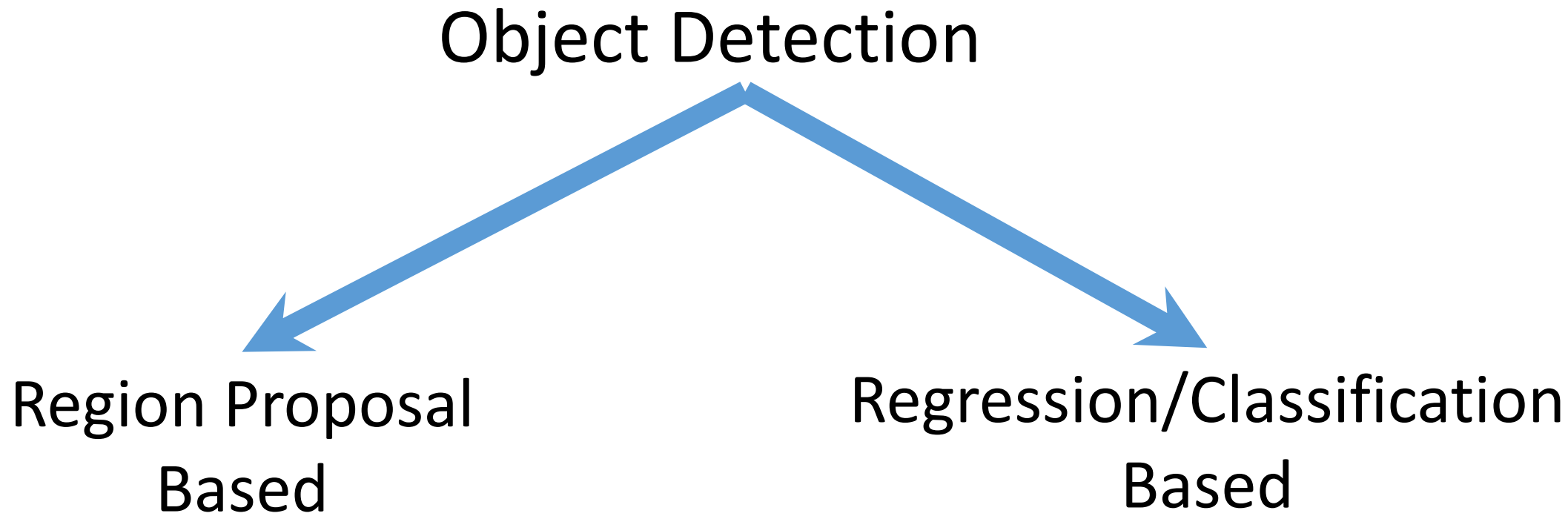
Intersection over Union is a metric used for the evaluation of an object detector, i.e. how good is the predicted bounding box for an object detected closely matches

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$



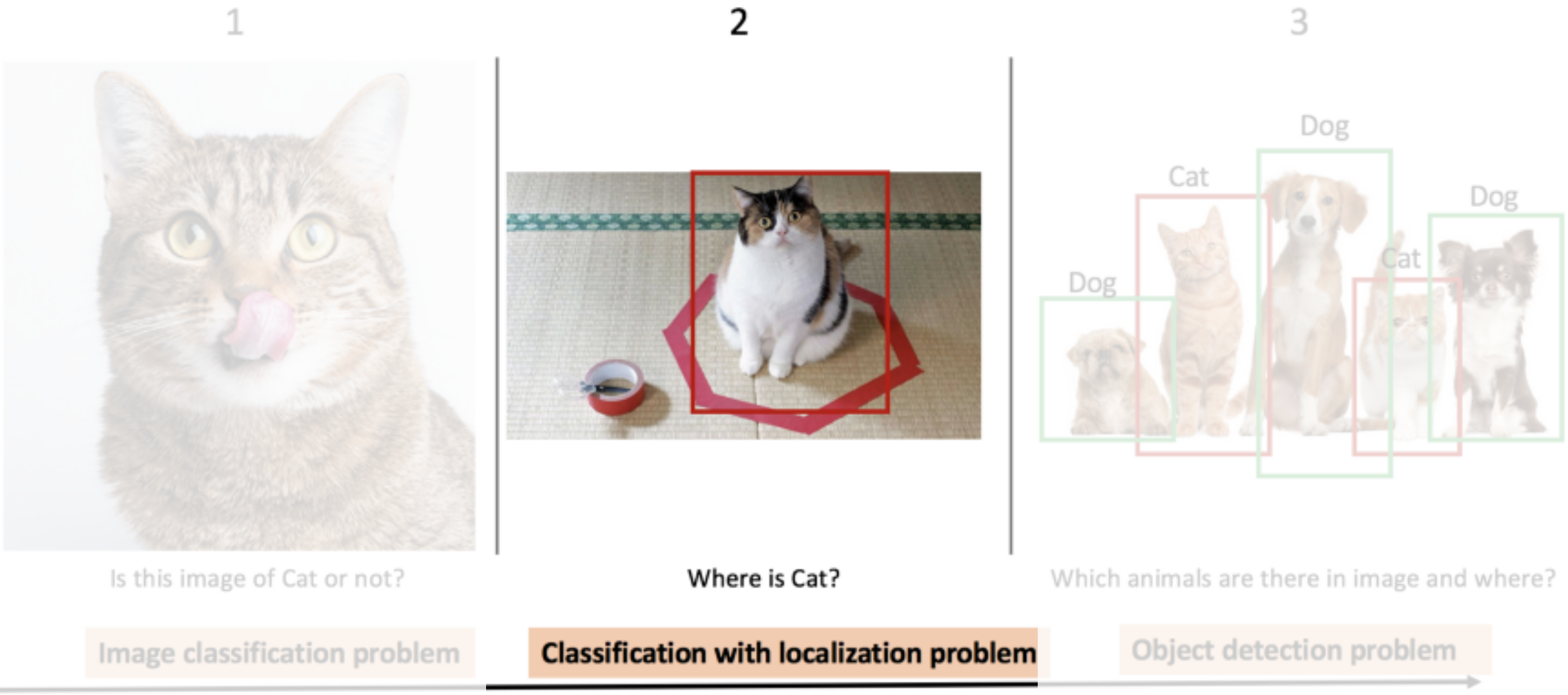
Reference: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

Current frameworks



Images source: https://towardsdatascience.com/evolution-of-object-detection-and-localization-algorithms-e241021d8bad

Task: Classification with Localization



Images source: https://towardsdatascience.com/evolution-of-object-detection-and-localization-algorithms-e241021d8bad

Task: Classification with Localization

Classification Task:

Input : Image

Output: Label

Performance Evaluation: Accuracy

Output : Dog

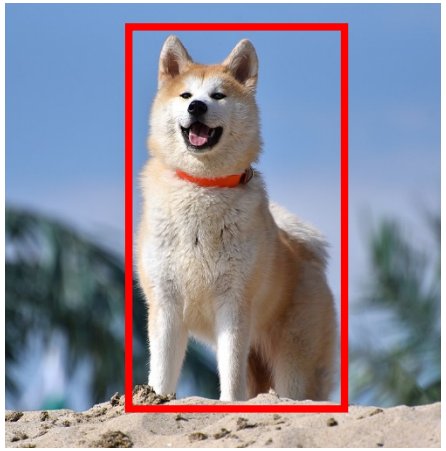
Localization Task:

Input : Image

Output: Bounding Box in the image

(x, y, Ht, Wd) or (x, y, x', y')

Performance Evaluation: IoU



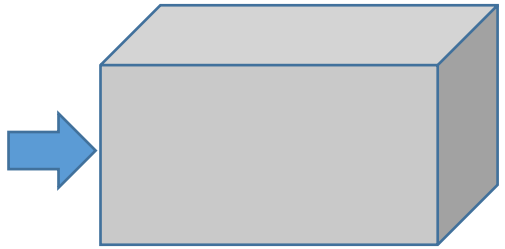
Output : (x, y, Ht, Wd)

Source and Reference: http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf

Image source: https://en.wikipedia.org/wiki/Akita_(dog)

Localization as a regression problem

Input Image



CNN

Output : 4 numbers
(x', y', Ht', Wd')

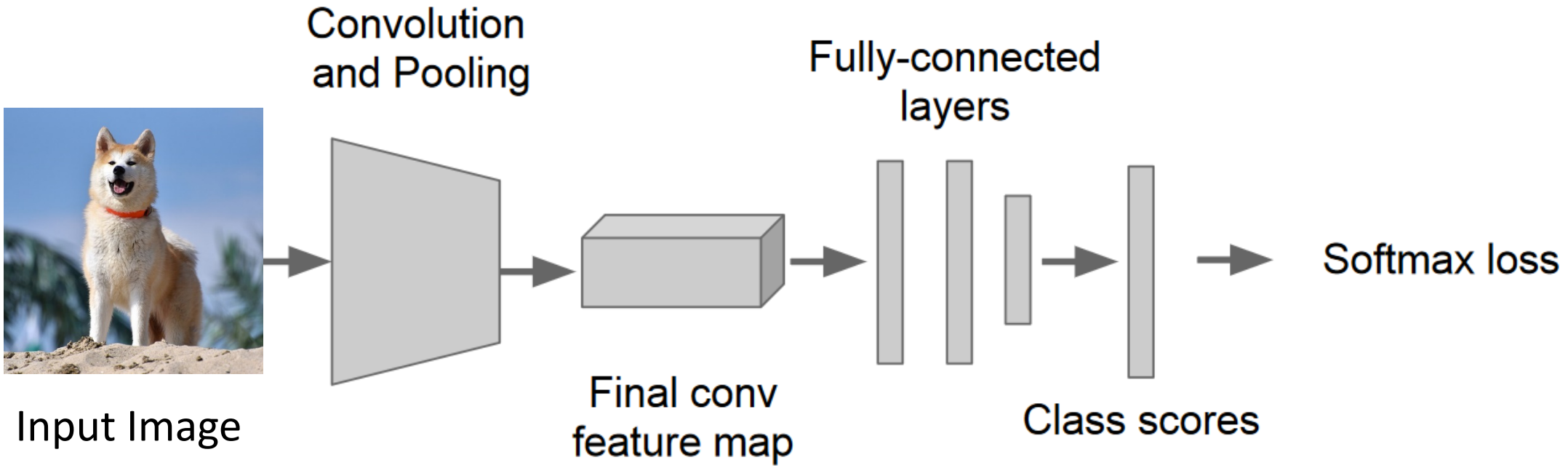
Ground Truth: 4 numbers
(x, y, Ht, Wd)

Calculate Loss
L2 Loss

Source and Reference: http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf

Image source: https://en.wikipedia.org/wiki/Akita_(dog)

Localization as a regression problem



Source and Reference: http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf

Image source: https://en.wikipedia.org/wiki/Akita_(dog)

Localization as a regression problem

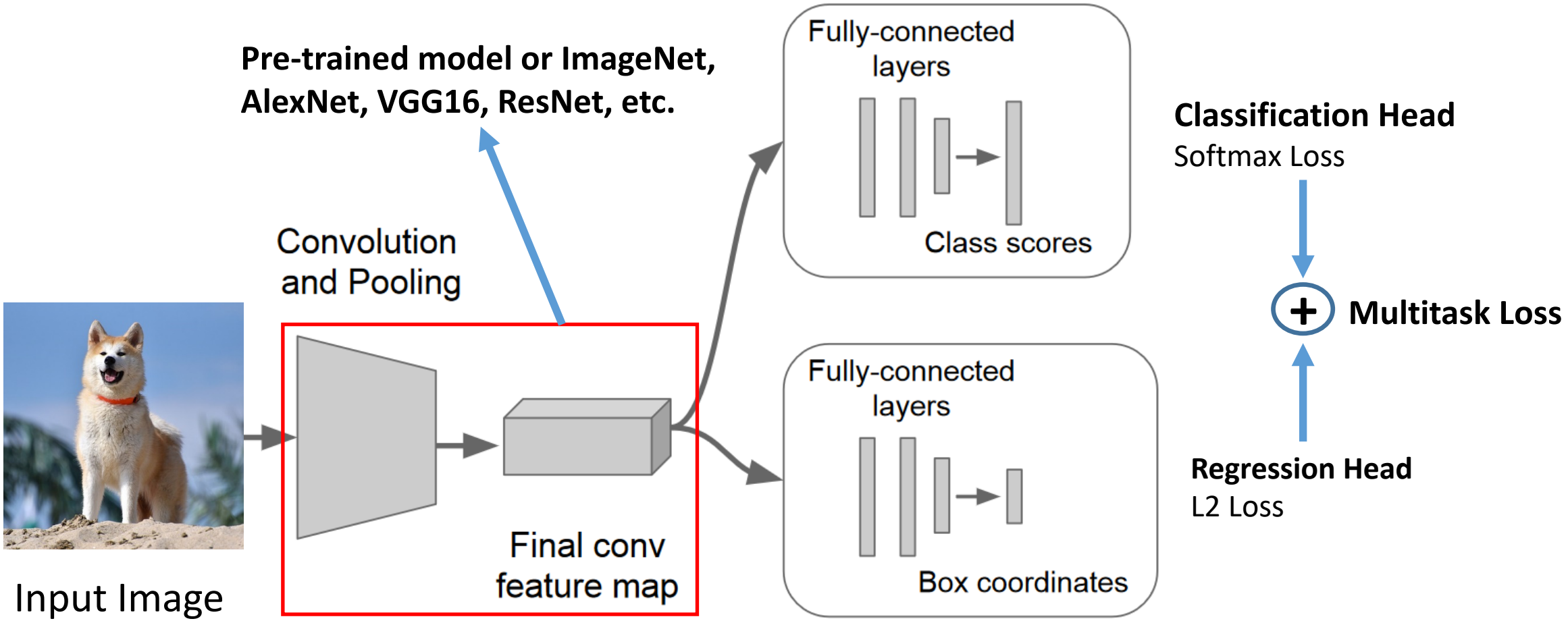
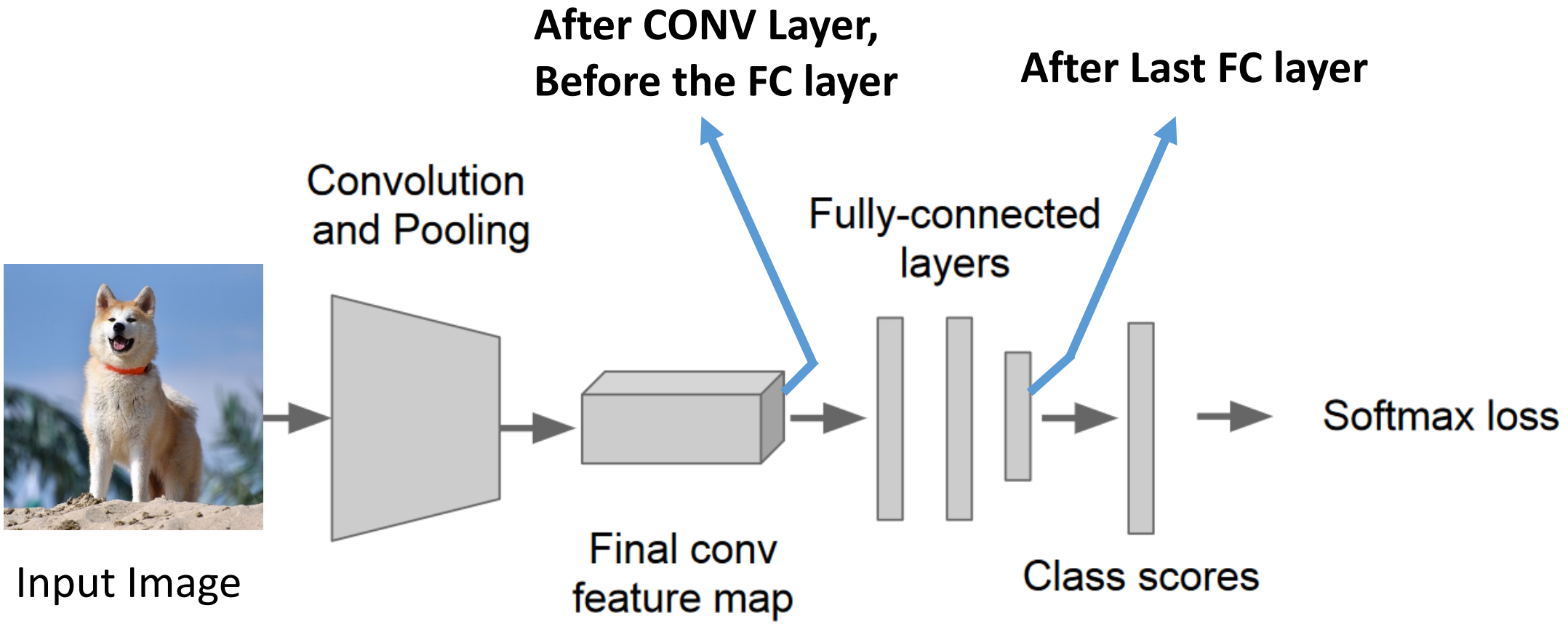


Image Source and Reference: http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf

Potential locations for Regression head in CNN



Source and Reference: http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf

Image source: https://en.wikipedia.org/wiki/Akita_(dog)

Task: Object Detection Problem

1

Is this image of Cat or not?

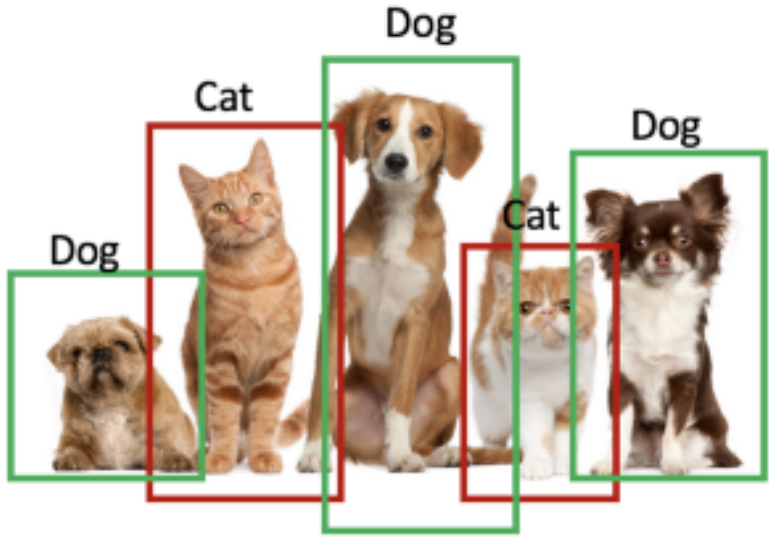
Image classification problem

2

Where is Cat?

Classification with localization problem

3

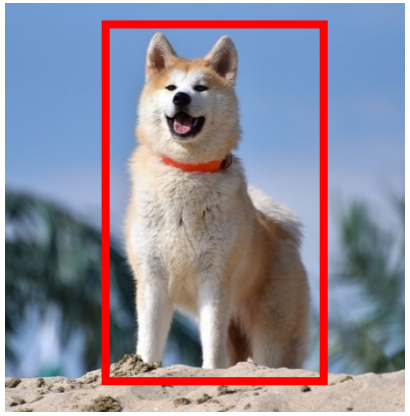


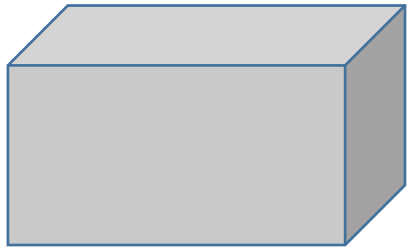
Which animals are there in image and where?

Object detection problem

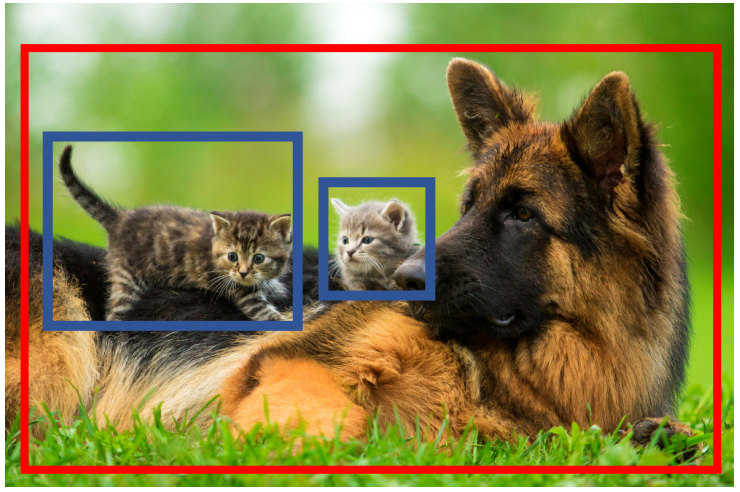
Images source: https://towardsdatascience.com/evolution-of-object-detection-and-localization-algorithms-e241021d8bad

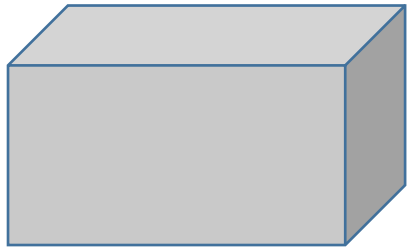
Detection as a regression problem





Output : Dog, (x, y, Ht, Wd)





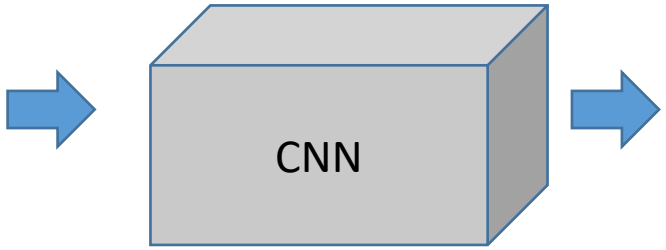
Output :
Dog, (x, y, Ht, Wd)
Cat, (x, y, Ht, Wd)
Cat, (x, y, Ht, Wd)

Source and Reference: http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf

Image source: https://en.wikipedia.org/wiki/Akita_(dog)
http://www.kamloopspropertyforsale.com/blog/does-owning-a-cat-depreciate-your-home.html

Detection as a classification problem

1. Apply Sliding Window technique
2. Apply CNN to different Windows and get a prediction



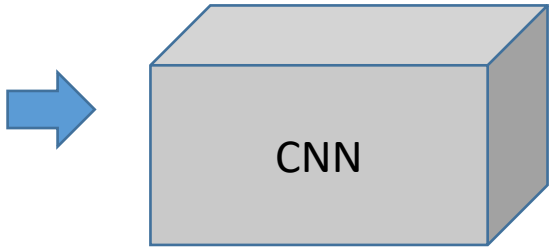
Output :
Dog? No
Cat? No
Background? Yes

Source and Reference: http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf

Image source: https://en.wikipedia.org/wiki/Akita_(dog)
http://www.kamloopspropertyforsale.com/blog/does-owning-a-cat-depreciate-your-home.html

Detection as a classification problem

1. **Apply Sliding Window technique**
2. **Apply CNN to different Windows and get a prediction**



