

# Short Course on Deep Learning and Convolutional Neural Network

Day 1: Machine Learning Basics  
Introduction to Image Processing

# Type of Machine Learning Systems

# Type of Machine Learning Systems

Supervised Learning  
Unsupervised Learning  
Semi-supervised Learning  
Reinforcement Learning

Depending on whether the system is trained with human supervision

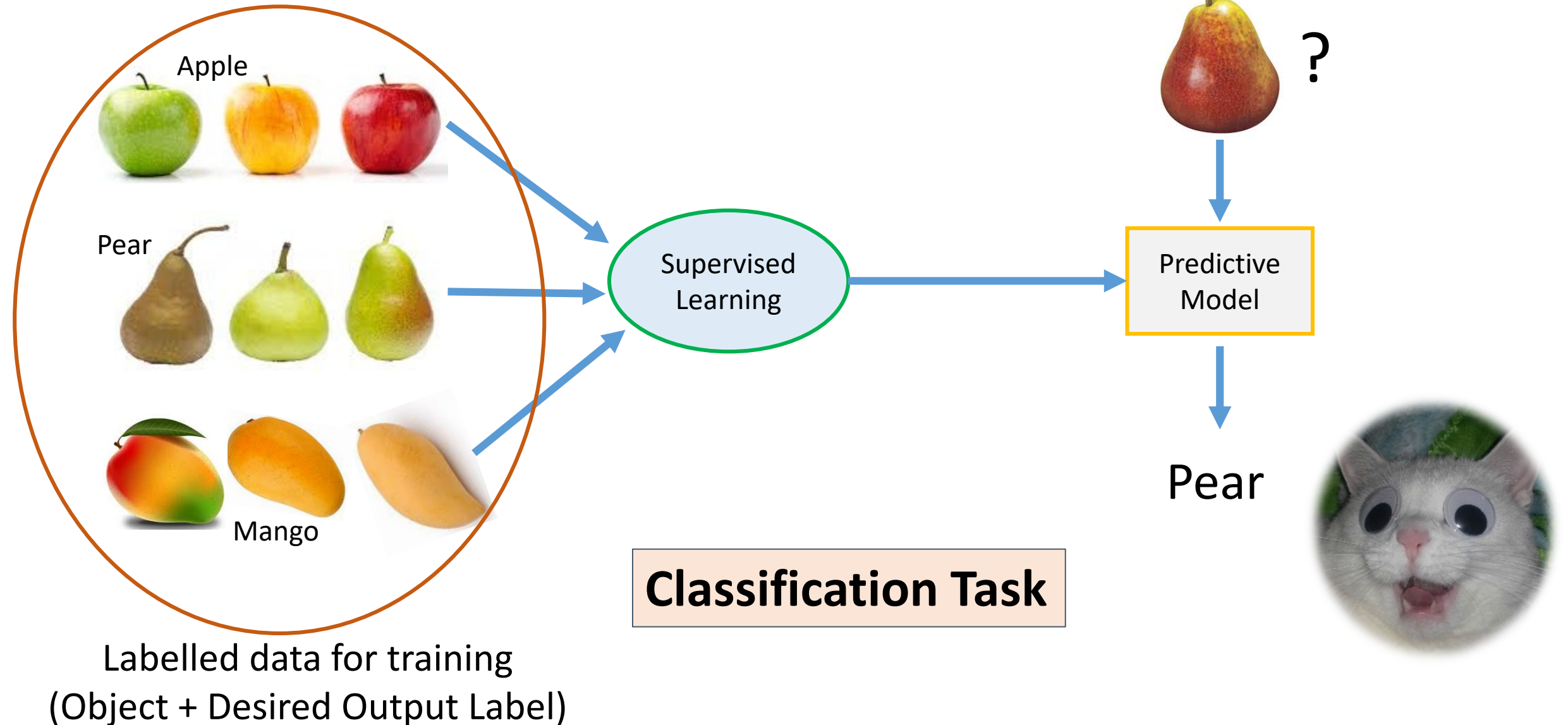
Whether System can learn on the fly

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



# Supervised Learning

## House Price prediction

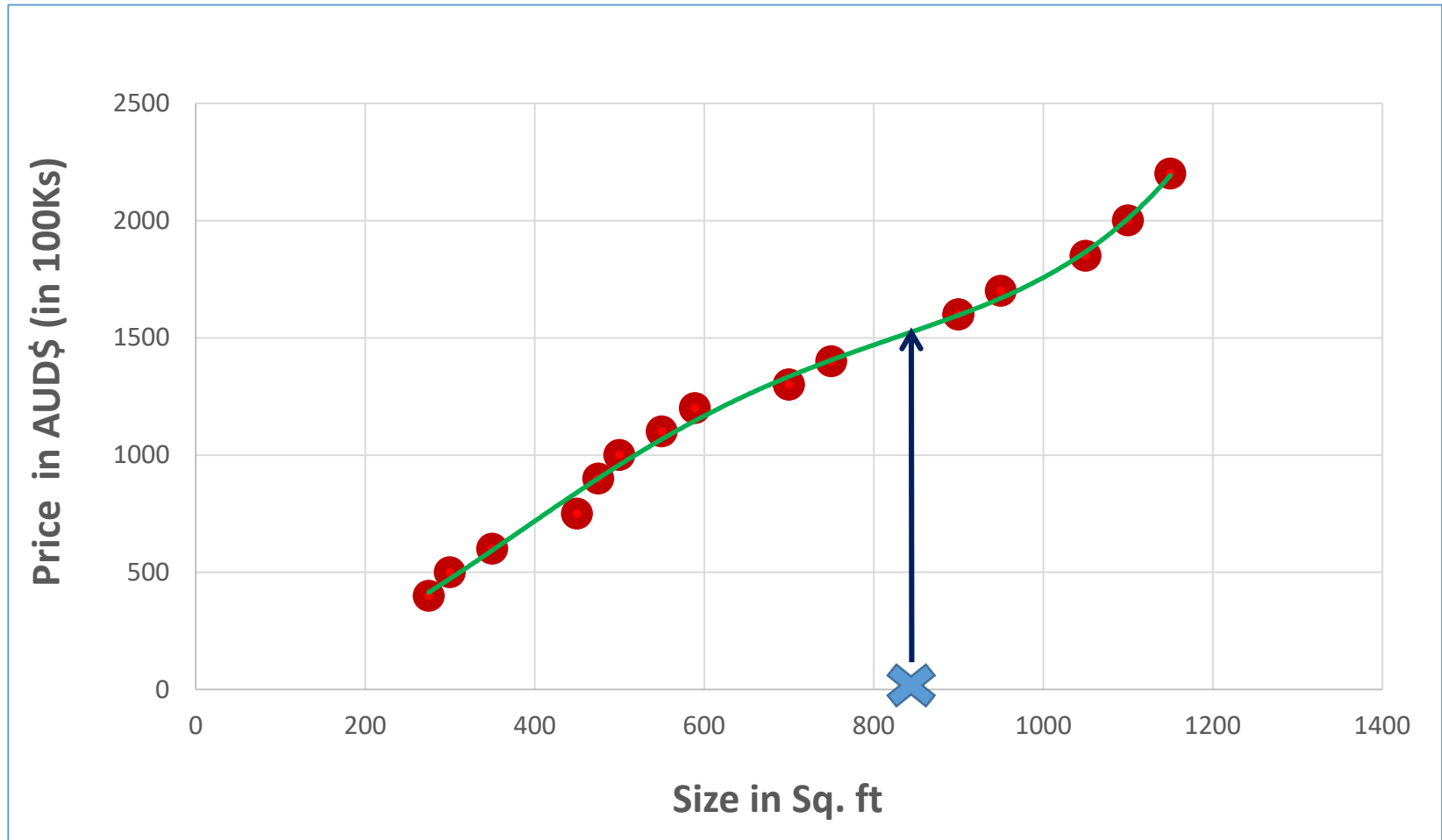
Feature:

- Size of the house

To Predict:

- Price of the house

**Regression Task**

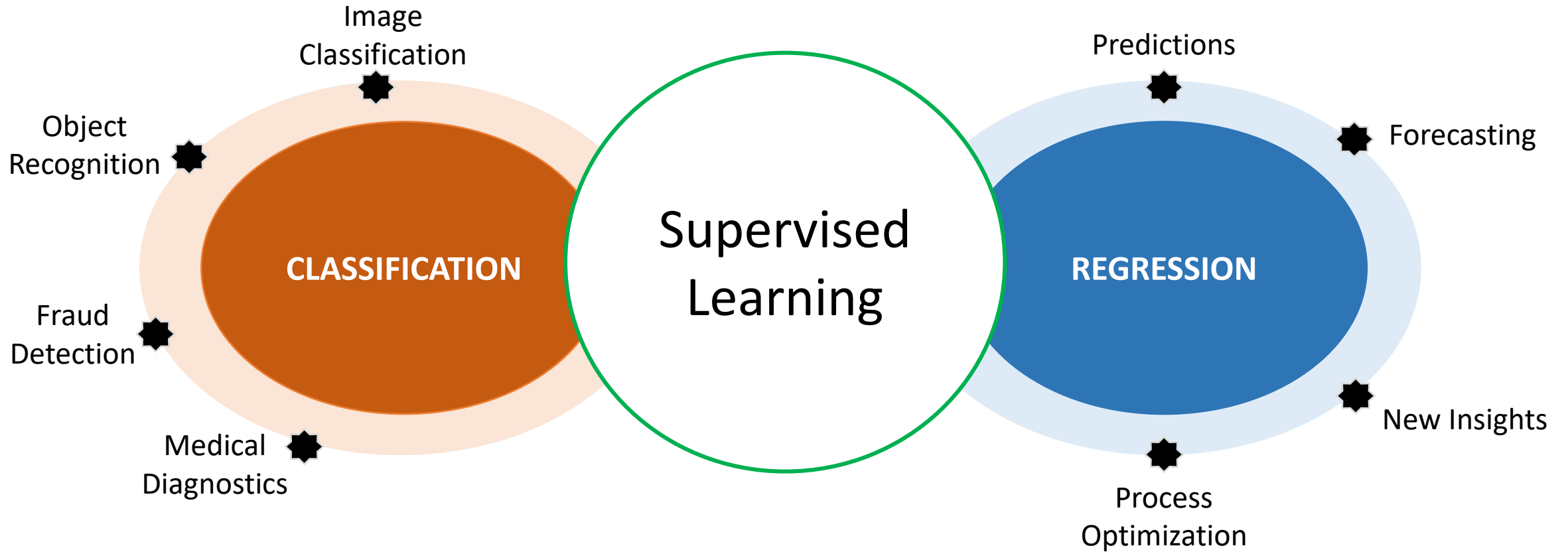


# Supervised Learning

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)