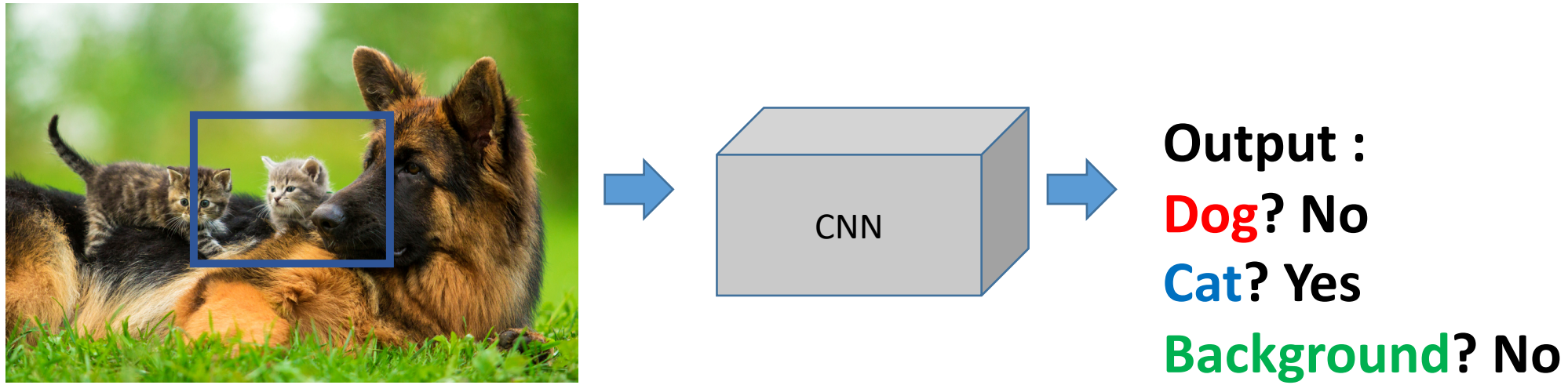
Output :
Dog? No
Cat? Yes
Background? No

Source and Reference: http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf

Image source: https://en.wikipedia.org/wiki/Akita_(dog)
http://www.kamloopspropertyforsale.com/blog/does-owning-a-cat-depreciate-your-home.html

Detection as a classification problem

1. Apply Sliding Window technique
2. Apply CNN to different Windows and get a prediction



Source and Reference: http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf

Image source: https://en.wikipedia.org/wiki/Akita_(dog)
http://www.kamloopspropertyforsale.com/blog/does-owning-a-cat-depreciate-your-home.html

Detection as a classification problem

Issue with Sliding Window technique

1. Apply CNN on large number of windows
2. Multiple scale and locations of windows
3. Inaccurate bounding boxes
4. Computationally expensive

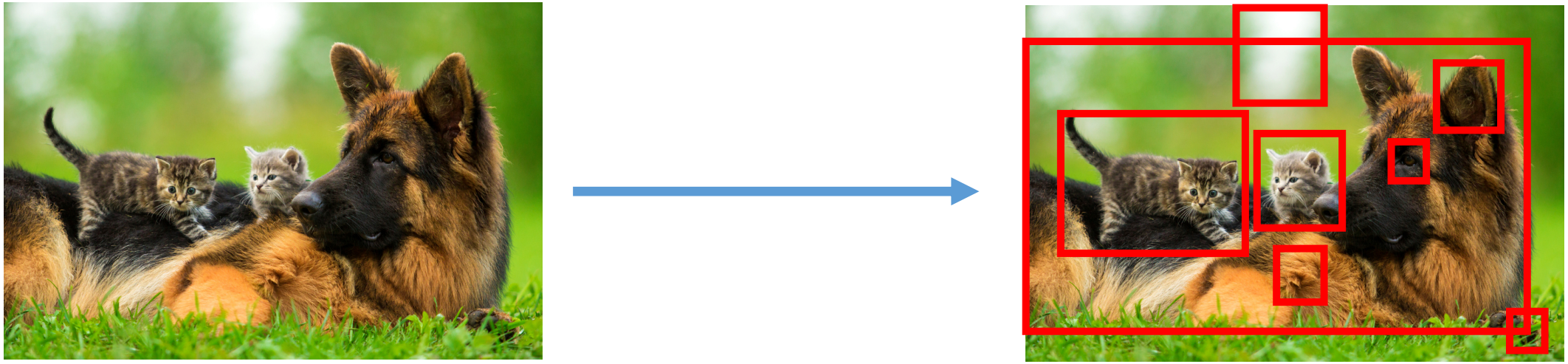
Source and Reference: http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf

Image source: https://en.wikipedia.org/wiki/Akita_(dog)
http://www.kamloopspropertyforsale.com/blog/does-owning-a-cat-depreciate-your-home.html

Detection as a classification problem

Region Proposal Technique:

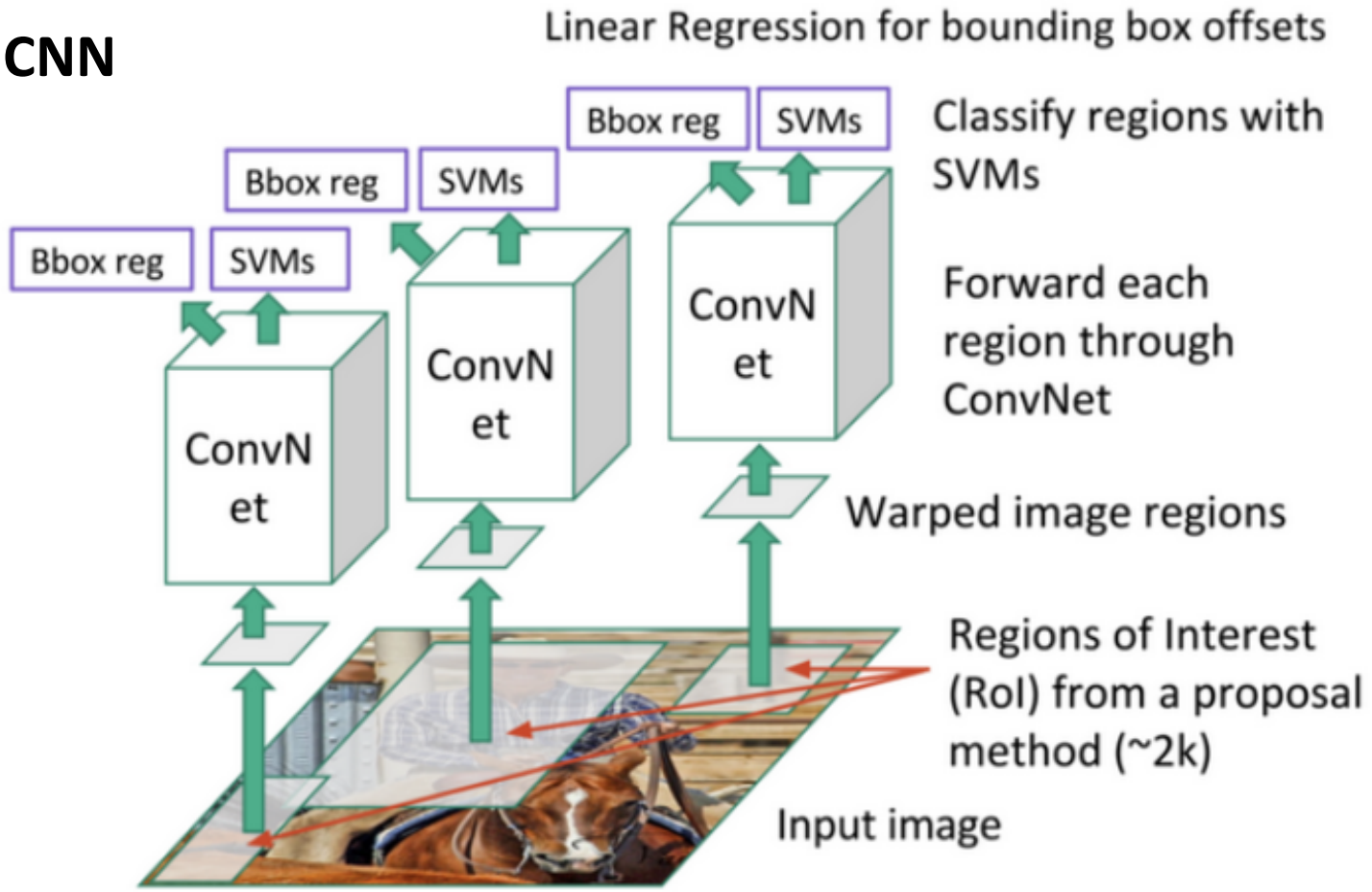
- Find blobs in the image that are most likely to contain objects
- E.g: Selective search → ~1000-2000 region proposals using CPU!



Source and Reference: http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf

Case Study: R-CNN

R-CNN: Region based CNN

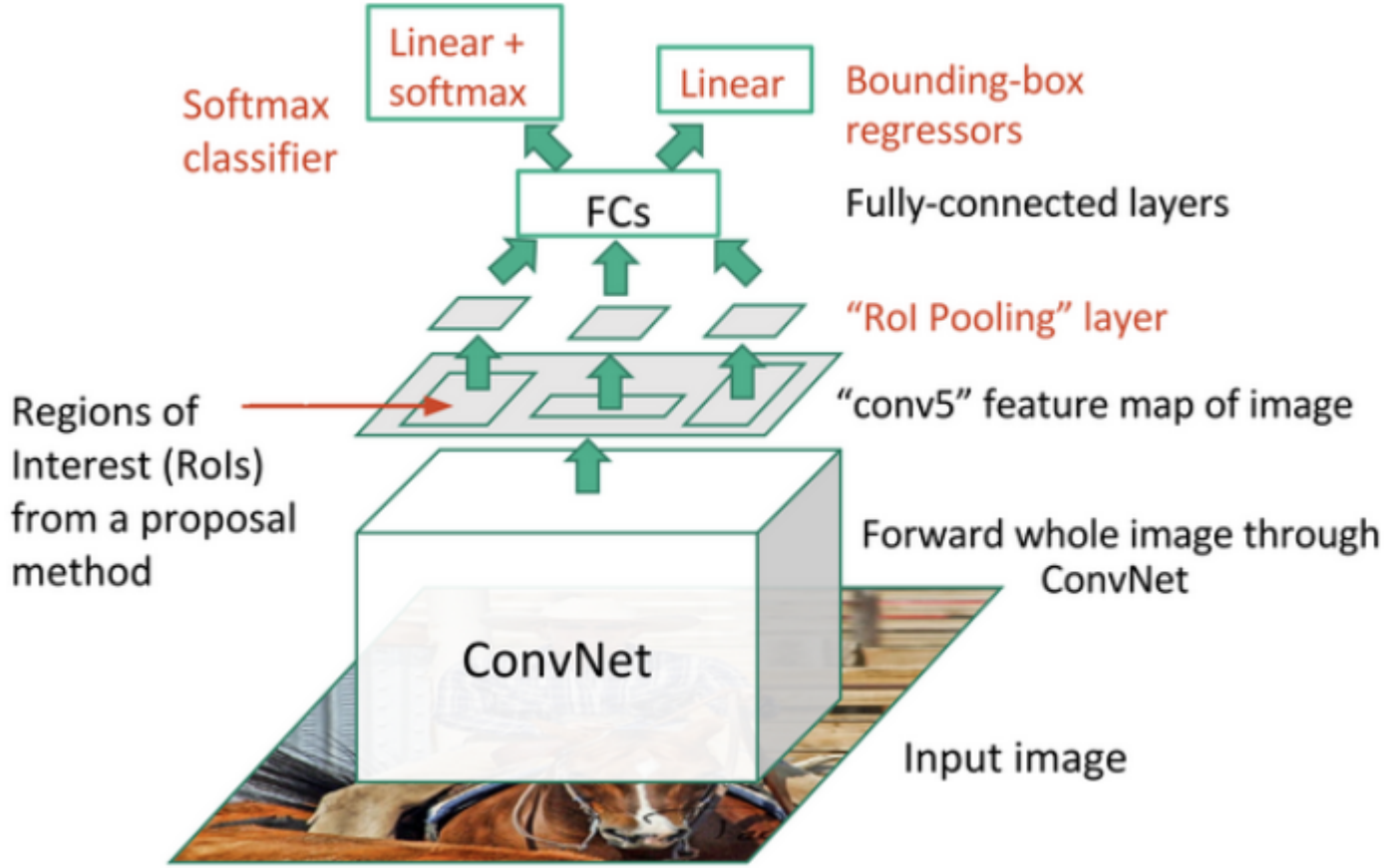


1. Resized to match the input to CNN requiremen.
2. mAP: 62.4% for 2007 PASCAL VOC

Source and Reference: http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf
https://medium.com/zylapp/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852

Case Study: FAST- R-CNN

1. Reduce computation
2. ROIs from feature maps using selective search
3. mAP: 70% for 2007 PASCAL VOC

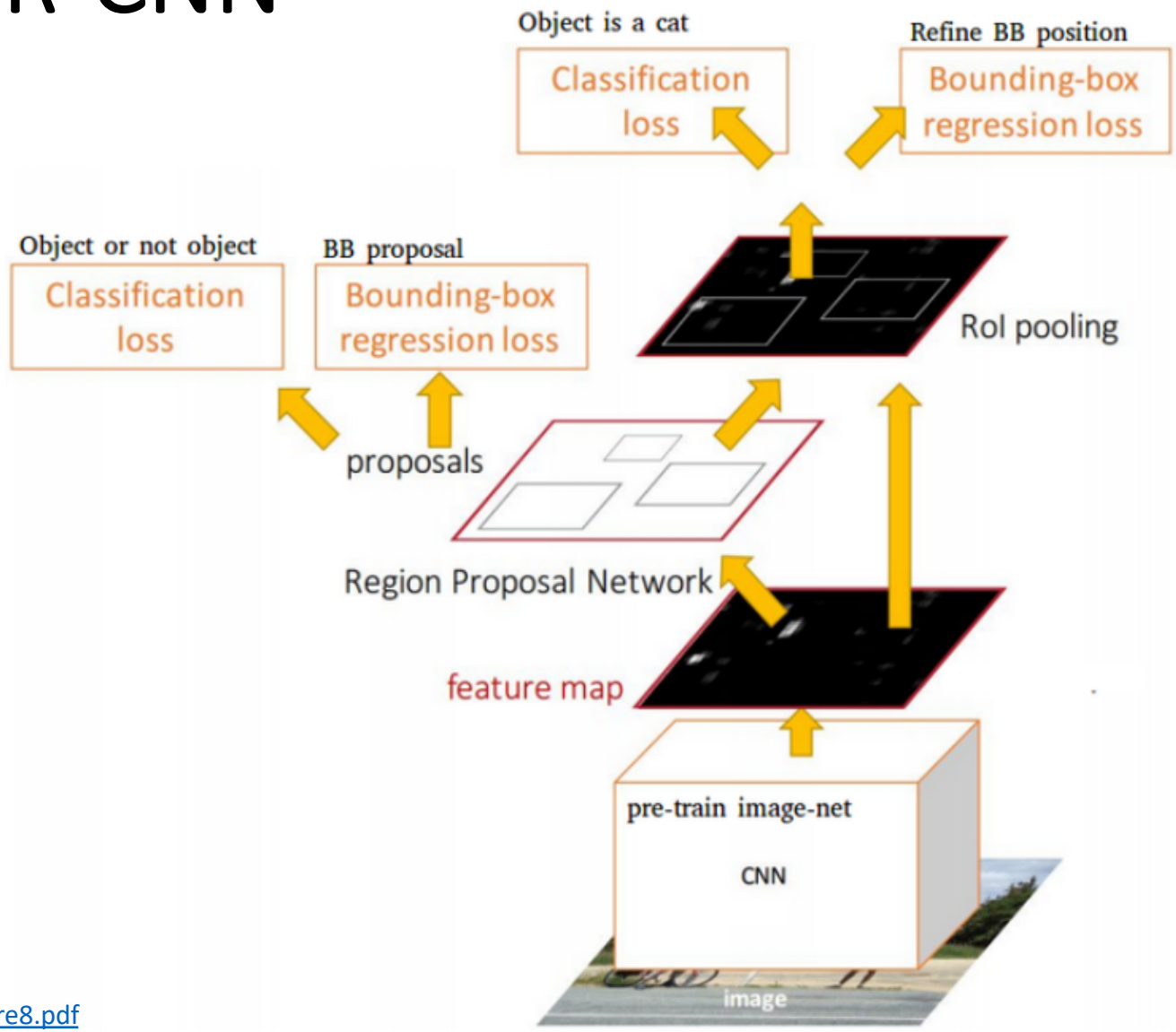


Source and Reference: http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf
https://medium.com/zylapp/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852

Case Study: FASTER- R-CNN

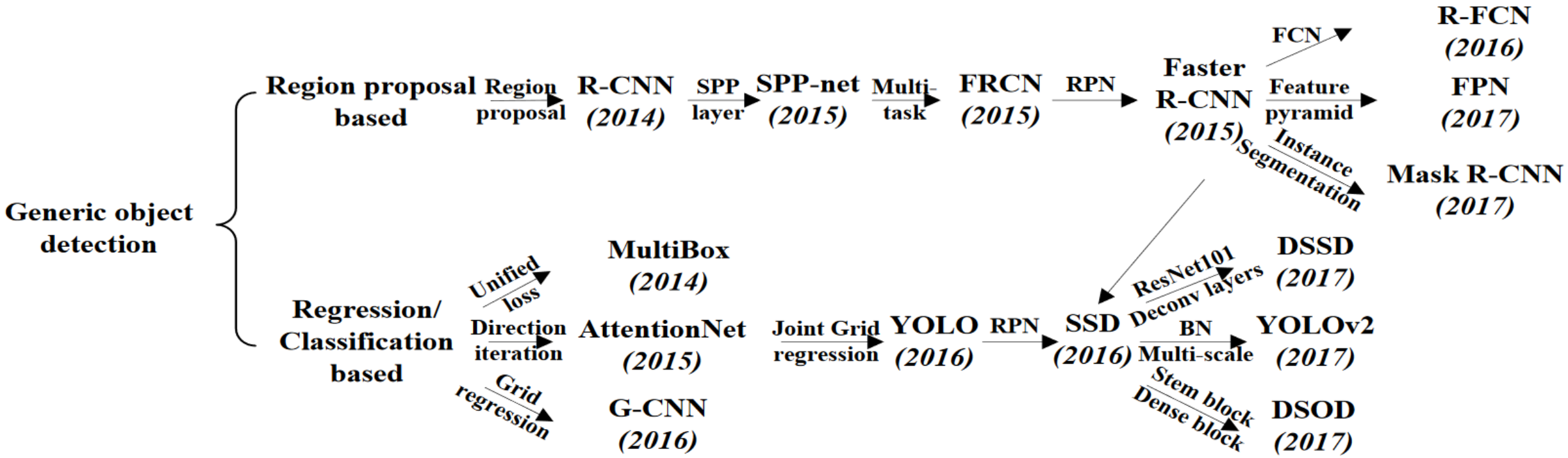
1. Introduced RPN (Region Proposal Network)

2. mAP: 78.8% for 2007 PASCAL VOC



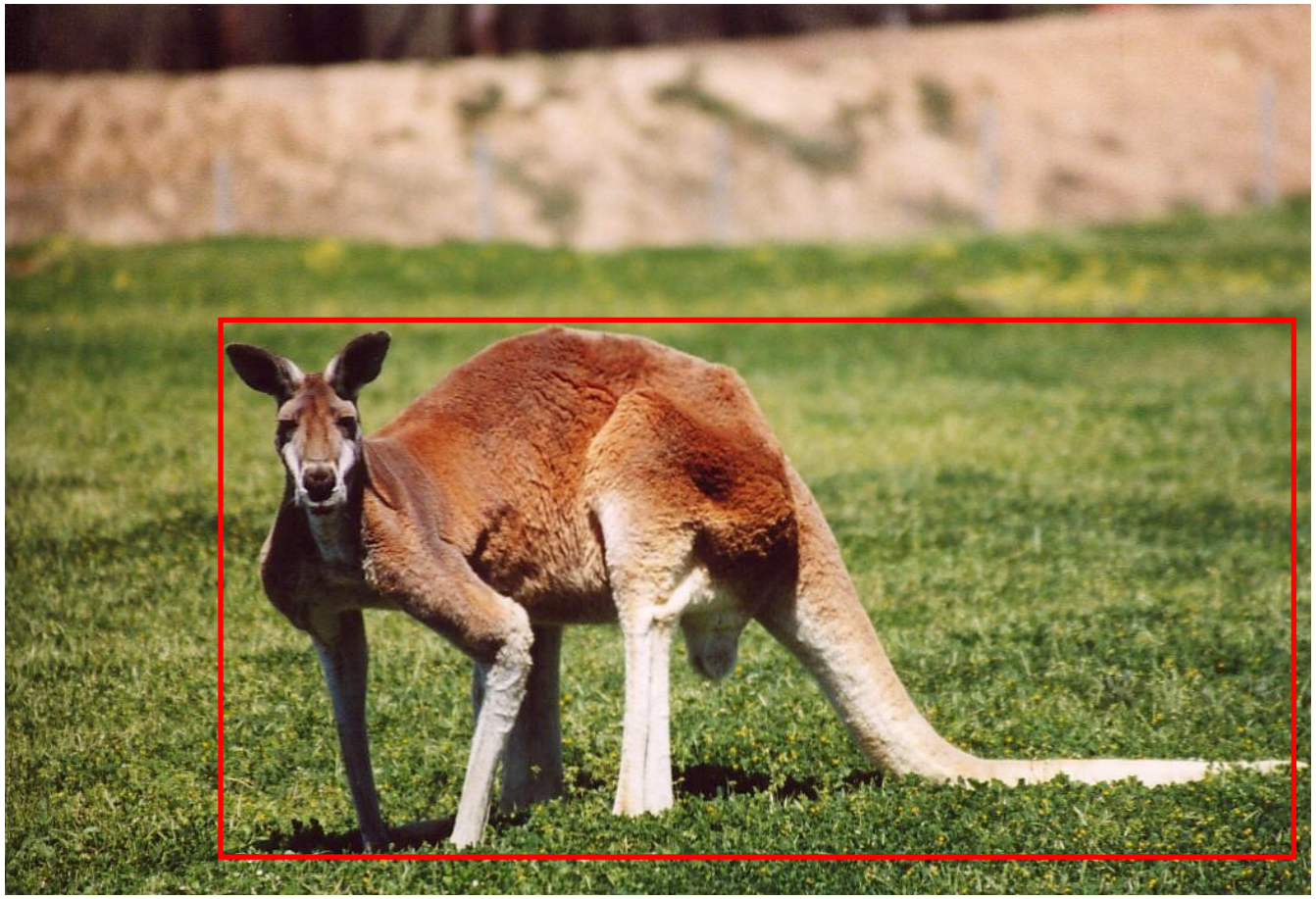
Source and Reference: http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf
https://medium.com/zylapp/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852

Object Detection Techniques History



Images source: https://arxiv.org/pdf/1807.05511.pdf

Image Annotation for Object Detection

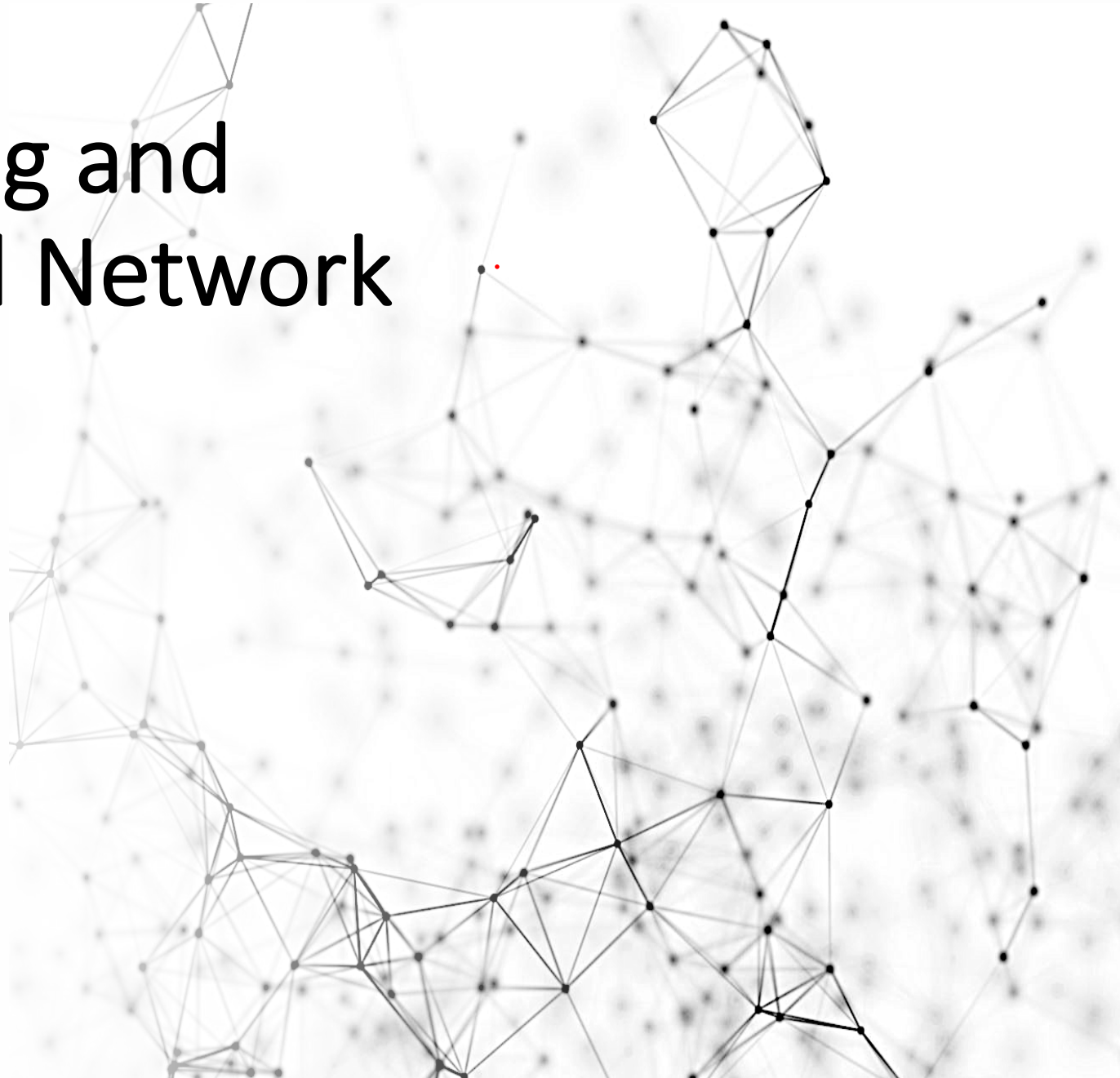


```
<annotation>
        <folder>Kangaroo</folder>
        <filename>00001.jpg</filename>
        <path>./Kangaroo/stock-12.jpg</path>
        <source>
                <database>Kangaroo</database>
        </source>
        <size>
                <width>450</width>
                <height>319</height>
                <depth>3</depth>
        </size>
        <segmented>0</segmented>
        <object>
                <name>kangaroo</name>
                <pose>Unspecified</pose>
                <truncated>0</truncated>
                <difficult>0</difficult>
                <bndbox>
                        <xmin>233</xmin>
                        <ymin>89</ymin>
                        <xmax>386</xmax>
                        <ymax>262</ymax>
                </bndbox>
        </object>
</annotation>
```

PASCAL VOC Format

Image Sources: https://en.wikipedia.org/wiki/Kangaroo#/media/File:RedRoo.JPG
https://towardsdatascience.com/coco-data-format-for-object-detection-a4c5eaf518c5

42028: Deep Learning and Convolutional Neural Network



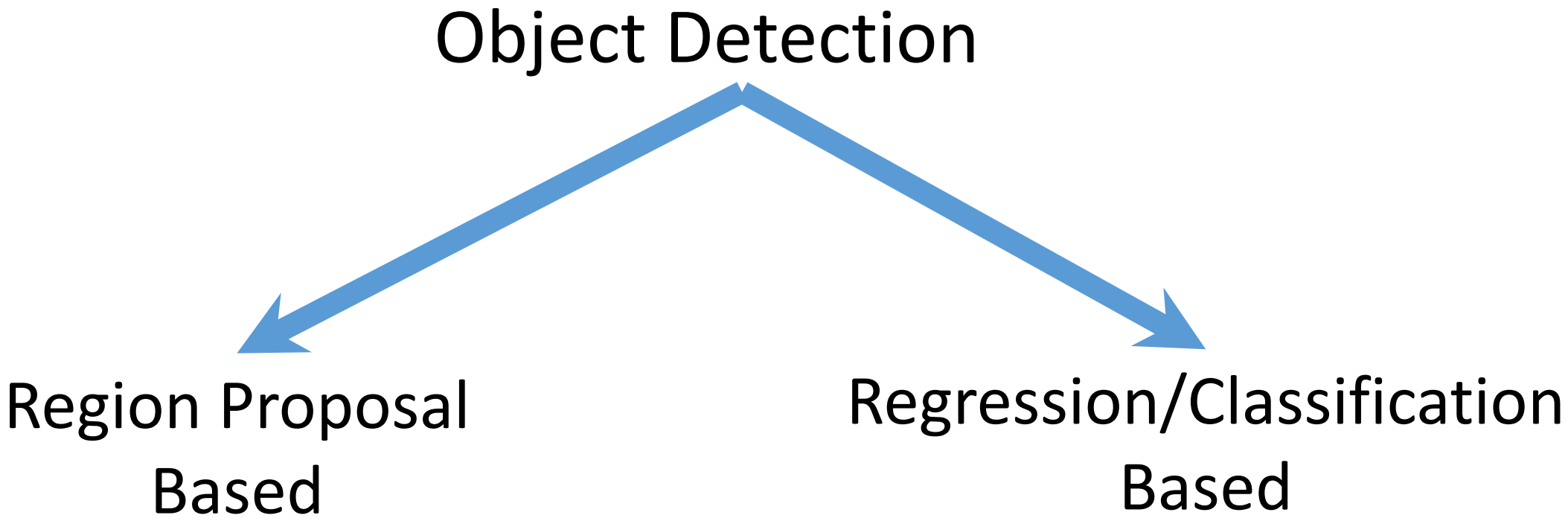
Week-9 Lecture

Object Detection -2

Outline

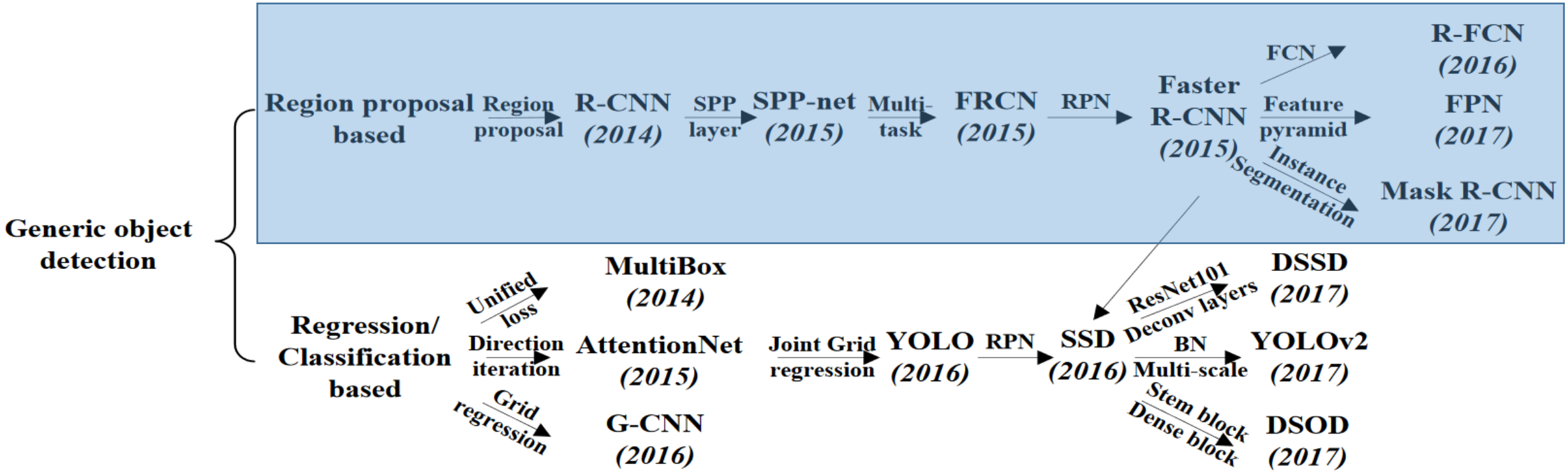
- Object detection techniques recap
- Strategies for predicting bounding boxes
- Non-Maxima suppression (NMS)
- Anchor boxes
- Case study:
  - Yolo (You Look Only Once)
  - SSD (Single Shot Detector)
- Object detection state-of-the-art

Frameworks



Images source: https://towardsdatascience.com/evolution-of-object-detection-and-localization-algorithms-e241021d8bad

Object Detection Techniques History



Images source: https://arxiv.org/pdf/1807.05511.pdf

Object Detection Techniques Recap

Sliding Window technique

Source and Reference: http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf
Image source: http://www.kamloopspropertyforsale.com/blog/does-owning-a-cat-depreciate-your-home.html

Object Detection Techniques Recap

Sliding Window technique

- Crop images and classify using CNN
- Try different sizes of the sliding window

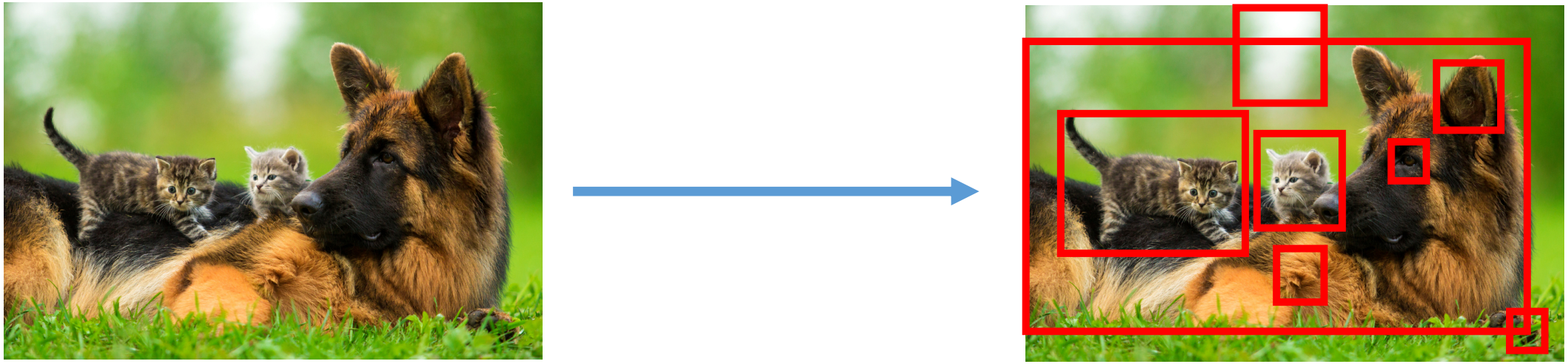
Issues:

- Slow
- Computationally very expensive
- Less accurate

Images source: https://arxiv.org/pdf/1807.05511.pdf

Object Detection Techniques Recap

Region Proposals



Source and Reference: http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf
Image source: http://www.kamloopspropertyforsale.com/blog/does-owning-a-cat-depreciate-your-home.html

Predicting Bounding Boxes

Currently:

- Sliding Window
- Selective Search
- Region Proposals

Task:

- Predict Bounding boxes from CNN

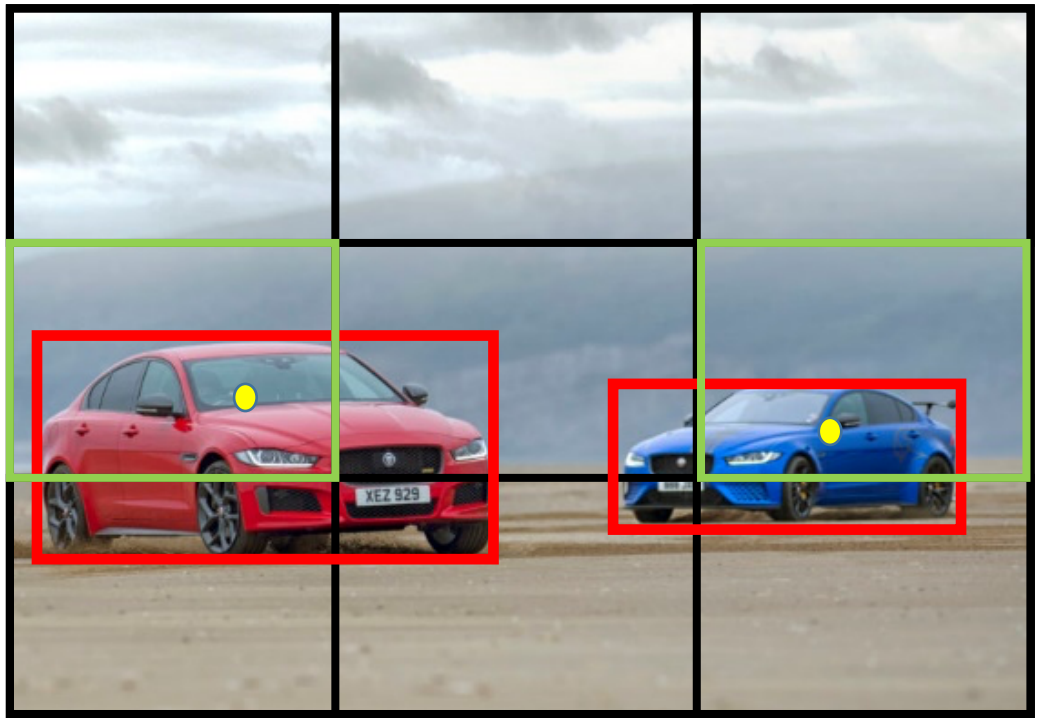
Source and Reference: http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf

Predicting Bounding Boxes

- Place a grid over the image
- Apply image classification and localization to each of the grid cells

Images source: https://yallacompare.com/car-deals/uae/en/two-cars-one-dnajaguar-xe-300-sport-and-xe-sv-project-8/
Source and Reference: https://www.youtube.com/watch?v=gKreZOUi-O0

Predicting Bounding Boxes



Class : {car, bike}

Training Strategy:

Input:

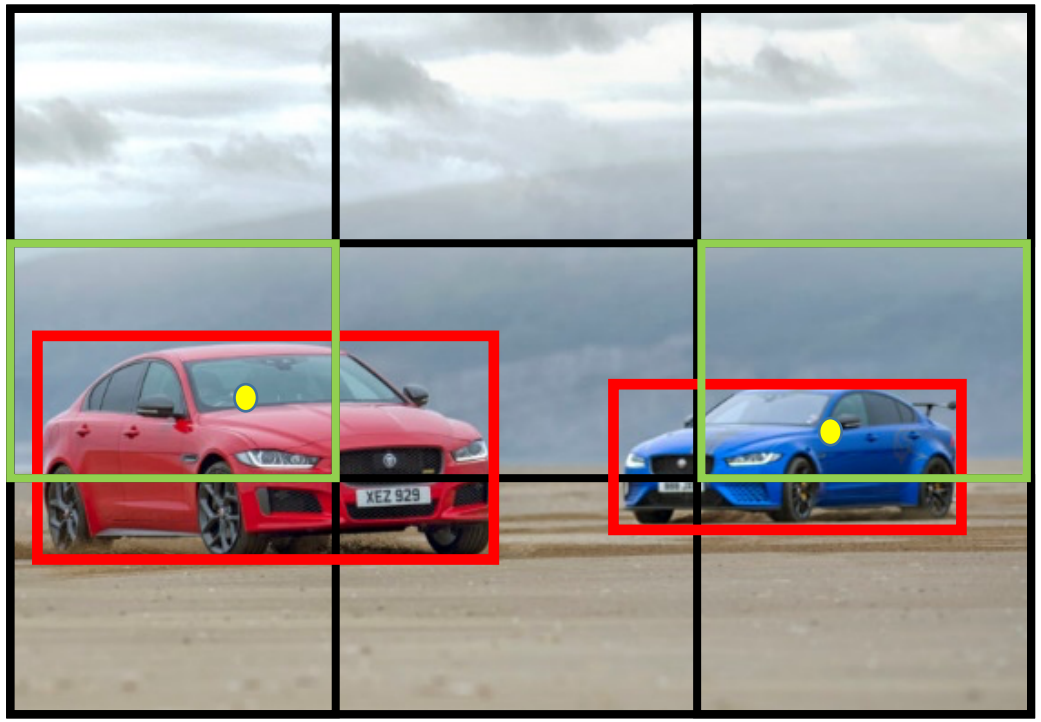
- Image: (ht x wd x 3)

Target:

- Bounding box information for each object
- Class for each object

Images source: https://yallacompare.com/car-deals/uae/en/two-cars-one-dnajaguar-xe-300-sport-and-xe-sv-project-8/
Source and Reference: https://www.youtube.com/watch?v=gKreZOUi-O0

Predicting Bounding Boxes



Class : {car, bike}

Idea: Take the mid-point of the object and Assign it to a grid cell based on its location

Training Strategy:

Target:

$Y = \{p_o, x, y, h, w, c_1, c_2\}$ for each cell

e.g:

Cell(1,1) = {0, ?, ?, ?, ?, ?, ?}

:

Cell(2,1) = {1, x, y, h, t, 1, 0}

Cell(2,2) = {0, ?, ?, ?, ?, ?, ?}

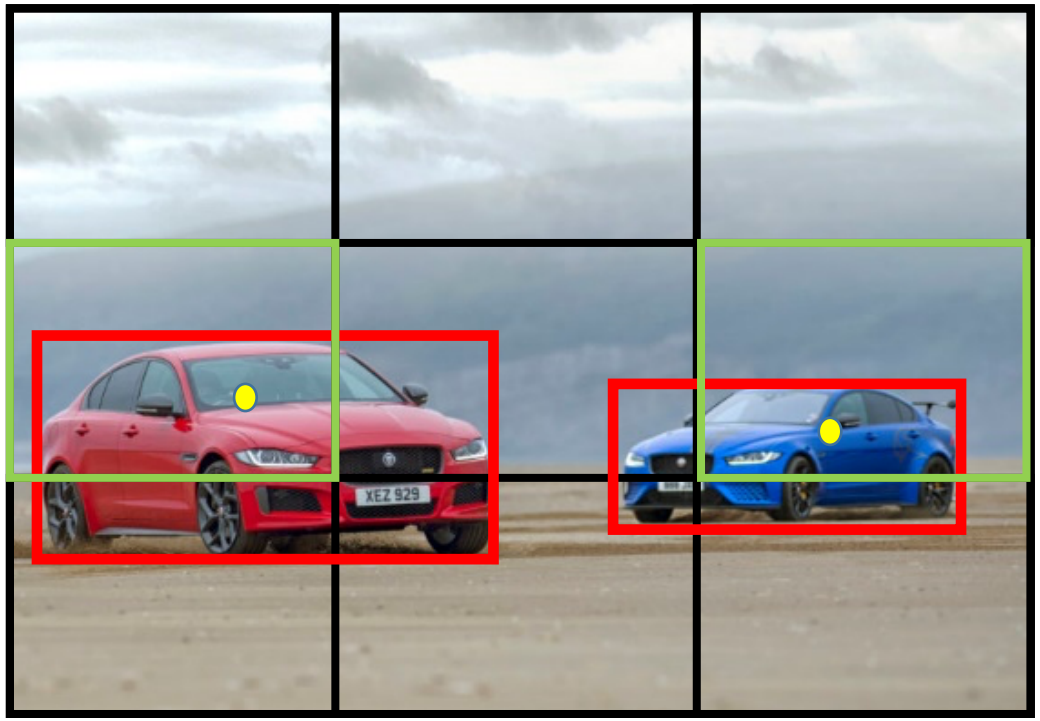
Cell(2,3) = {1, x, y, h, t, 1, 0}

:

Cell(3,3) = {0, ?, ?, ?, ?, ?, ?}

Images source: https://yallacompare.com/car-deals/uae/en/two-cars-one-dnajaguar-xe-300-sport-and-xe-sv-project-8/
Source and Reference: https://www.youtube.com/watch?v=gKreZOUi-O0

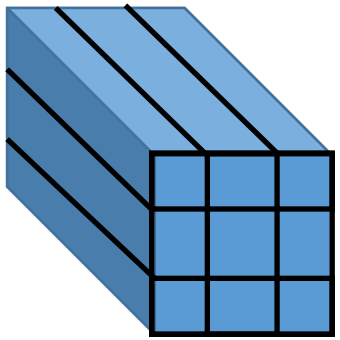
Predicting Bounding Boxes



Class : {car, bike}

Training Strategy:

Target output vector:
3 X 3 X 7
3 X 3: Grid size
7: (5 + Number-of-Classes)



3 X 3 X 7

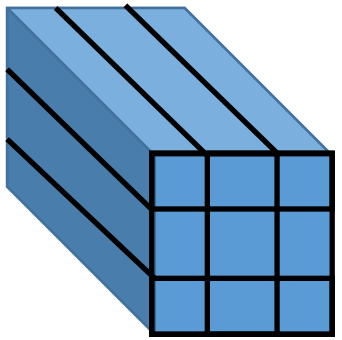
Images source: https://yallacompare.com/car-deals/uae/en/two-cars-one-dnajaguar-xe-300-sport-and-xe-sv-project-8/
Source and Reference: https://www.youtube.com/watch?v=gKreZOUi-O0

Predicting Bounding Boxes

Training Strategy:

Input: X

Target: Y

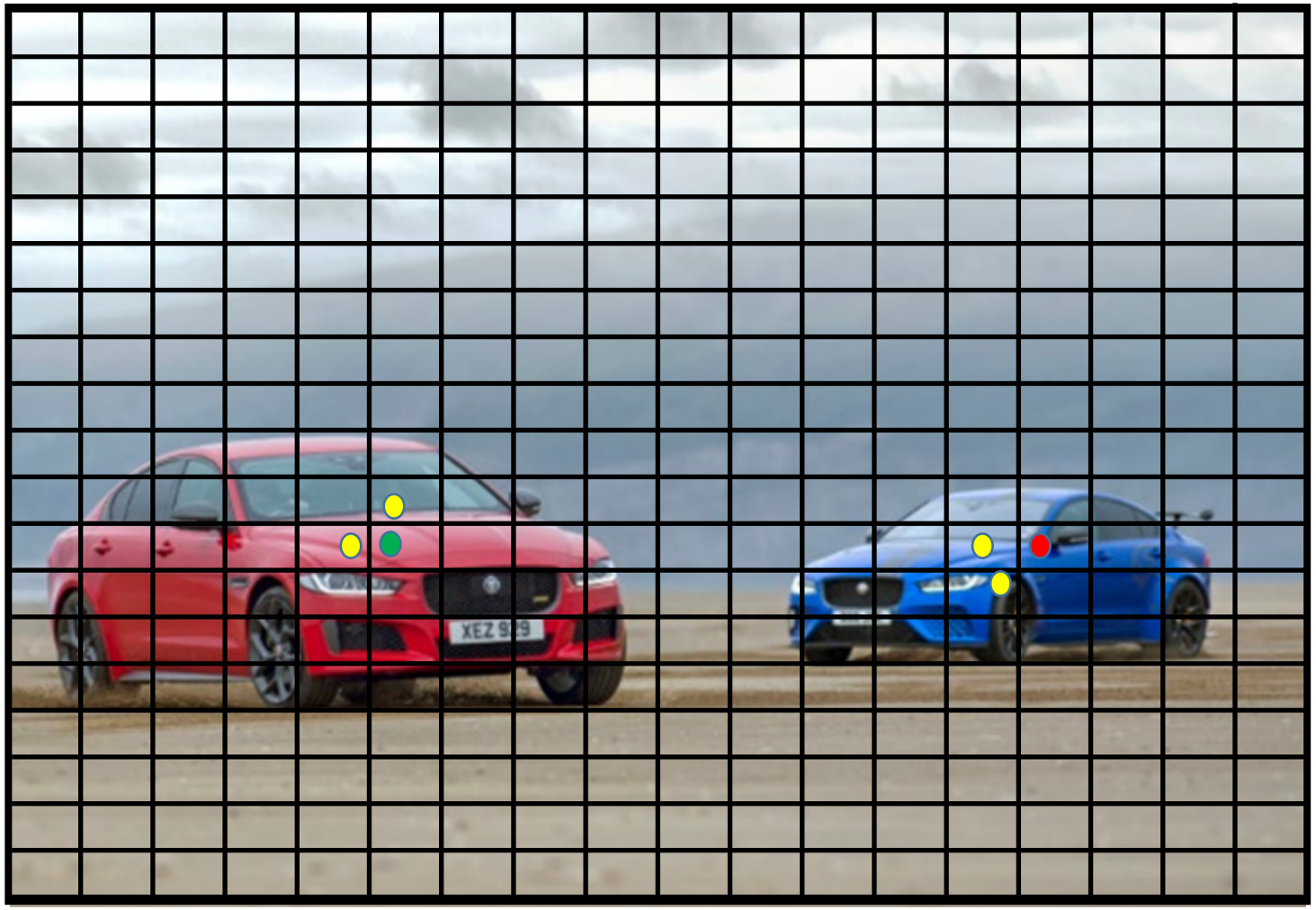


Class : {car, bike}

In practice: The grid is finer, 19 X 19 instead of 3 X 3
So, Target will be of size: 19 X 19 X 7
Works well for non-overlapping objects

Images source: https://yallacompare.com/car-deals/uae/en/two-cars-one-dnajaguar-xe-300-sport-and-xe-sv-project-8/
Source and Reference: https://www.youtube.com/watch?v=gKreZOUi-O0

Non Maxima Suppression (NMS)

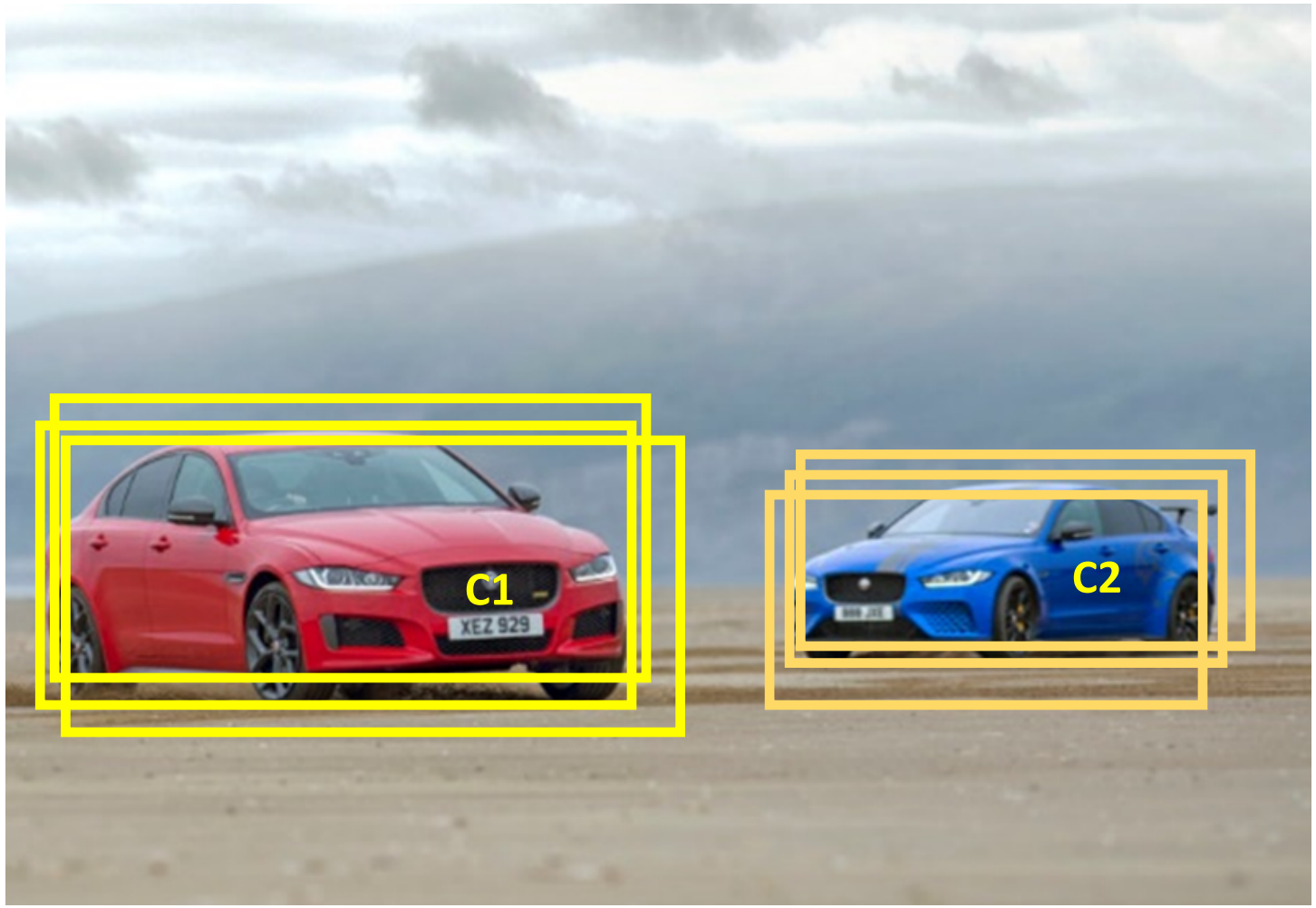


Issues with Object Detection:

1. **Each object has one mid-point**
2. **Each cells are subjected to object localization + classification**
3. **Hence, neighbouring cells might assume that it has the mid-point**
4. **Hence, Multiple detection bounding box**

Images source: https://arxiv.org/pdf/1807.05511.pdf

Non Maxima Suppression (NMS)



Sample prediction:

<u>For C1:</u>

Box1: 0.9 (Confidence Score)

Box2: 0.79

Box3: 0.82

<u>For C2:</u>

Box1: 0.92

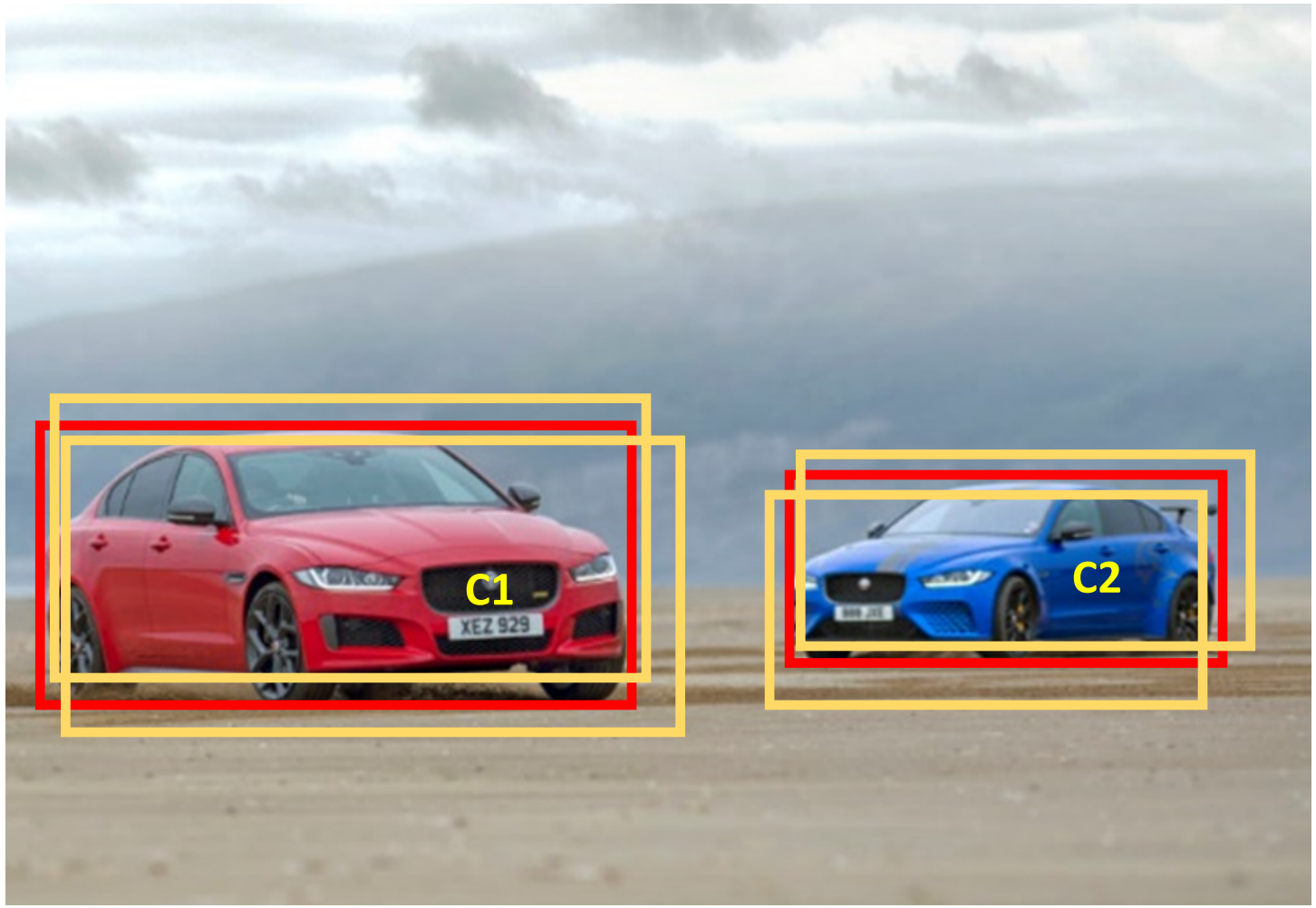
Box2: 0.85

Box3: 0.7

NMS cleans/removes the multiple detection and only keeps the one with very high confidence

Images source: https://arxiv.org/pdf/1807.05511.pdf

Non Maxima Suppression (NMS)

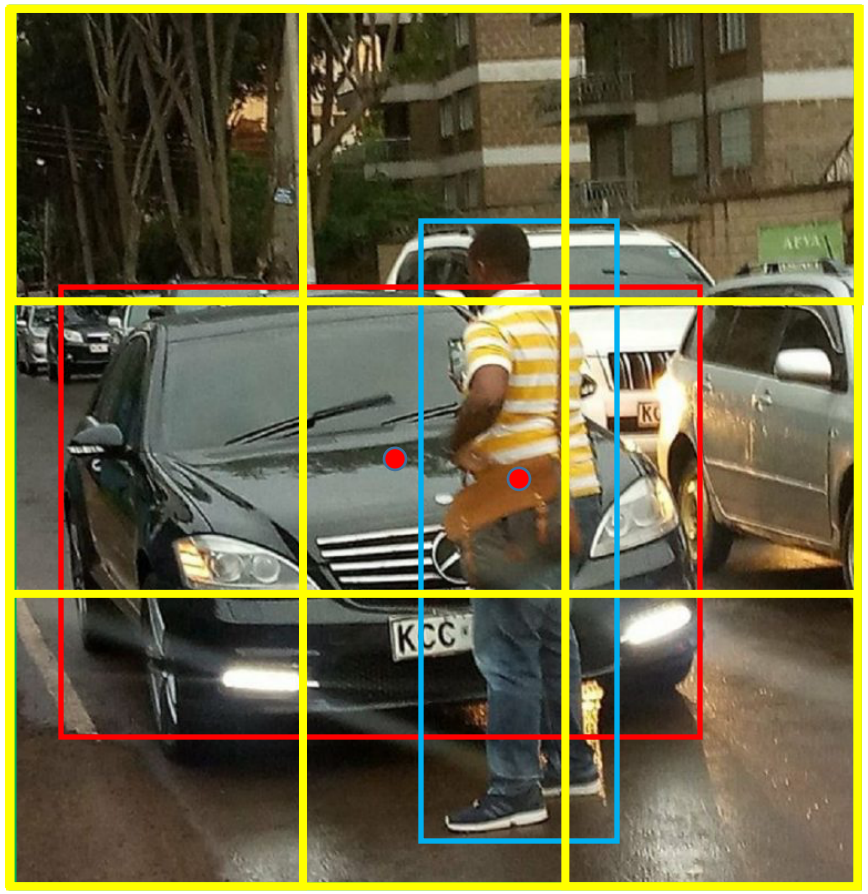


1. **Check the probabilities of each detection and keep ones with *score > Threshold (0.7)***

2. **For remaining boxes:**
   - Box with highest score is the detection results.
   - Discard any remaining boxes with IoU > 0.5 with final detected box,
   i.e: overlap with the box with highest score.

Images source: https://arxiv.org/pdf/1807.05511.pdf

YOLO: You Only Look Once Algorithm

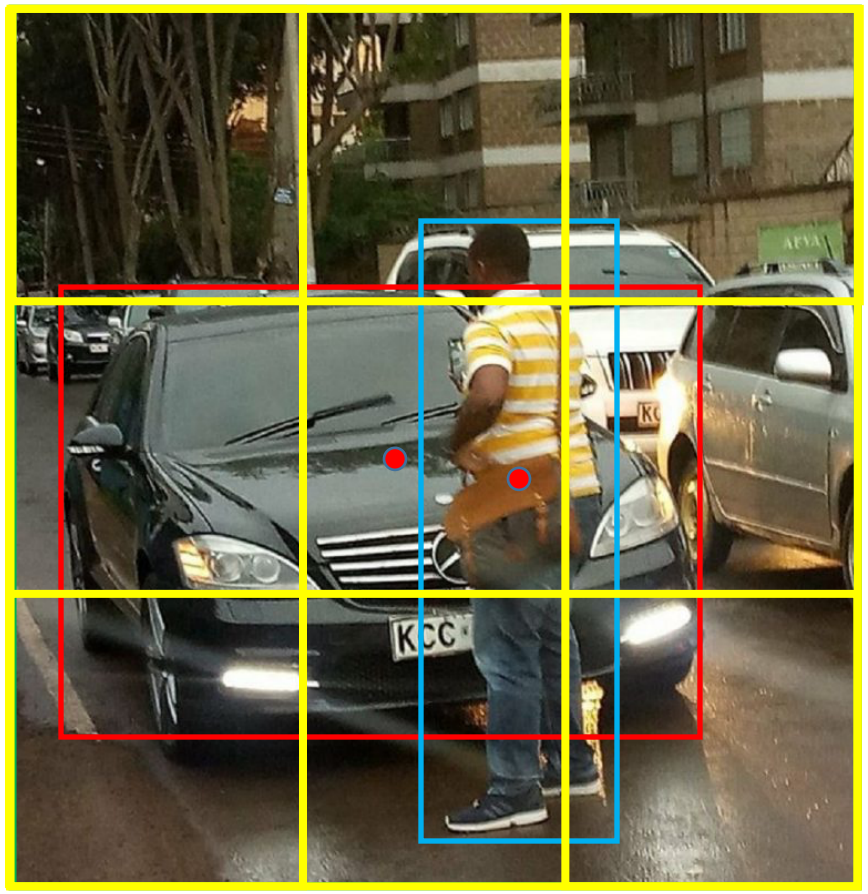


Challenges with overlapping objects

- Each grid cell detect only one object
- For multiple overlapping objects, Mid point are on the same grid cell

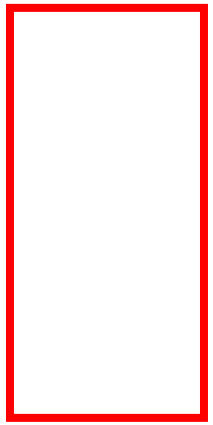
Images source: https://businesstoday.co.ke/brave-kenyan-blocks-mps-overlapping-car/
Source and Reference: https://www.youtube.com/watch?v=gKreZOUi-O0

Anchor Boxes



So, Currently the Target Y = {1, x, y, h, w, C1, C2},
As the mid-points for both the objects are on the same grid cell, only one of the objects will be associated

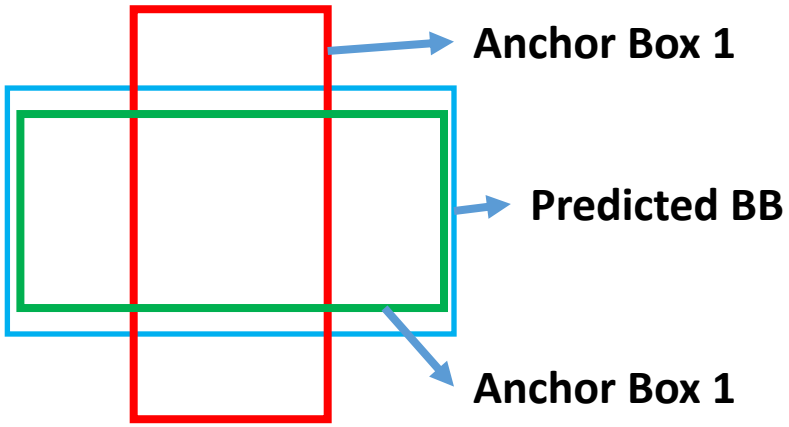
Anchor Box 1



Anchor Box 2

Images source: https://businesstoday.co.ke/brave-kenyan-blocks-mps-overlapping-car/
Source and Reference: https://www.youtube.com/watch?v=gKreZOUi-O0

Anchor Boxes



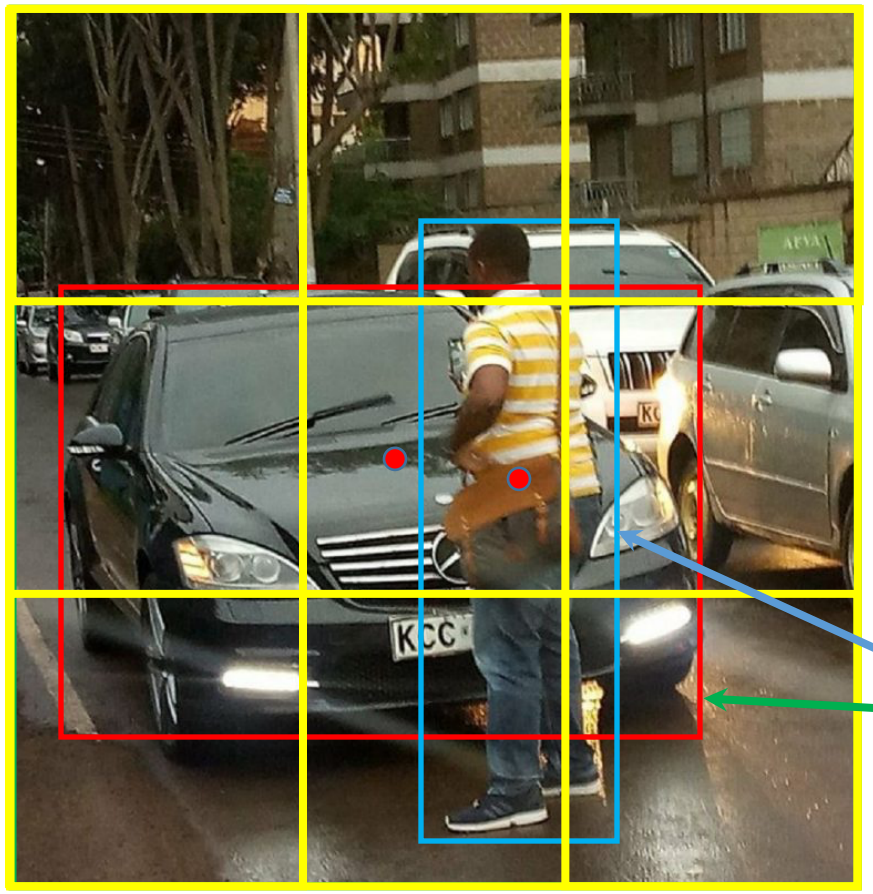
Calculate the IoU of
Anchor boxes and predicted BB

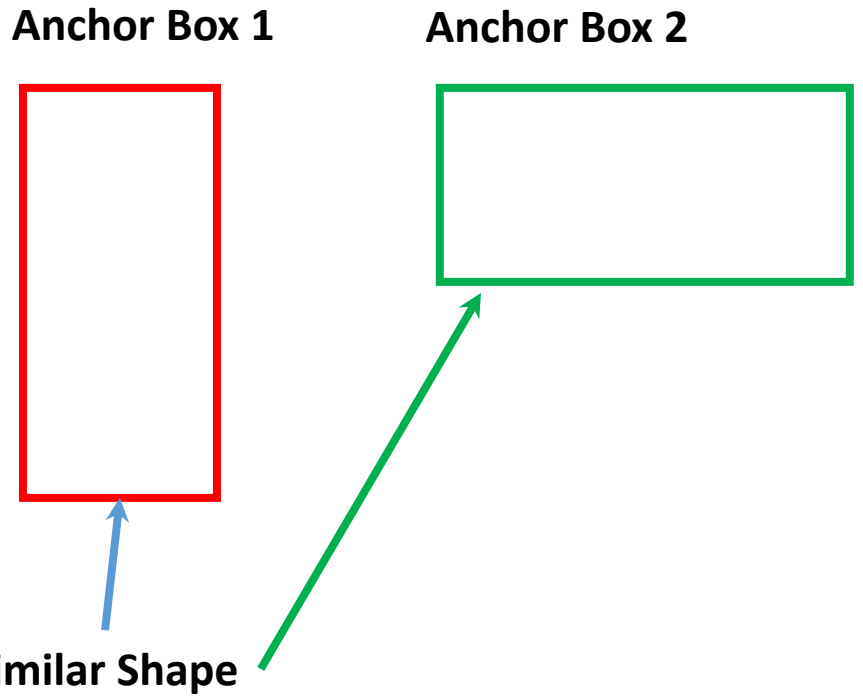
Associate each object to:

1. **A cell which contains its mid-point and**
2. **Anchor box for the cell with highest IoU**

Source and Reference: https://www.youtube.com/watch?v=gKreZOUi-O0

Anchor Boxes





So, with Anchor boxes:

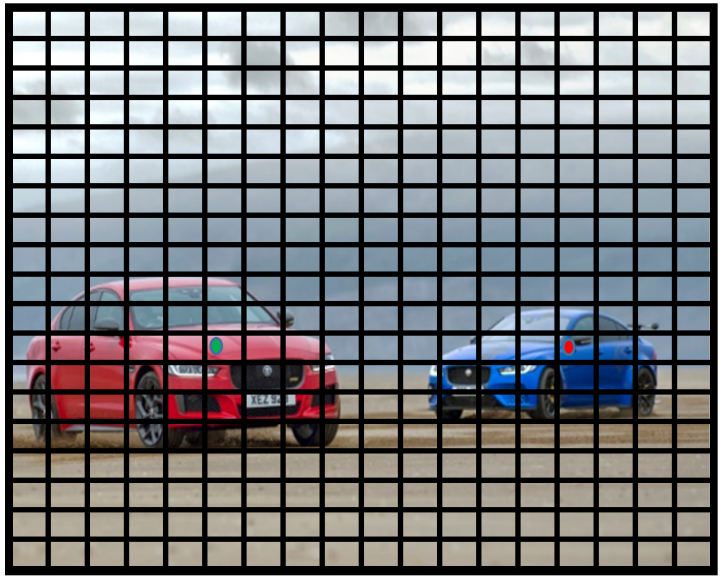
Target Y = {P_o, x, y, h, w, C1, C2, P_o, x, y, h, w, C1, C2},