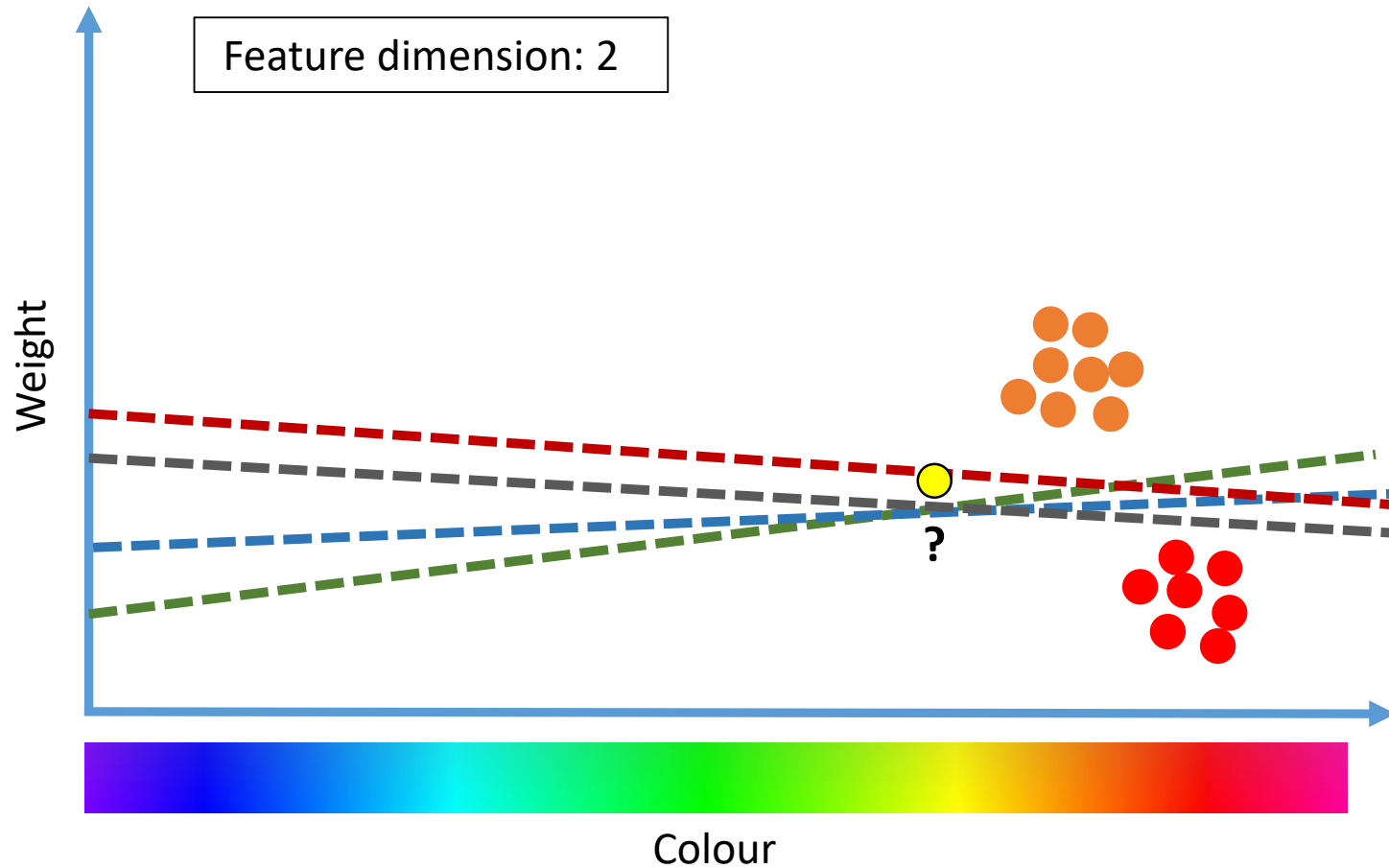
- A Support Vector Machine is a very powerful and versatile Machine Learning model, capable of performing linear or non-linear classification, regression, and also outlier detection.
- Defined by a separating hyperplane
- Suitable for small or medium sized datasets

## 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

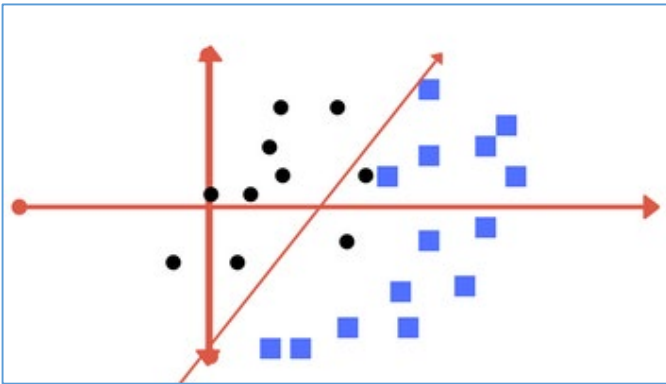
# we create an instance of SVM and fit out data. We do not scale our
# data since we want to plot the support vectors
C = 1.0 # SVM regularization parameter
models = (svm.SVC(kernel='linear', C=C),
          svm.LinearSVC(C=C),
          svm.SVC(kernel='rbf', gamma=0.7, C=C),
          svm.SVC(kernel='poly', 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://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](https://en.wikipedia.org/wiki/Iris_flower_data_set)