Anchor Box 1 Anchor Box 2

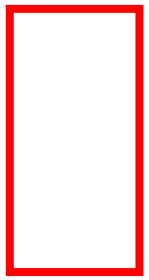
Images source: https://arxiv.org/pdf/1807.05511.pdf

YOLO: You Only Look Once Algorithm

Input X



Anchor Box 1



Anchor Box 2

Class : {1:car, 2:bike}

Y size : (19 X 19 X 2 X 7)

Grid Size

#Anchor Box

5(P_o, x,y,h,w) + #Classes(2)

Training Set

Y = {P_o, x, y, h, w, C1, C2, P_o, x, y, h, w, C1, C2}

Cell(1,1) = {0, ?, ?, ?, ?, ?, ?, 0, ?, ?, ?, ?, ?, ?}

:

Cell(12,6)= {0, ?, ?, ?, ?, ?, ?, 1, x, y, h, w, 1, 0}

:

Cell(12,15)= {0, ?, ?, ?, ?, ?, ?, 1, x, y, h, w, 1, 0}

:

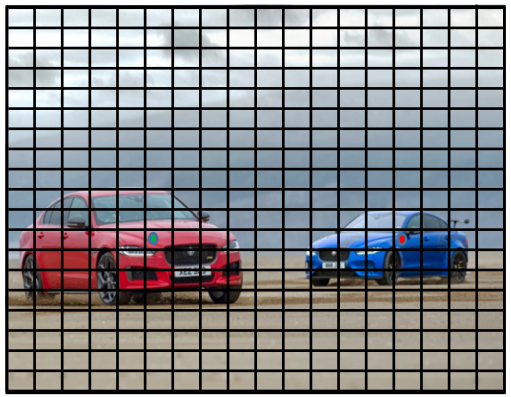
Cell(19,19)= {0, ?, ?, ?, ?, ?, ?, 0, ?, ?, ?, ?, ?, ?}

Images source: https://arxiv.org/pdf/1807.05511.pdf

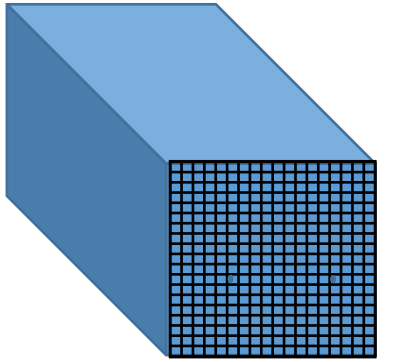
YOLO: You Only Look Once Algorithm

Training:

Input: X



Target: Y



19 X 19 X 2 X 7

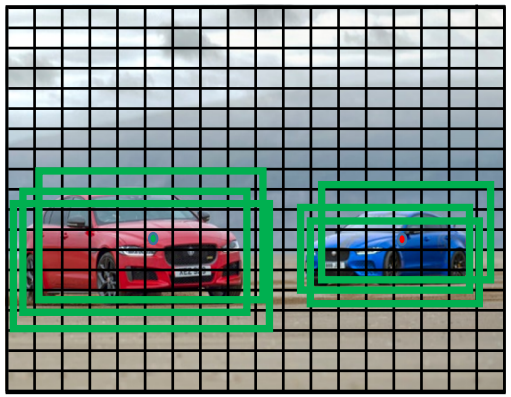
Class : {car, bike}

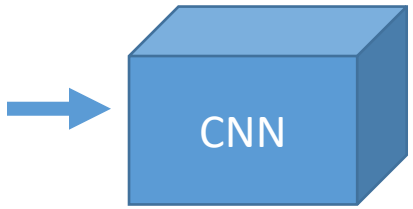
Images source: https://businesstoday.co.ke/brave-kenyan-blocks-mps-overlapping-car/
Source and Reference: https://www.youtube.com/watch?v=gKreZOUi-O0

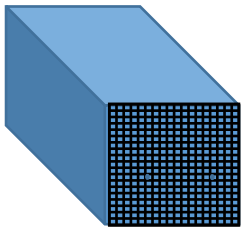
YOLO: You Only Look Once Algorithm

Testing:

Input: X







19 X 19 X 2 X 7

Class : {car, bike}

$Y = \{P_o, x, y, h, w, C1, C2, P_o, x, y, h, w, C1, C2\}$

{0, ?, ?, ?, ?, ?, ?, 0, ?, ?, ?, ?, ?, ?}

:

{0, ?, ?, ?, ?, ?, ?, 1, x, y, h, w, 1, 0}

:

{0, ?, ?, ?, ?, ?, ?, 1, x, y, h, w, 1, 0}

:

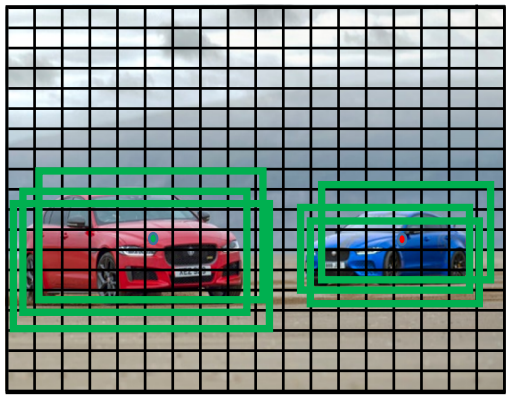
{0, ?, ?, ?, ?, ?, ?, 0, ?, ?, ?, ?, ?, ?}

Images source: https://businesstoday.co.ke/brave-kenyan-blocks-mps-overlapping-car/
Source and Reference: https://www.youtube.com/watch?v=gKreZOUi-O0

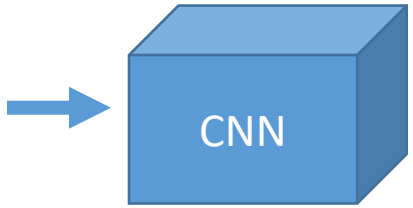
YOLO: You Only Look Once Algorithm

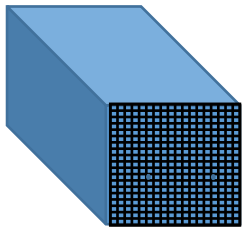
Testing:

Input: X



Class : {car, bike}





19 X 19 X 2 X 7

Y = {P_o, x, y, h, w, C1, C2, P_o, x, y, h, w, C1, C2}

{0, ?, ?, ?, ?, ?, ?, 0, ?, ?, ?, ?, ?, ?}

:

{0, ?, ?, ?, ?, ?, ?, 1, x, y, h, w, 1, 0}

:

{0, ?, ?, ?, ?, ?, ?, 1, x, y, h, w, 1, 0}

:

{0, ?, ?, ?, ?, ?, ?, 0, ?, ?, ?, ?, ?, ?}

Apply NMS

Images source: https://businesstoday.co.ke/brave-kenyan-blocks-mps-overlapping-car/
Source and Reference: https://www.youtube.com/watch?v=gKreZOUi-O0

YOLO: You Only Look Once Algorithm

- Real-time performance with 45 frames per sec, 0.02 sec per image
- Not suitable for small objects
- Issues with new or multiple aspect ratios and unable to generalize

Source: https://arxiv.org/pdf/1807.05511.pdf

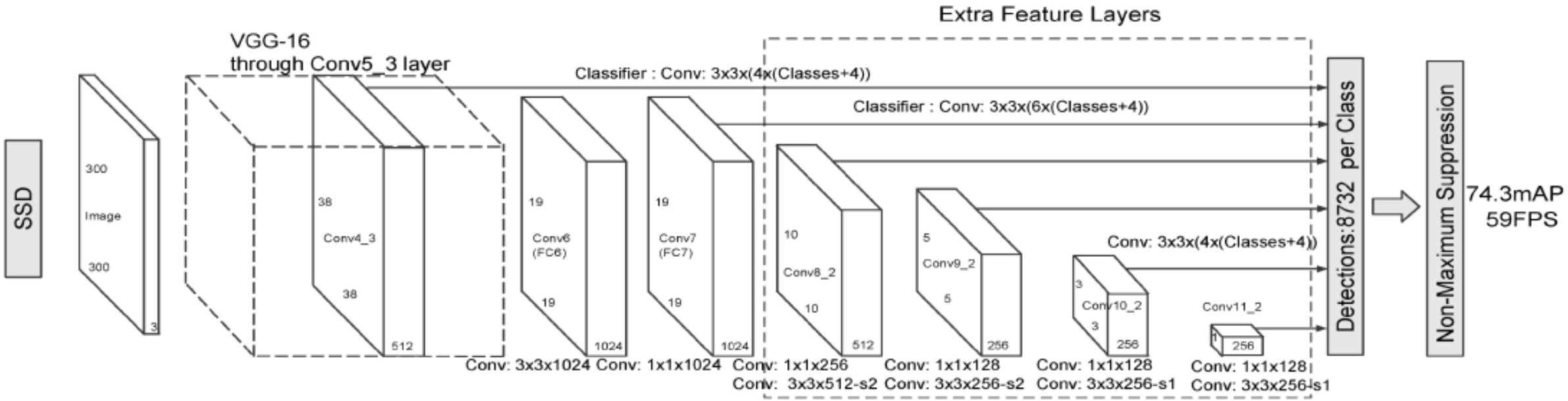
Single Shot Detector(SSD)

- Similar to YOLO
- VGG16 base CONV layers
- Take advantage of Anchor boxes with different aspect ratios
- Large number of anchors boxes are chosen
- Not suitable for small objects
- 3 times faster than Faster-RCNN
- With ResNet101 base SSD may be help in detecting small objects with better features from the CONV base

Source: https://arxiv.org/pdf/1807.05511.pdf

Single Shot Detector(SSD)

SSD300 architecture:



Source: https://arxiv.org/pdf/1807.05511.pdf

Object Detection State-of-the-Art

Dataset: PASCAL VOC 2007 and 2017
Test Dataset : PASCAL VOC 2007

Method	Train Dataset	mAP	Time in sec/image	Time Frame /sec
RCNN (VGG16)	Pascal VOC 2007	66.0	50	-
Fast RCNN	VOC 2007+2012	70.0	2	-
Faster RCNN (VGG16)	VOC 2007+2012	73.2	0.11	9
Faster RCNN (ResNet101)	VOC 2007+2012	**83.8**	2.24	0.4
Yolo	VOC 2007+2012	63.4	0.02	45
SSD300	VOC 2007+2012	74.3	0.02	45
SSD512	VOC 2007+2012	**76.8**	0.05	19

Source: https://arxiv.org/pdf/1807.05511.pdf

Object Detection Summary

Base Networks:

- VGG16
- REsNet101
- Inception V2
- Inception V3
- ResNet
- MobileNet
- Alexnet
- ZFNet

Etc.

Object Detection FrameWorks:

- RCNN Family (RCNN, Fast/Faster RCNN)
- Yolo
- SSD
- F-RCN

Summary:

- Faster-RCNN is more accurate but slower
- Yolo/SSD are faster/real-time but not much accurate

Source: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf

42028: Deep Learning and Convolutional Neural Network

Week-10 Lecture

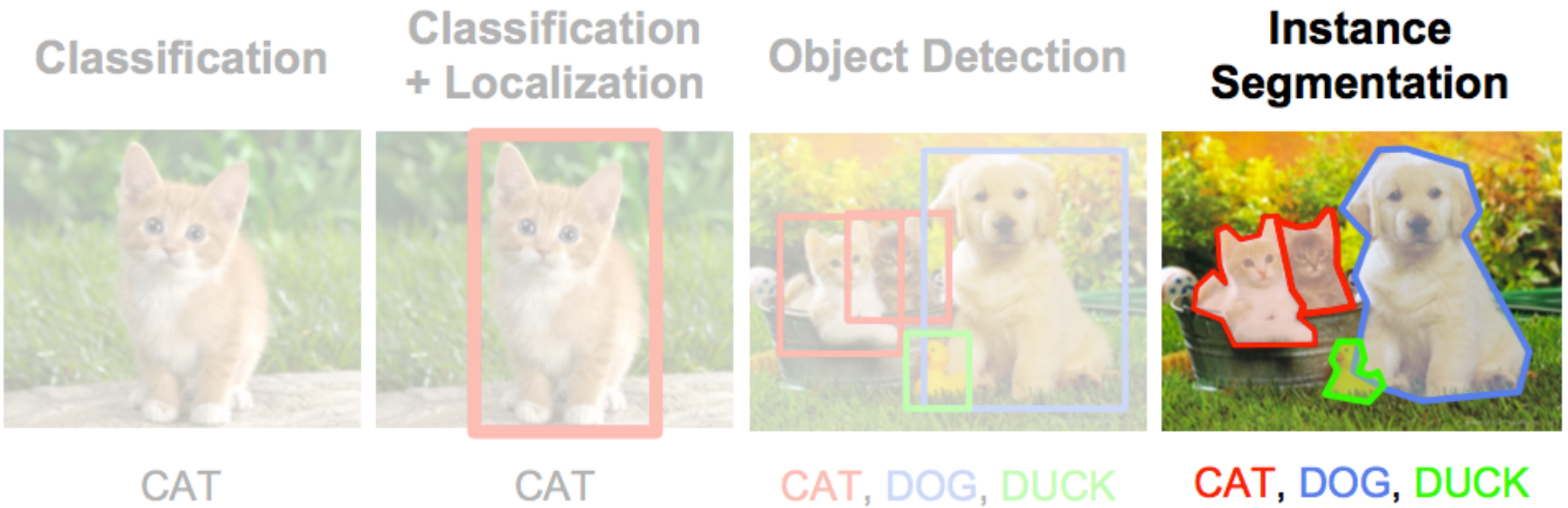
Instance Segmentation, Inception, GoogleNet and ResNet

Outline

- Introduction to Instance Segmentation
- Techniques and application of Instance Segmentation
- Case Study: Mask R-CNN
- 1X1 Convolution and it’s use
- Motivation behind Inception
- Inception V1, V2 and V3 Modules
- Introduction to ResNet
- Motivation behind ResNet
- Residual Block

Introduction to Instance Segmentation

Computer Vision Problems:



Images source: cs224d course

Introduction to Instance Segmentation

Semantic Segmentation Vs Instance Segmentation:

- Semantic segmentation classifies object pixels to specific classes/category
- Instance Segmentation identifies each pixels object instance

Input Image

Semantic Segmentation

Instance Segmentation

Images source: https://datascience.stackexchange.com/questions/52015/what-is-the-difference-between-semantic-segmentation-object-detection-and-insta

Introduction to Instance Segmentation

Popular Techniques:

Semantic Segmentation	Instance Segmentation
Conditional Random Field (CRF) Fully Convolutional Network (FCN) U-Net Pyramid Scene Parsing Network (PSPNet) etc.	SegNet, DeepMask, SharpMask, **MaskRCNN**, etc.

Images source: https://datascience.stackexchange.com/questions/52015/what-is-the-difference-between-semantic-segmentation-object-detection-and-insta

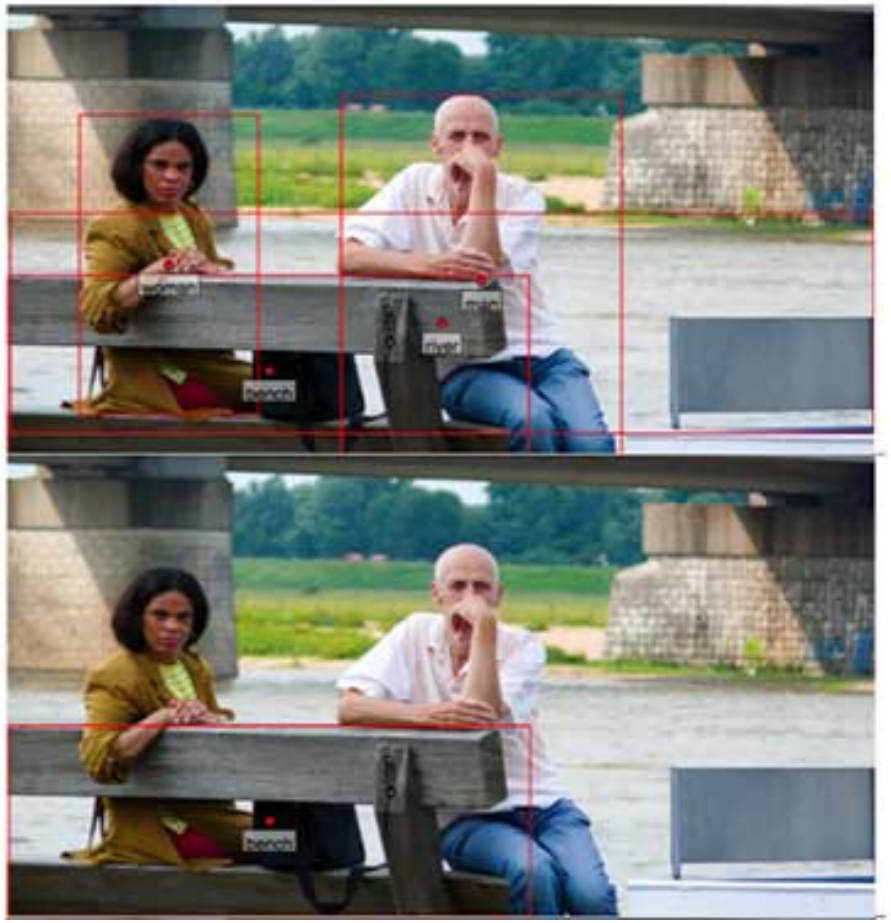
Introduction to Instance Segmentation

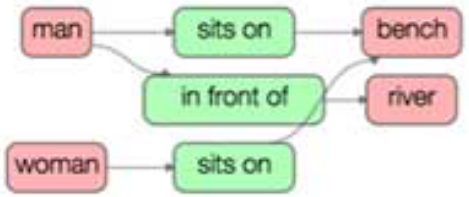
Applications: Autonomous Driving

Images source: https://slideplayer.com/slide/784090/3/images/2/Dense+CRF+construction.jpg

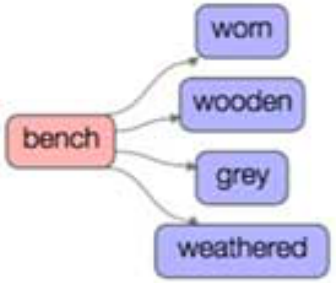
Introduction to Instance Segmentation

Applications: Scene Understanding





A man and a woman sit on a park bench along a river.

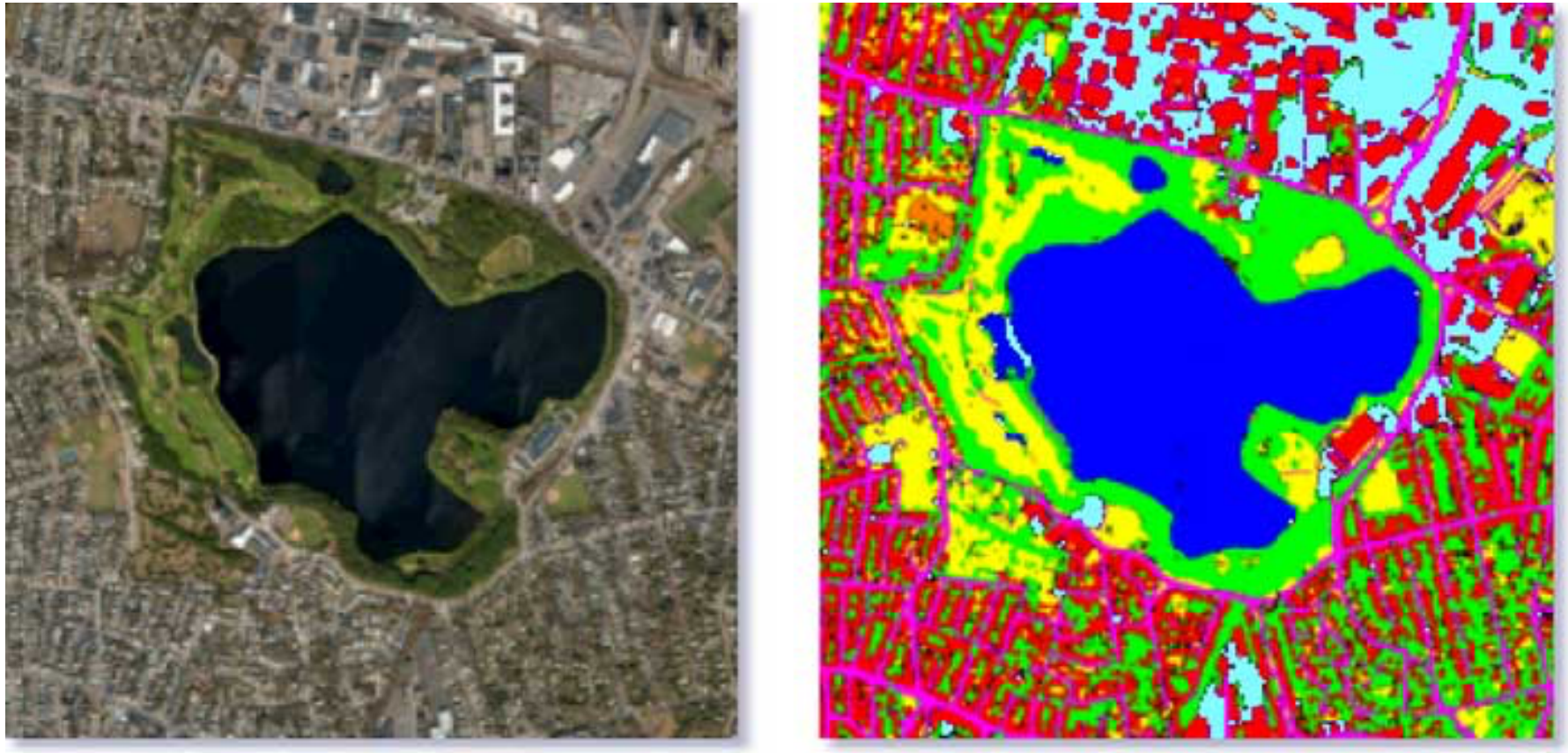


Park bench is made of gray weathered wood

Images source: https://www.topbots.com/semantic-segmentation-guide/

Introduction to Instance Segmentation

Applications: Aerial Image processing

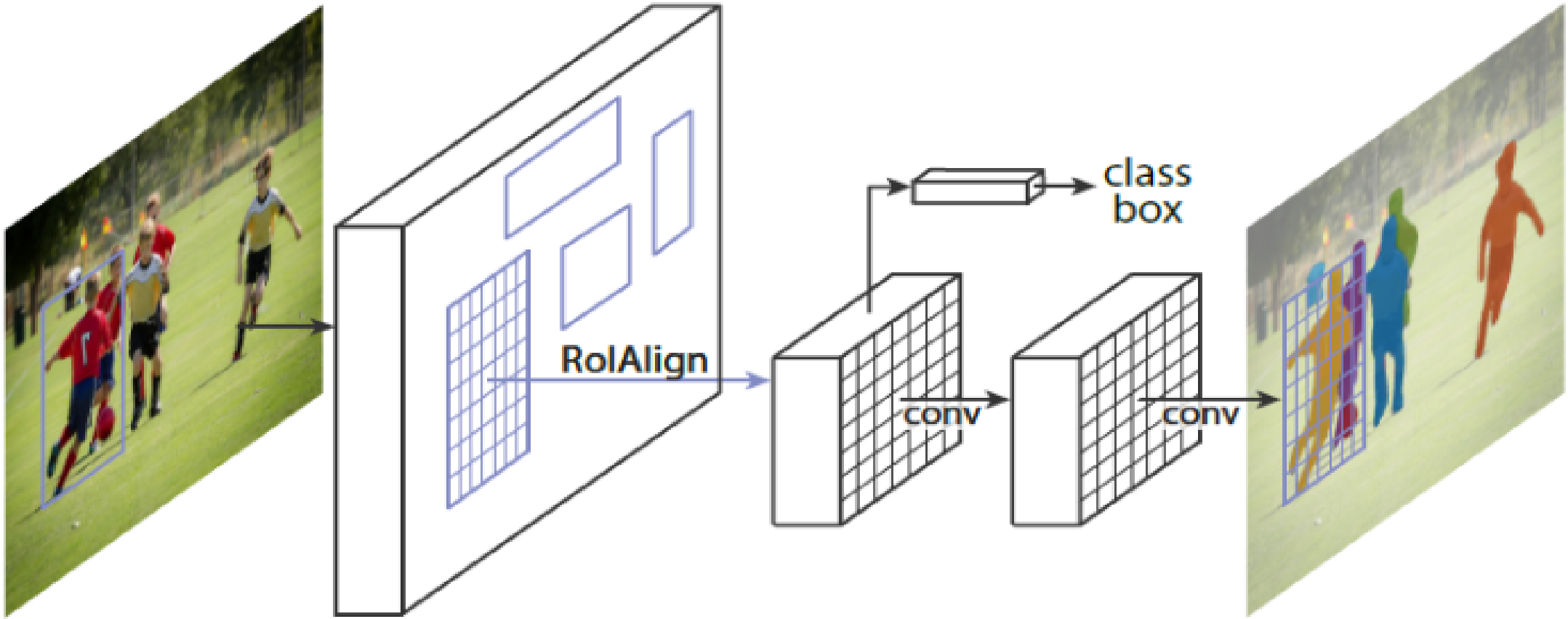


Images source: https://ai2-s2-public.s3.amazonaws.com/figures/2017-08-08/59cbe15b43e6ca172fce40786be68340f50be541/12-Figure1.1-1.png

Introduction to Mask R-CNN

- Mask-RCNN → Mask-Region Convolutional Neural Network
- An addition to the RCNN family, performing instance segmentation
- Improved over FasterRCNN
- A Full Convolutional Network (FCN) for predicting Mask for each class/object.
- 2 Stages:
  - Stage 1: RPN proposes candidate object bounding boxes.
  - Stage 2: Classify the Candidates, refine bounding boxes, and predict mask.

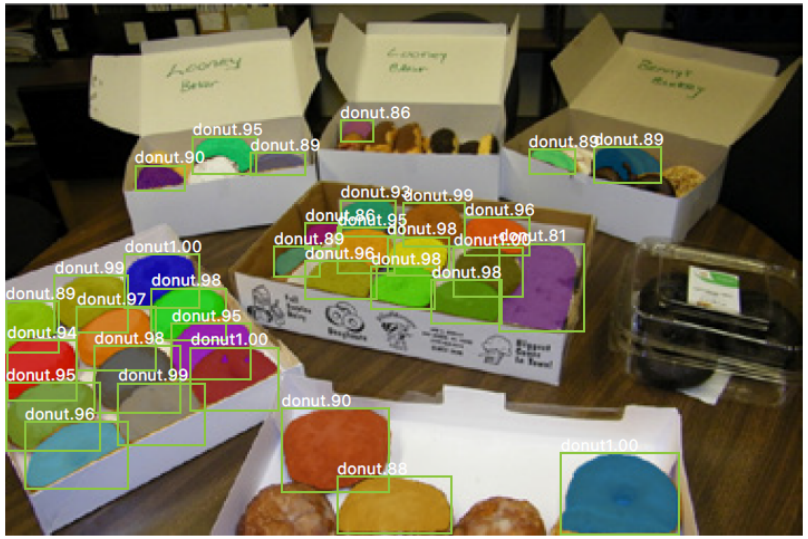
Introduction to Mask R-CNN



Images source: https://arxiv.org/abs/1703.06870, He et. al. “Mask RCNN” 2017

Introduction to Mask R-CNN

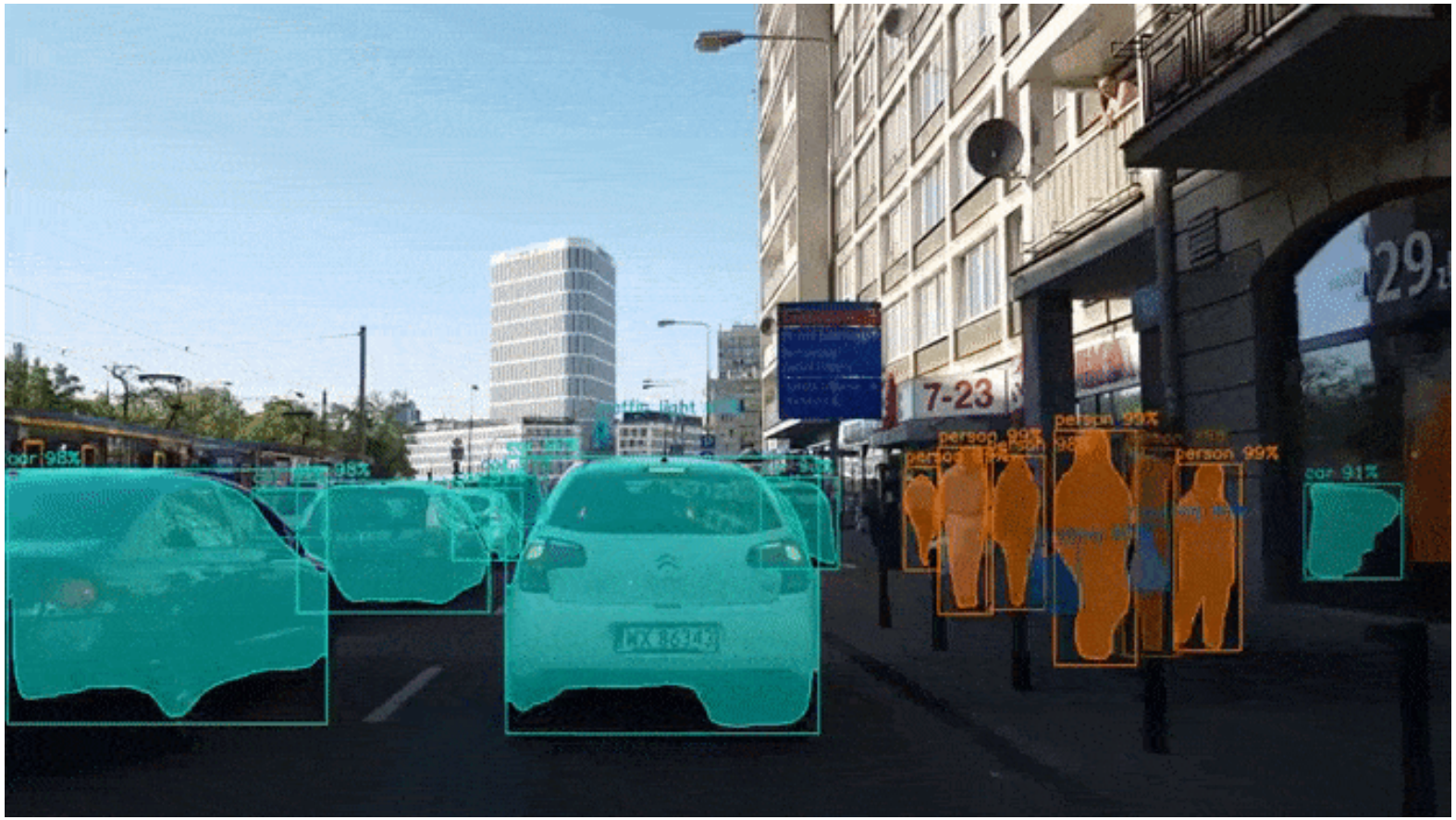
- Sample Results



Images source: https://arxiv.org/abs/1703.06870, He et. al. “Mask RCNN” 2017

Introduction to Mask R-CNN

- Sample Results on video:



Images source: https://github.com/matterport/Mask_RCNN

Advanced Topics -1

Foundation: 1 X 1 convolution

100	**100**	**100**	**0**	**0**	**0**
100	100	100	**0**	**0**	**0**
100	100	100	**0**	**0**	**0**
100	100	100	**0**	**0**	**0**
100	100	100	**0**	**0**	**0**
100	100	100	**0**	**0**	**0**

6 X 6 X 1 dimension image

Is this useful?

*

3

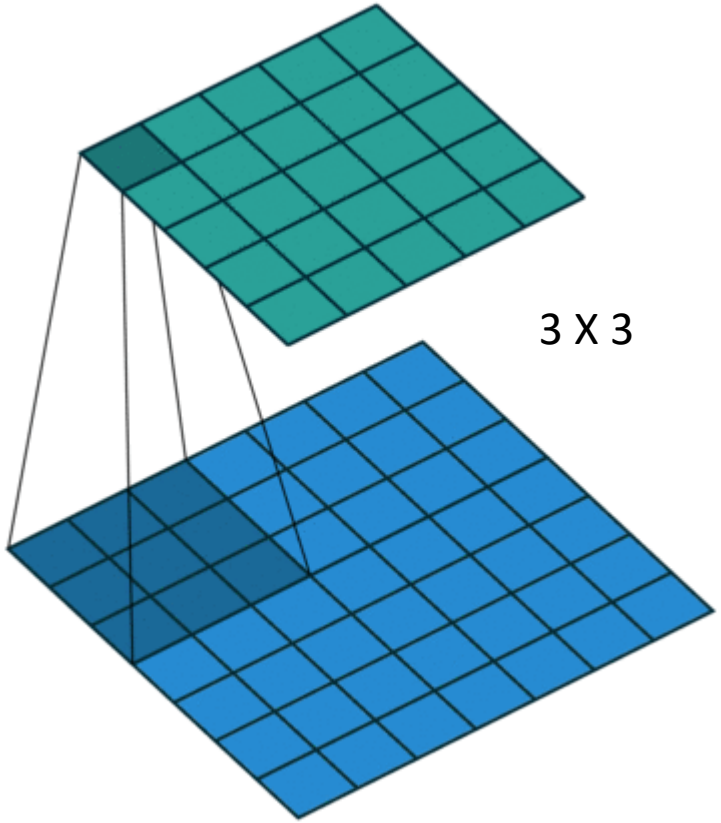
1 X 1 filter/Kernel

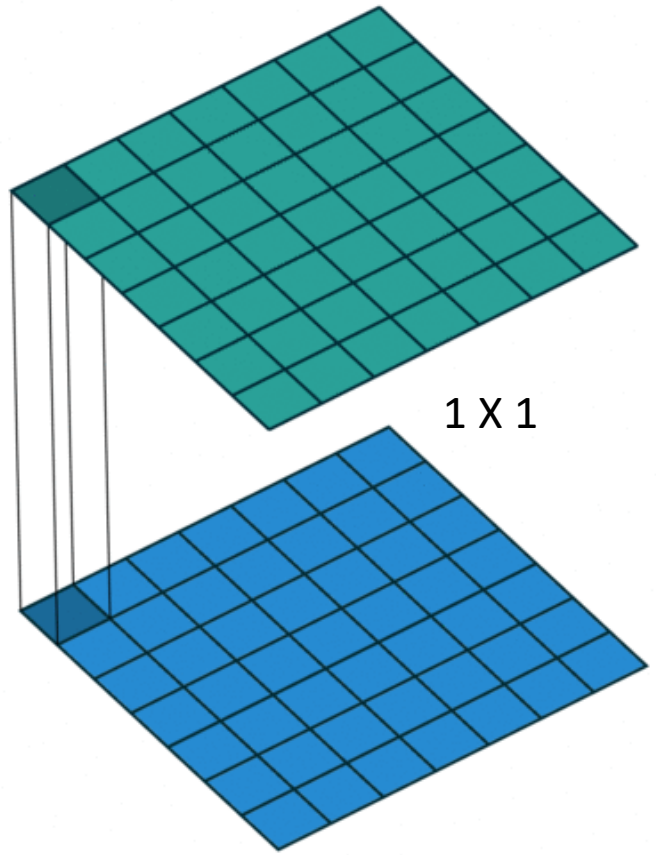
Convolution operator

300	**300**	**300**	**0**	**0**	**0**
300	**300**	**300**	**0**	**0**	**0**
300	**300**	**300**	**0**	**0**	**0**
300	**300**	**300**	**0**	**0**	**0**
300	**300**	**300**	**0**	**0**	**0**
300	**300**	**300**	**0**	**0**	**0**

6 X 6 X 1 dimension volume

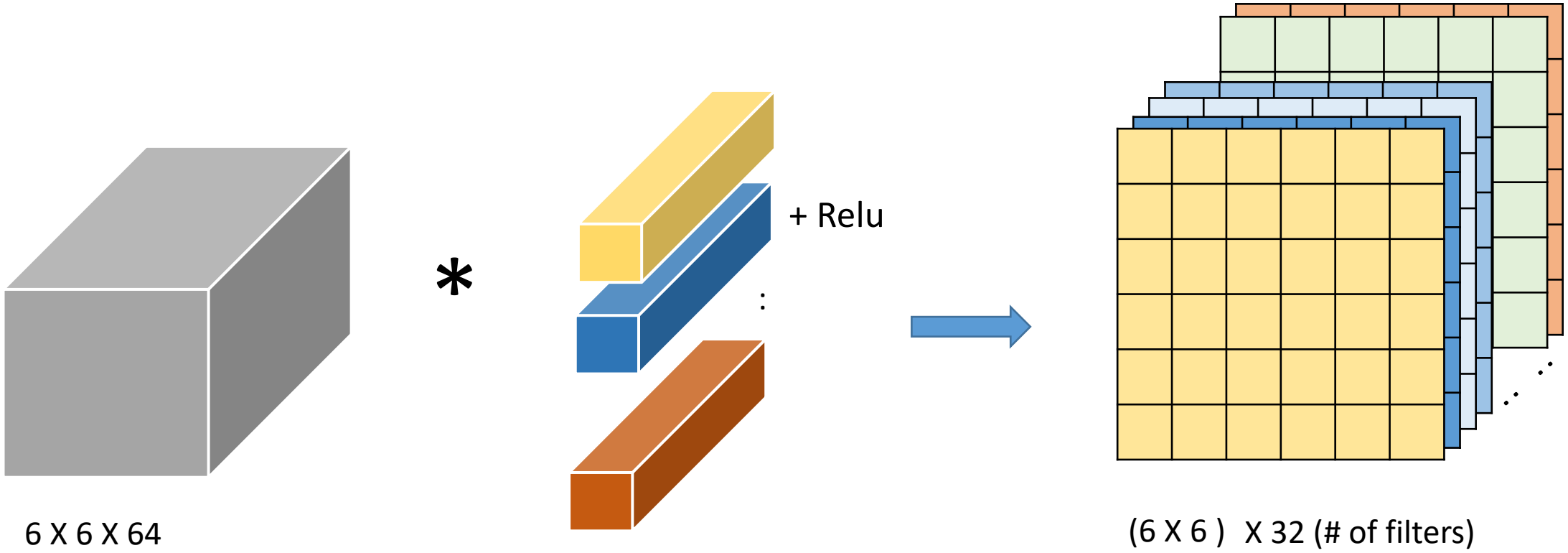
Foundation: 1 X 1 convolution





Images source: https://iamaaditya.github.io/2016/03/one-by-one-convolution/

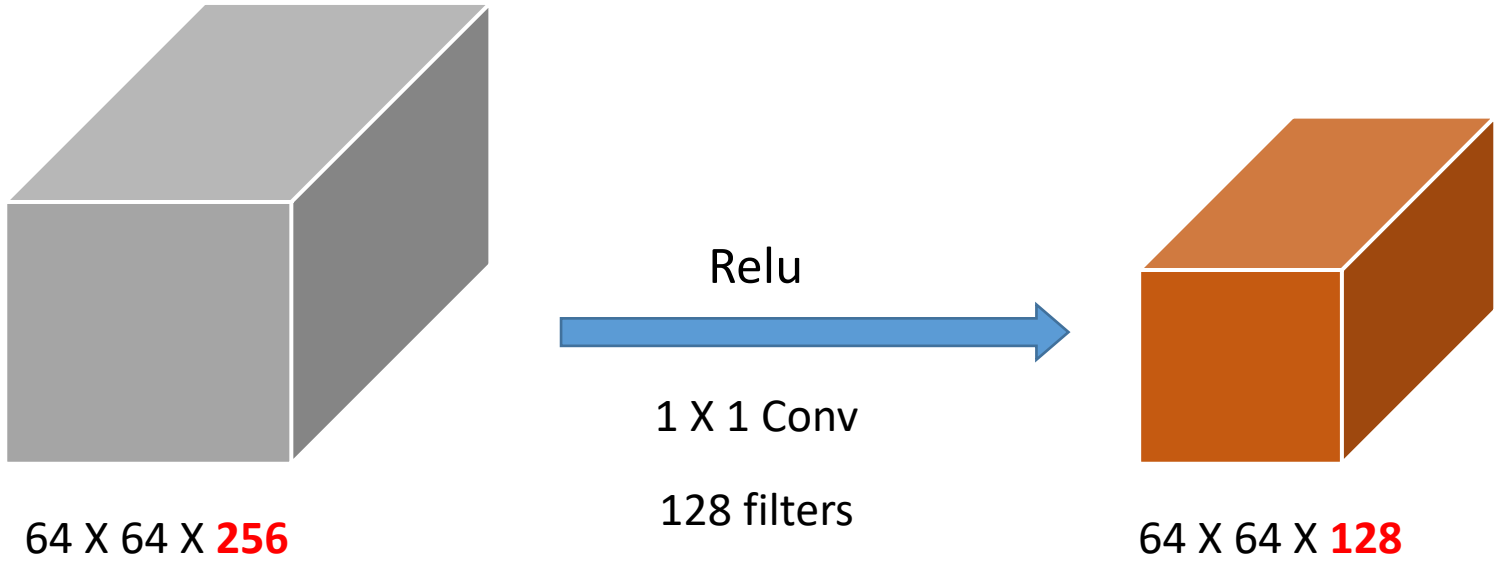
Foundation: 1 X 1 convolution



So, (6 X 6 X 64) → (6 X 6 X 32) … reduced!

Reference and source: https://www.coursera.org/lecture/convolutional-neural-networks/networks-in-networks-and-1x1-convolutions-ZTb8x

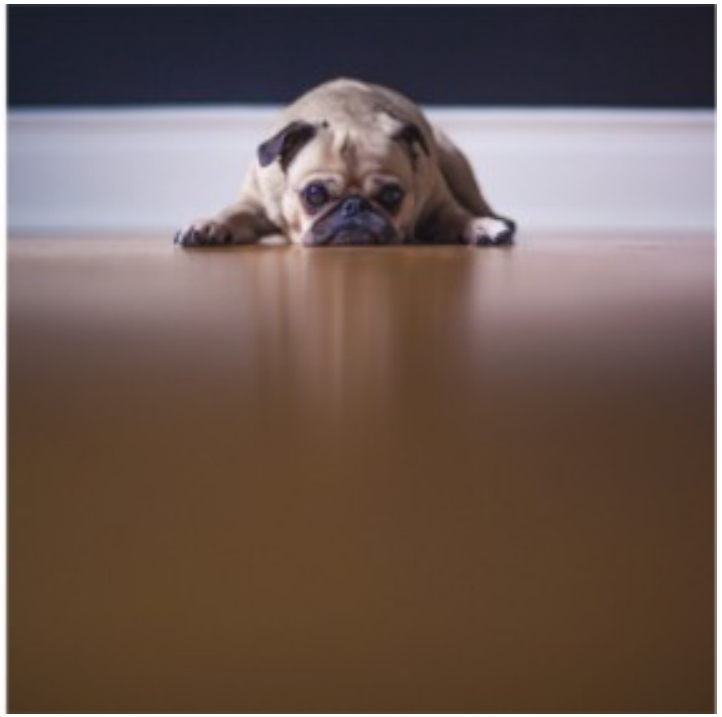
Foundation: 1 X 1 convolution



Reference and source: https://www.coursera.org/lecture/convolutional-neural-networks/networks-in-networks-and-1x1-convolutions-ZTb8x

Inception - Motivation





- Large variation in object size
- How to choose the right filter size?

- Large filter preferred for large objects
- Small filters for small objects

Images source: https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202

Inception - Motivation

Designing CNN requires:

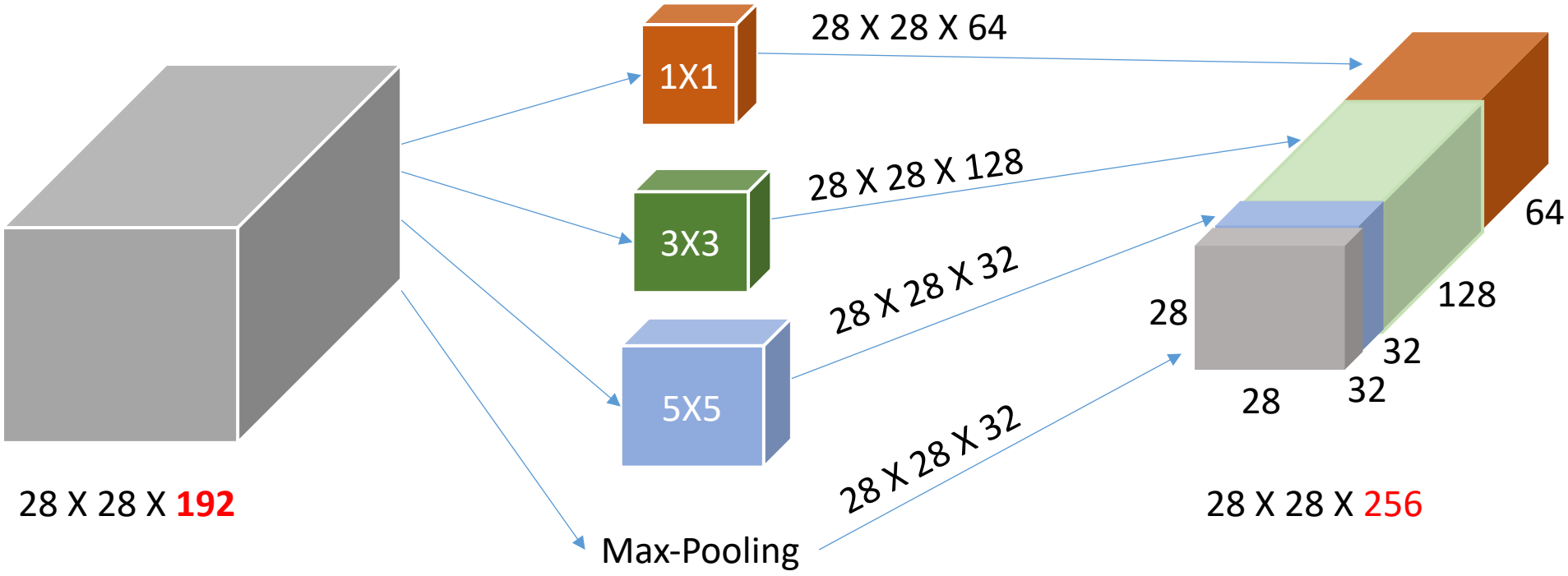
- Deciding filter size and number
- Number and type of layers etc.