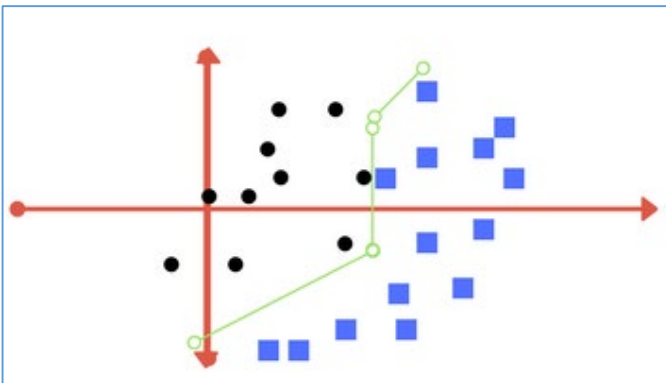
# Support Vector Machines (SVM)

## SVM Parameters: Kernel, Gamma, Regularization (C)

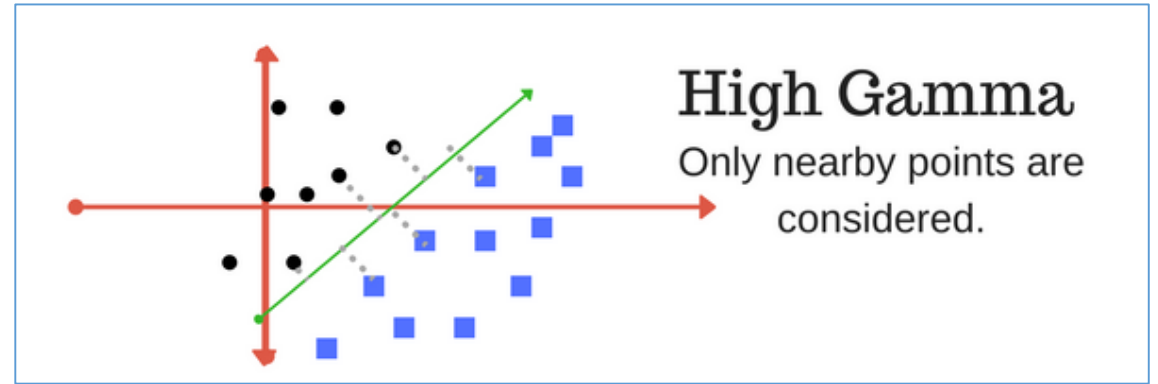
Low  
Regularization  
value



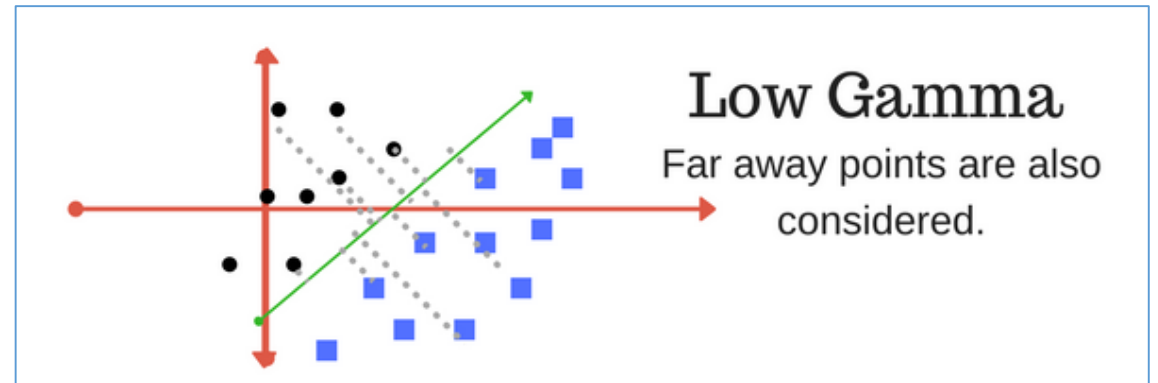
High  
Regularization  
value



**High Gamma**  
Only nearby points are considered.