Inception suggests:

- Use filters with different size together!
- Use different types of layers (CONV, POOL etc.) together

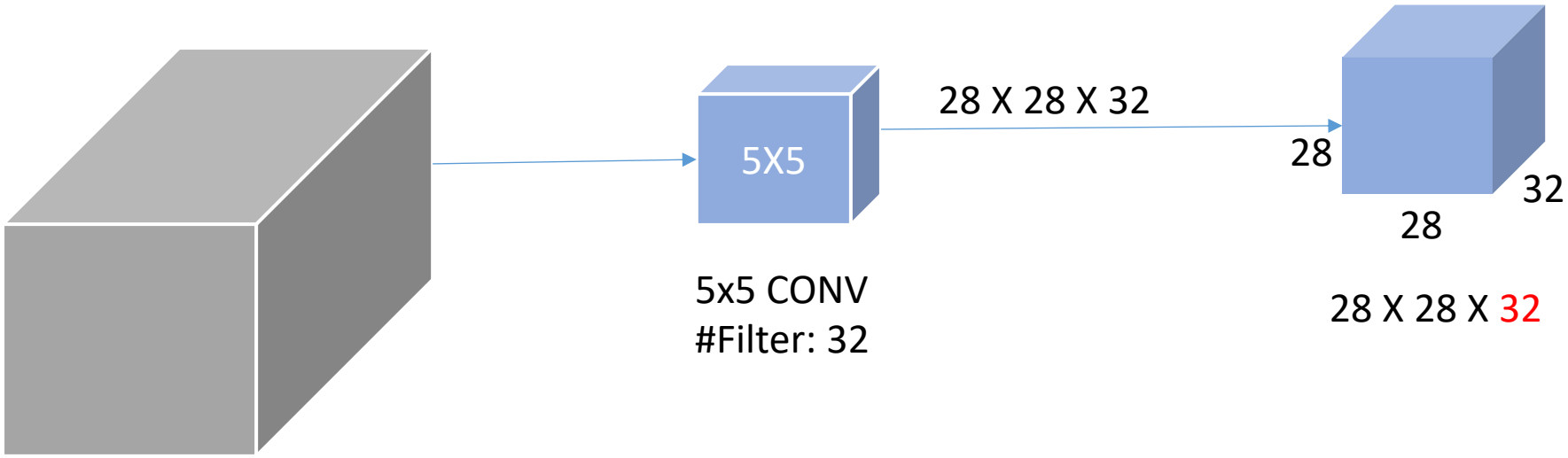
Result → Complicated Architecture! & better performance

Inception - Motivation



Reference, Images source: https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202

Computation cost

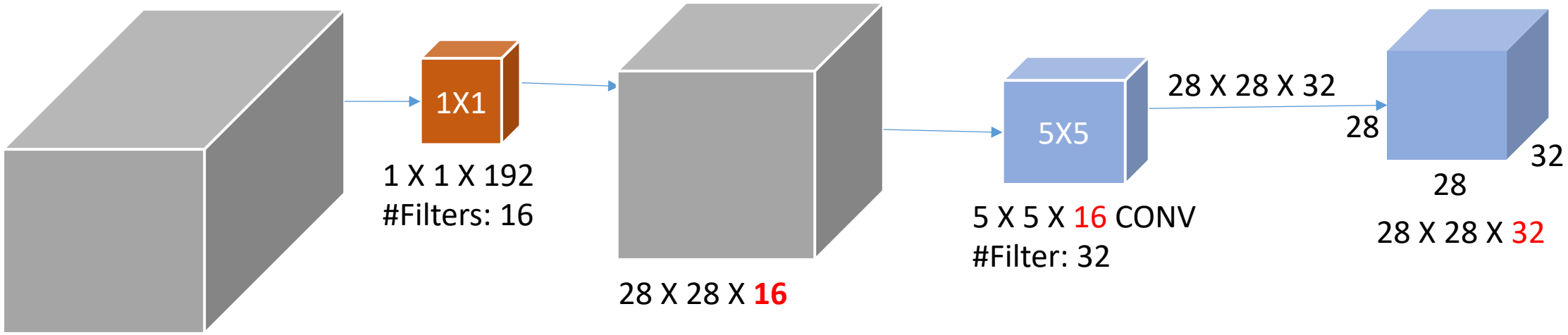


Computation Cost:

28 X 28 X 32 X 5 X 5 X 192 ≈ **120M multiplications!**

Quite expensive !

Reduce Computation cost using 1X1 CONV



Computation Cost:

1X1: 28 X 28 X 16 X 192 ≈ 2.4M multiplications!

5X5: 28 X 28 X 32 X 5 X 5 X 16 ≈ 10M multiplications!

Total : **12.4M multiplications**! ← **Reduced by 10 times!**

Bottleneck Layer

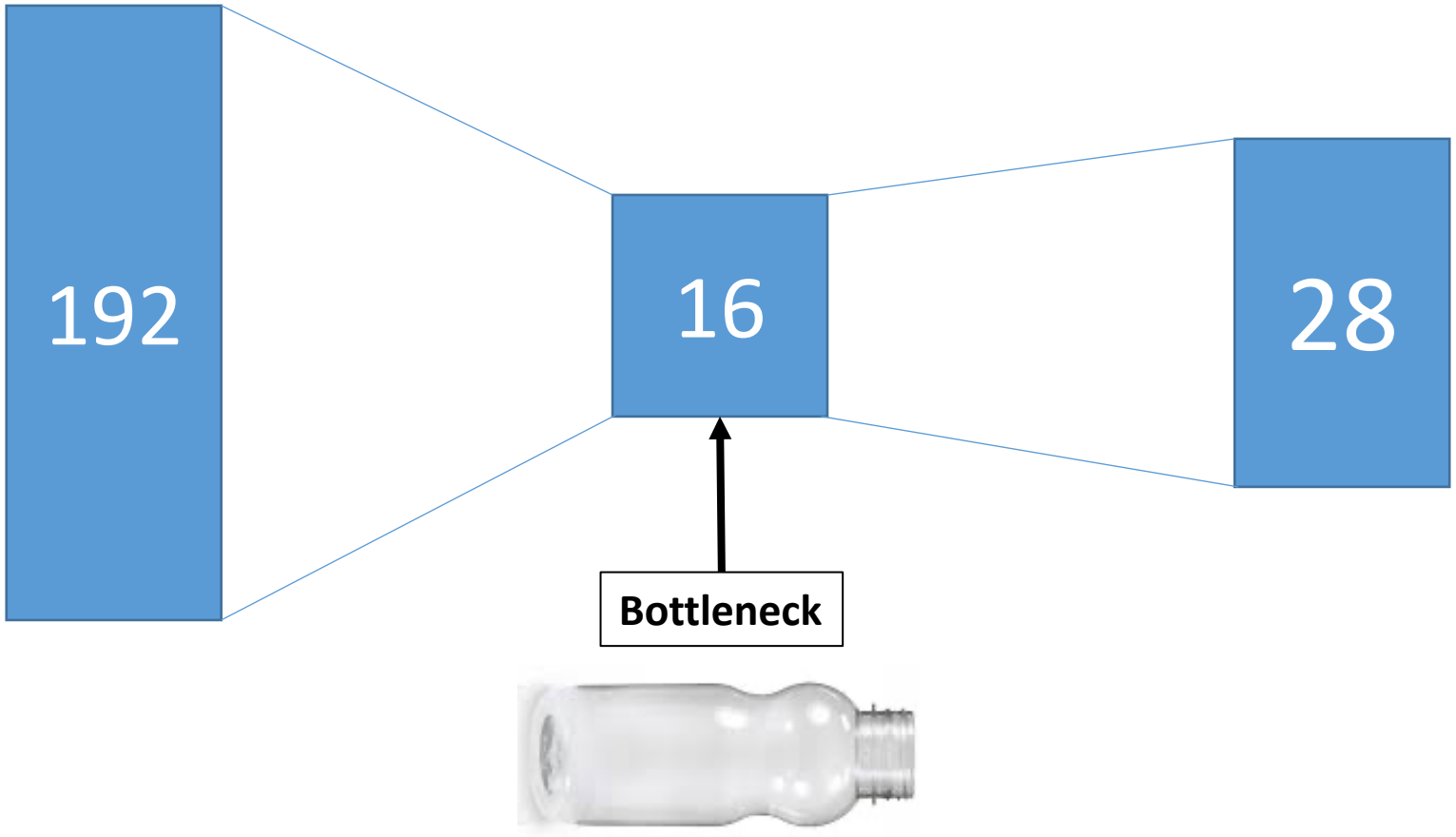
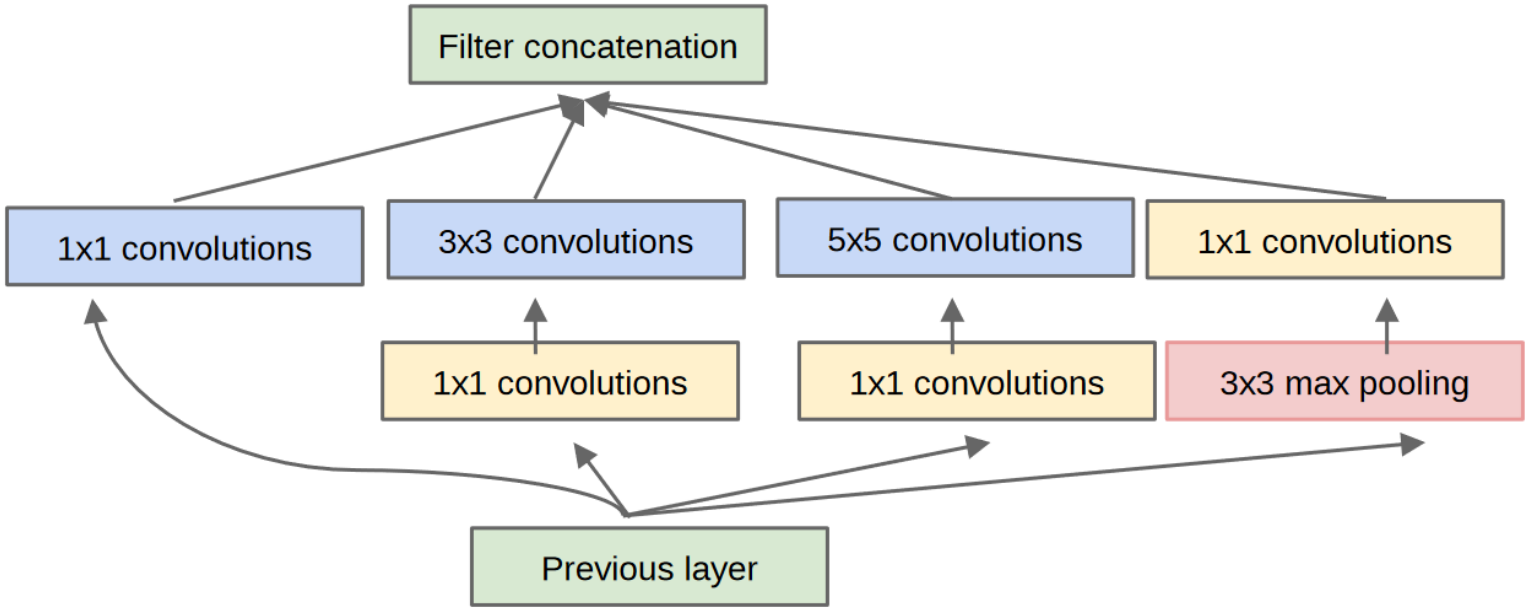


Image adapted from and Reference: http://www.pabloruizruiz10.com/resources/CNNs/1x1Convolutions.pdf

Inception Module V1



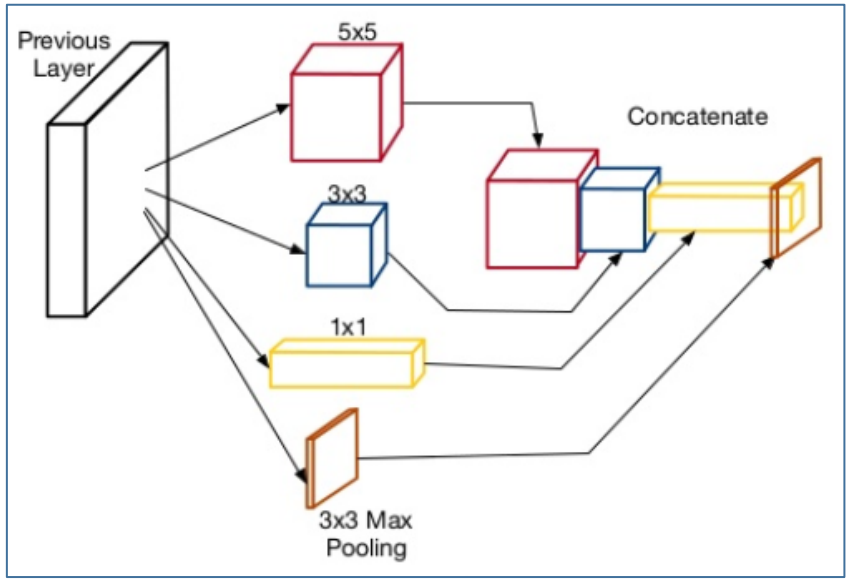
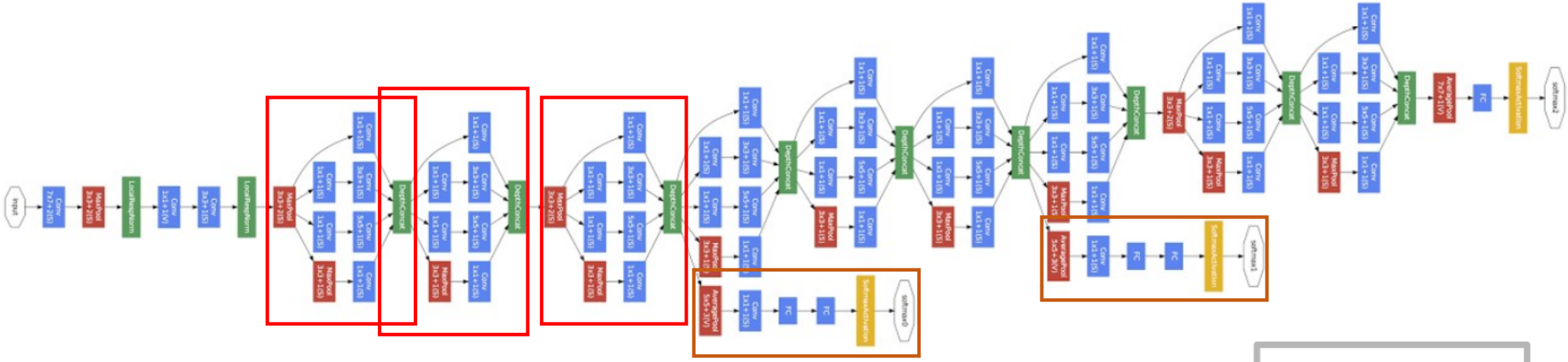


Image Source and reference: https://www.cs.unc.edu/~wliu/papers/GoogLeNet.pdf

Inception Network

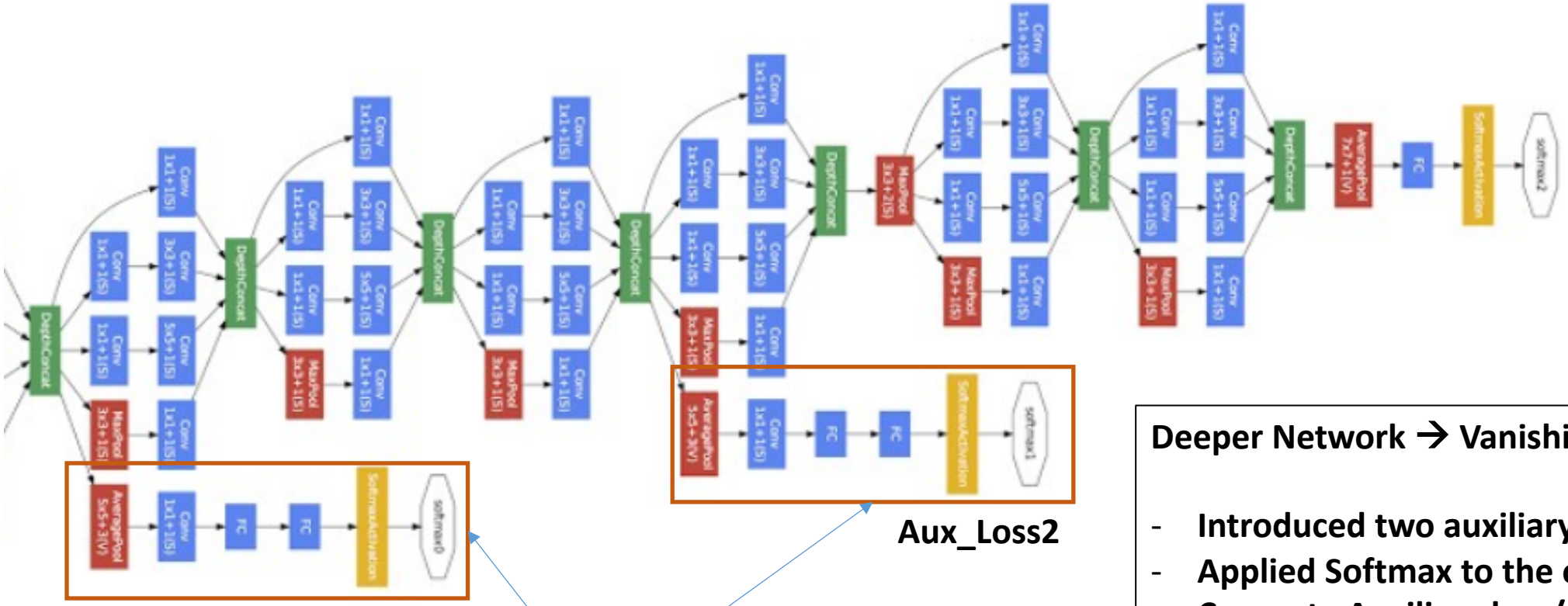


GoogleNet(2014): 9 Inception modules stacked together

Convolution
Pooling
Softmax
Other

Image Source and reference: https://www.cs.unc.edu/~wliu/papers/GoogLeNet.pdf

Inception Network



Deeper Network → Vanishing Gradient

- **Introduced two auxiliary classifier**
- **Applied Softmax to the output**
- **Compute Auxiliary loss/cost**
- **Only used for training**

Total Loss/cost = Real_Loss + 0.3 X Aux_Loss1 + 0.3 X Aux_Loss1

Image Source and reference: https://www.cs.unc.edu/~wliu/papers/GoogLeNet.pdf

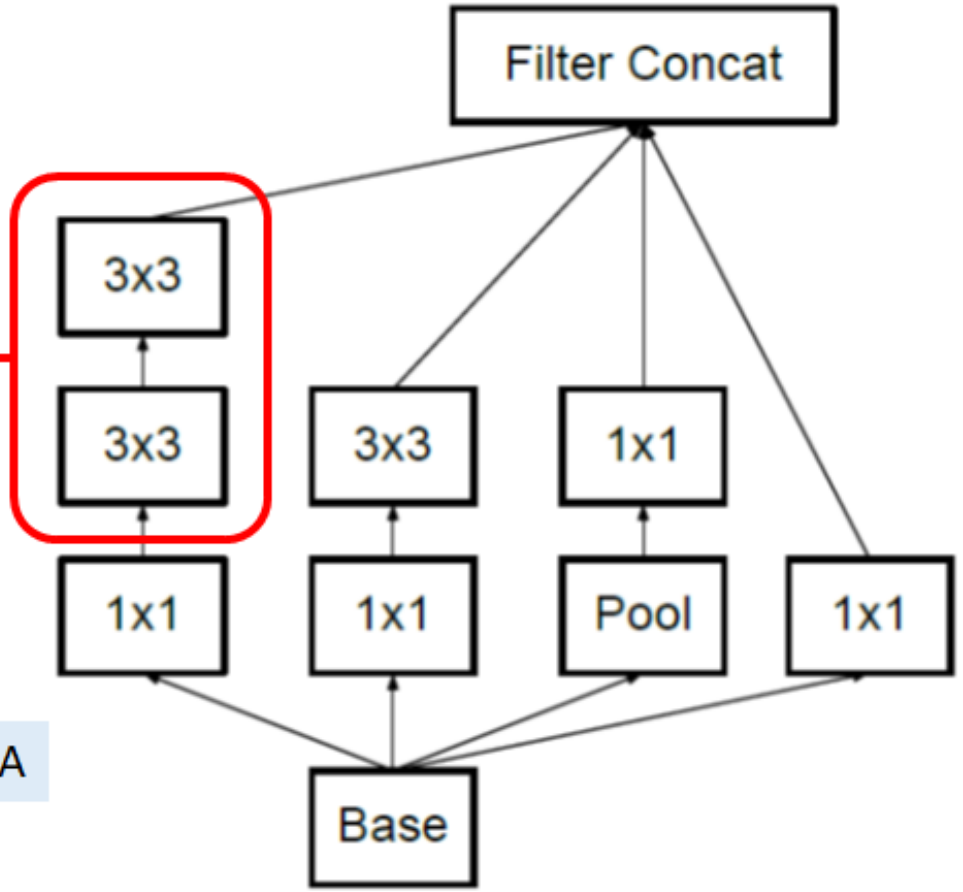
Inception V2 & V3 Modules

Authors suggested 3 different modules
-Factorizing Convolutions:
Reducing the number of parameters

1 layer of 5×5 filter, #parameters = **5×5=25**
2 layers of 3×3 filters, #parameters = **3×3+3×3=18**

Number of parameters is reduced by 28%

5×5 in GoogLeNet (Inception-v1)



Inception Module A

Image Source and reference: https://medium.com/@sh.tsang/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c

Inception V2 & V3 Modules

3×3 filter, #parameters = **3×3=9**
3×1 and 1×3 filters, #parameters = **3×1+1×3=6**
Number of parameters is reduced by 33%

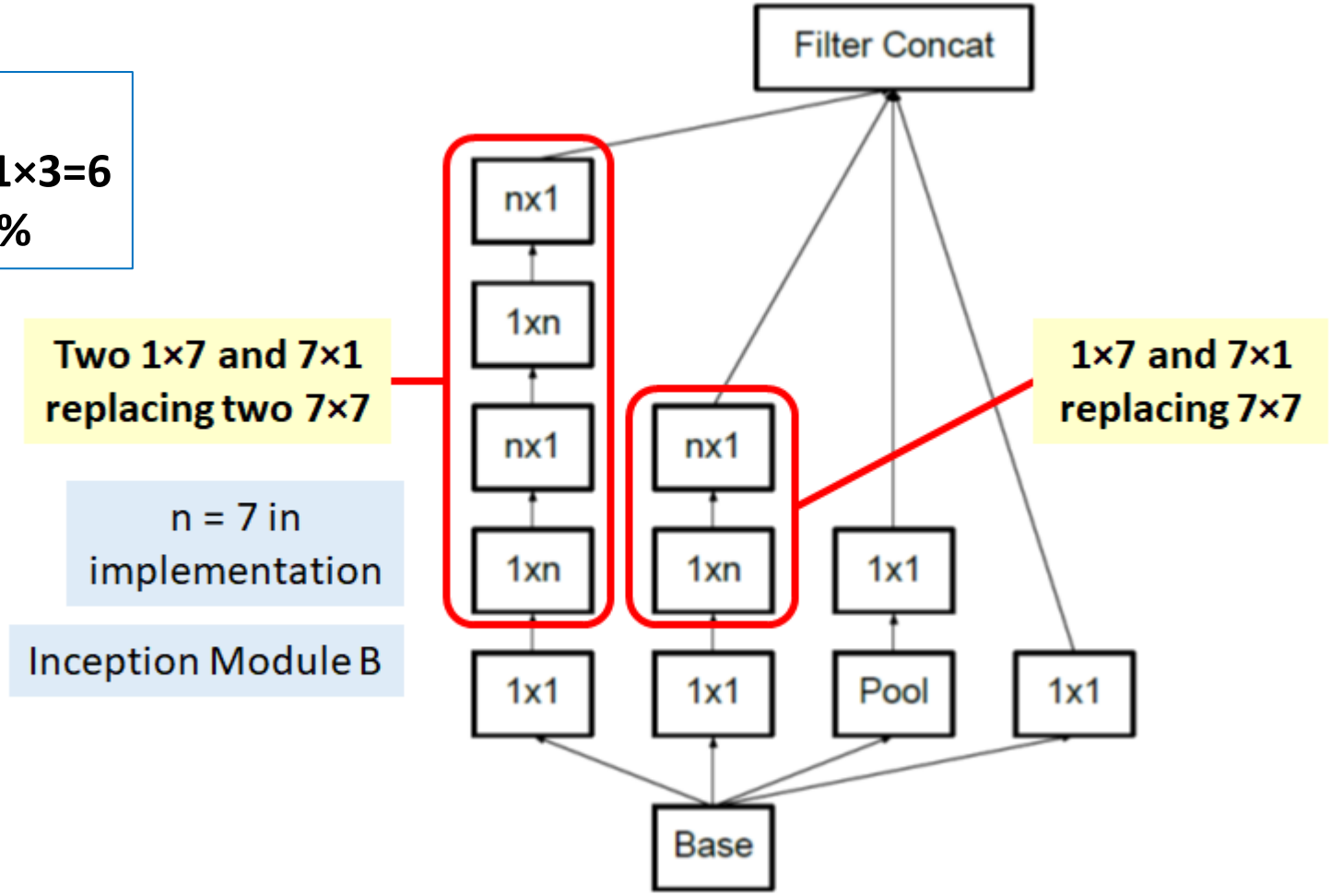


Image Source and reference: https://medium.com/@sh.tsang/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c

Inception V2 & V3 Modules

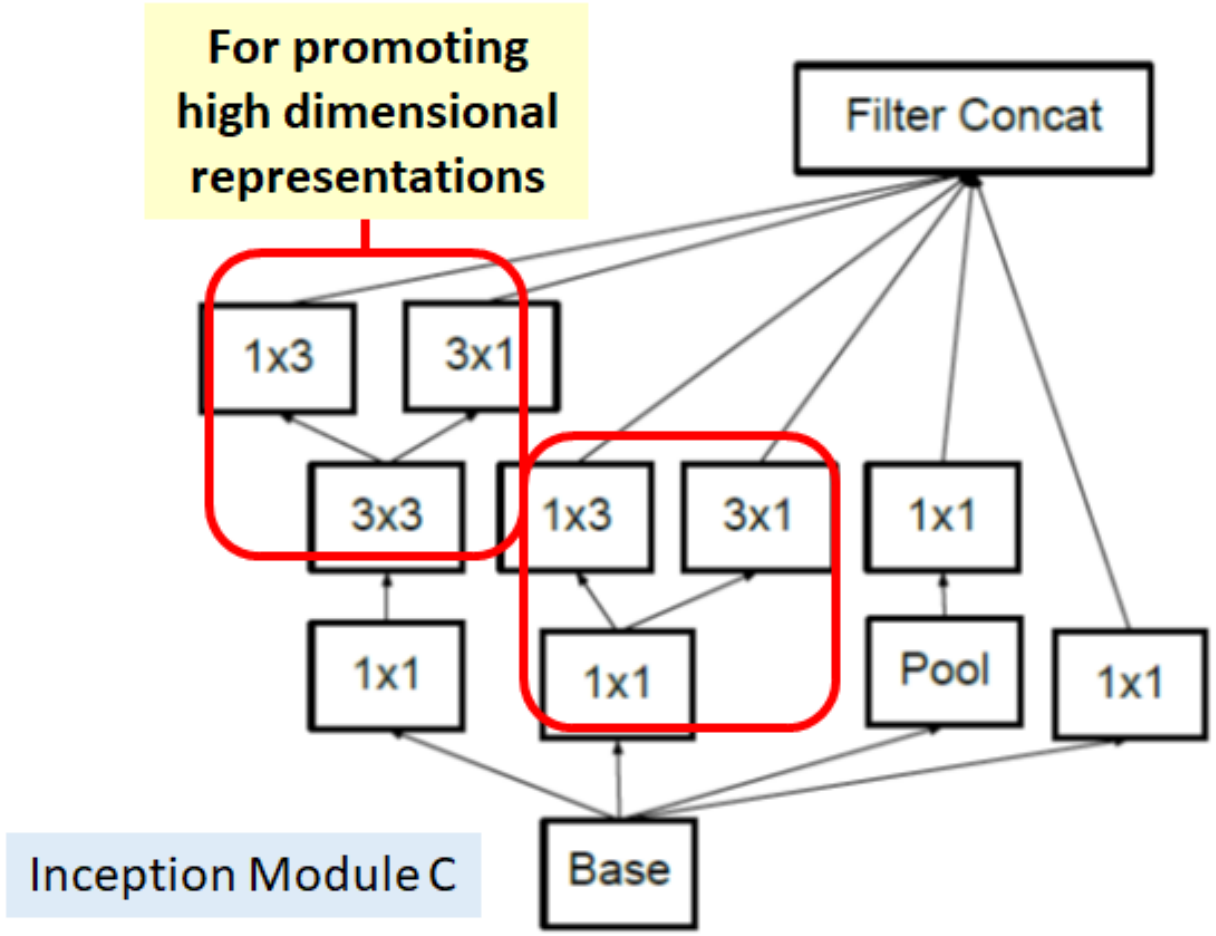
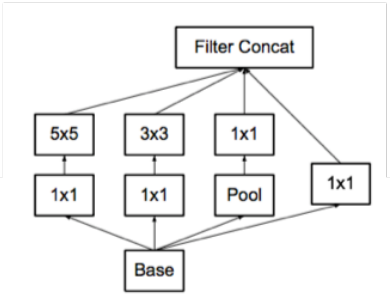
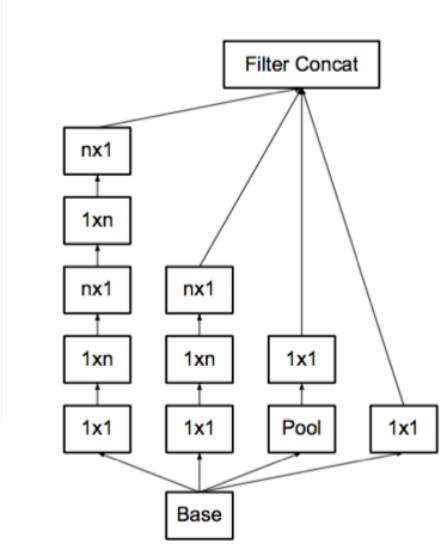
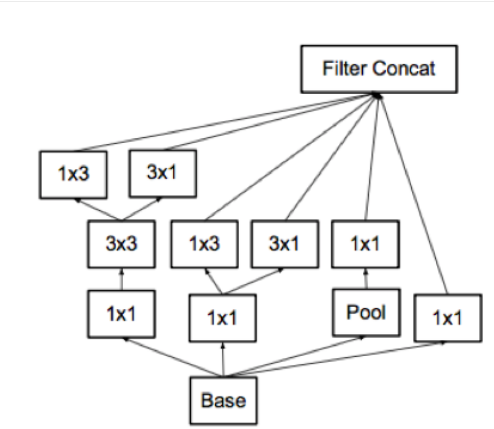


Image Source and reference: https://medium.com/@sh.tsang/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c

Inception V3 Architecture







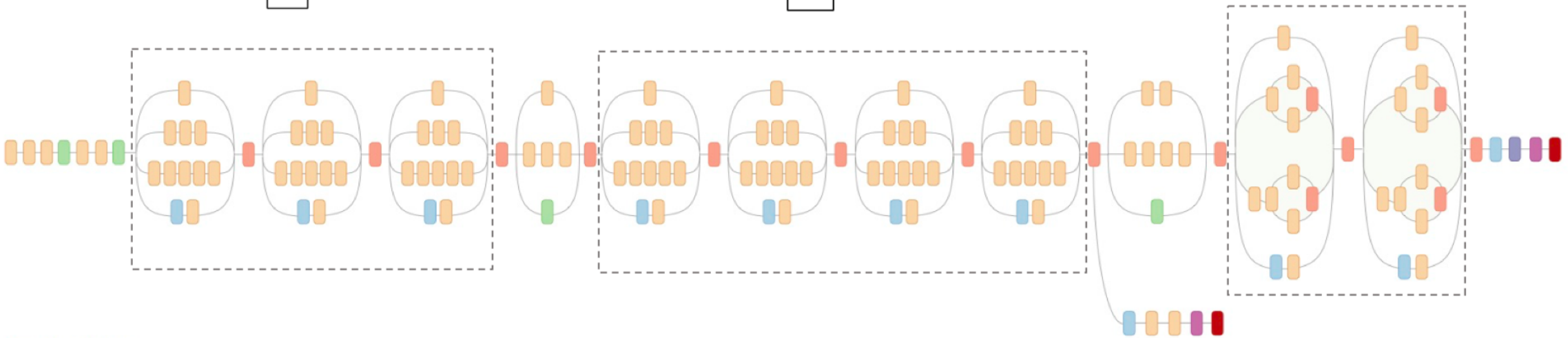


Image Source and reference: https://www.jeremyjordan.me/convnet-architectures/
For More details: https://cloud.google.com/tpu/docs/inception-v3-advanced

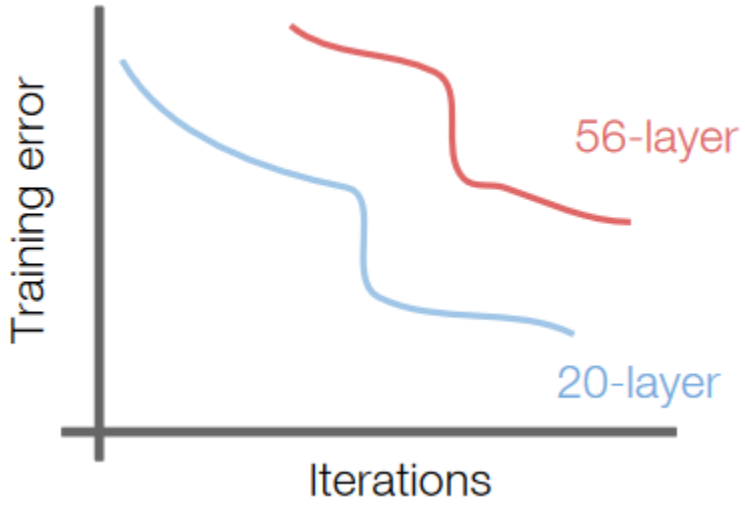
ResNet

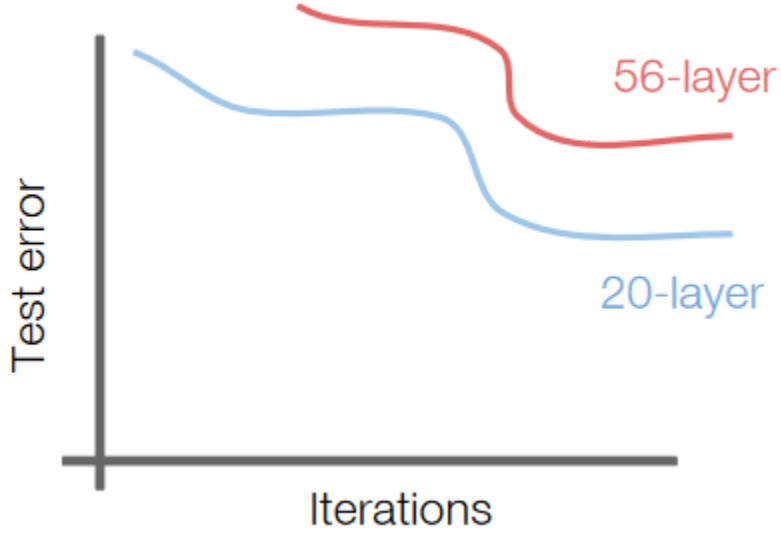
- Deep Residual networks (ResNet) → Skip connections
- Enabled the development of the much deeper networks (100s of layers!)
- ResNet is composed of Residual Blocks were introduced!
- Degradation problem: Adding more layers eventually have negative effect on the final performance.

Image Source and reference: https://www.jeremyjordan.me/convnet-architectures/

ResNet

What wrong with this curves? Overfitting?



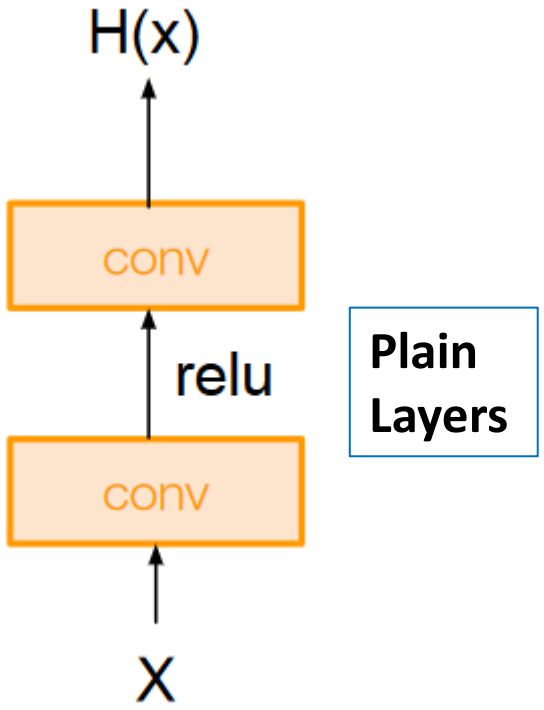


- 56 layer model is not better than the 20 layers!
- What happens when we keep add more layers to a plain CNN to make it deeper?

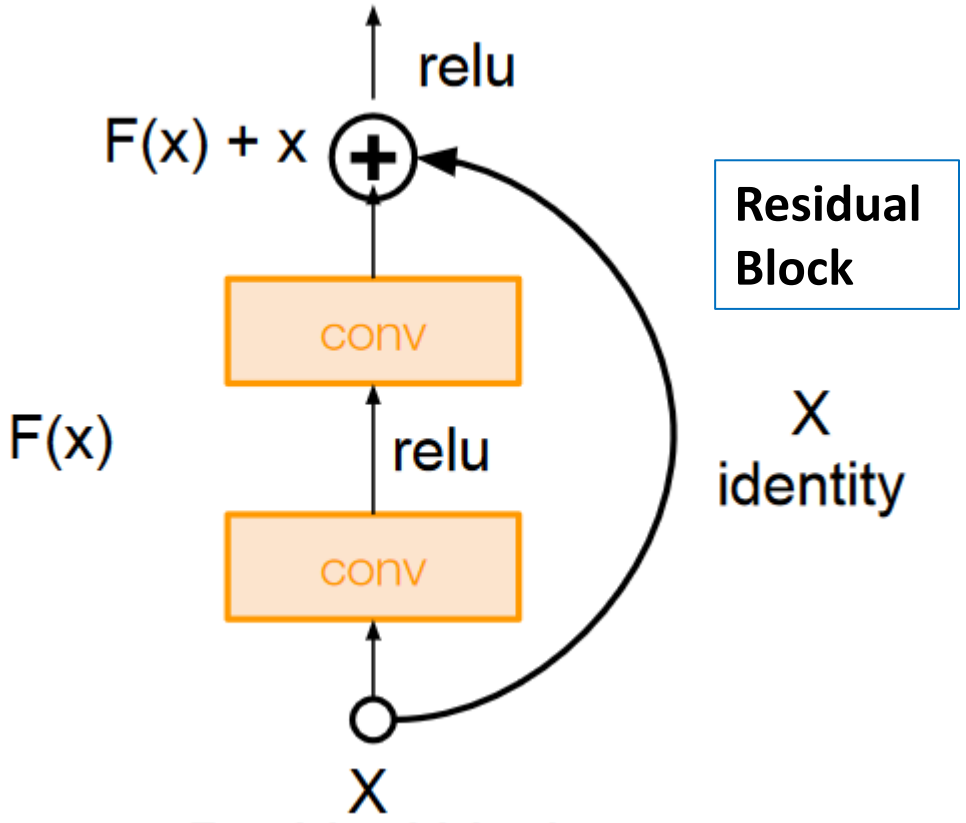
Image Source and reference: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

ResNet

In principle deeper model should perform better than shallow CNNs



Plain Layers



Residual Block

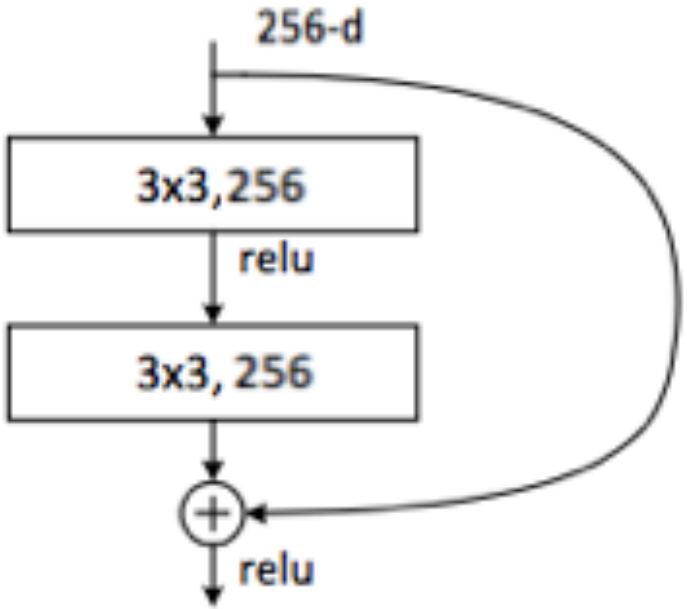
Image Source and reference: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

ResNet

Image Source and reference: https://www.jeremyjordan.me/convnet-architectures/

ResNet

ResNet 34
residual block



ResNet 50
residual block

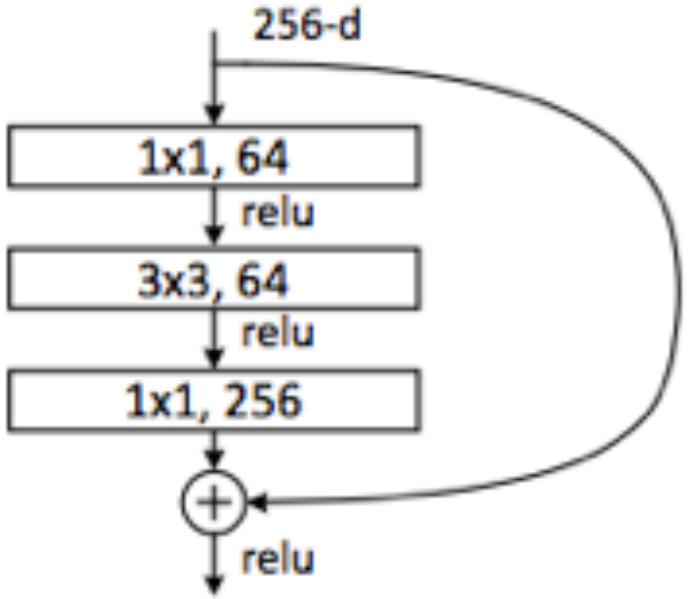
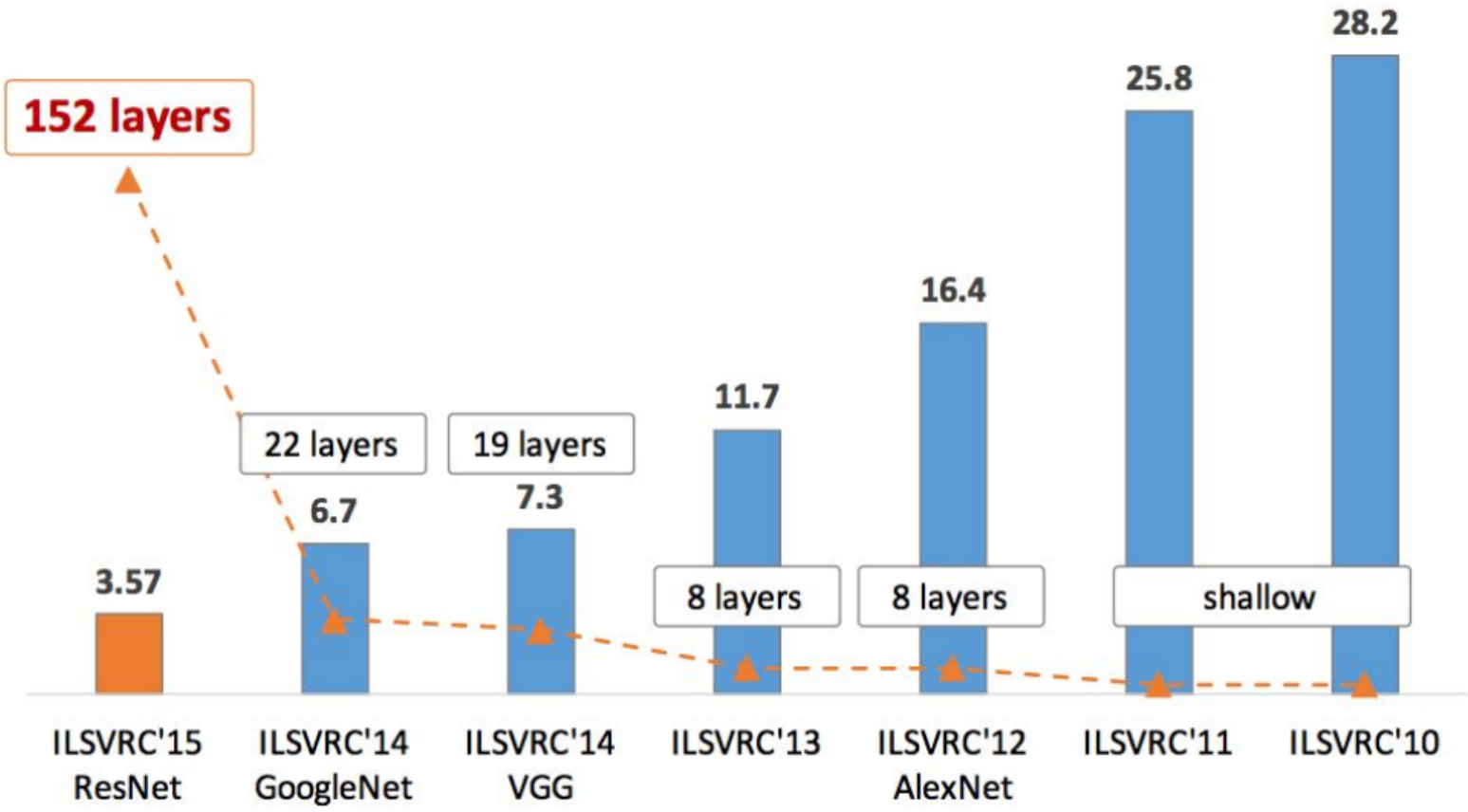


Image Source and reference: https://www.jeremyjordan.me/convnet-architectures/

Summary



Reference and image Source: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf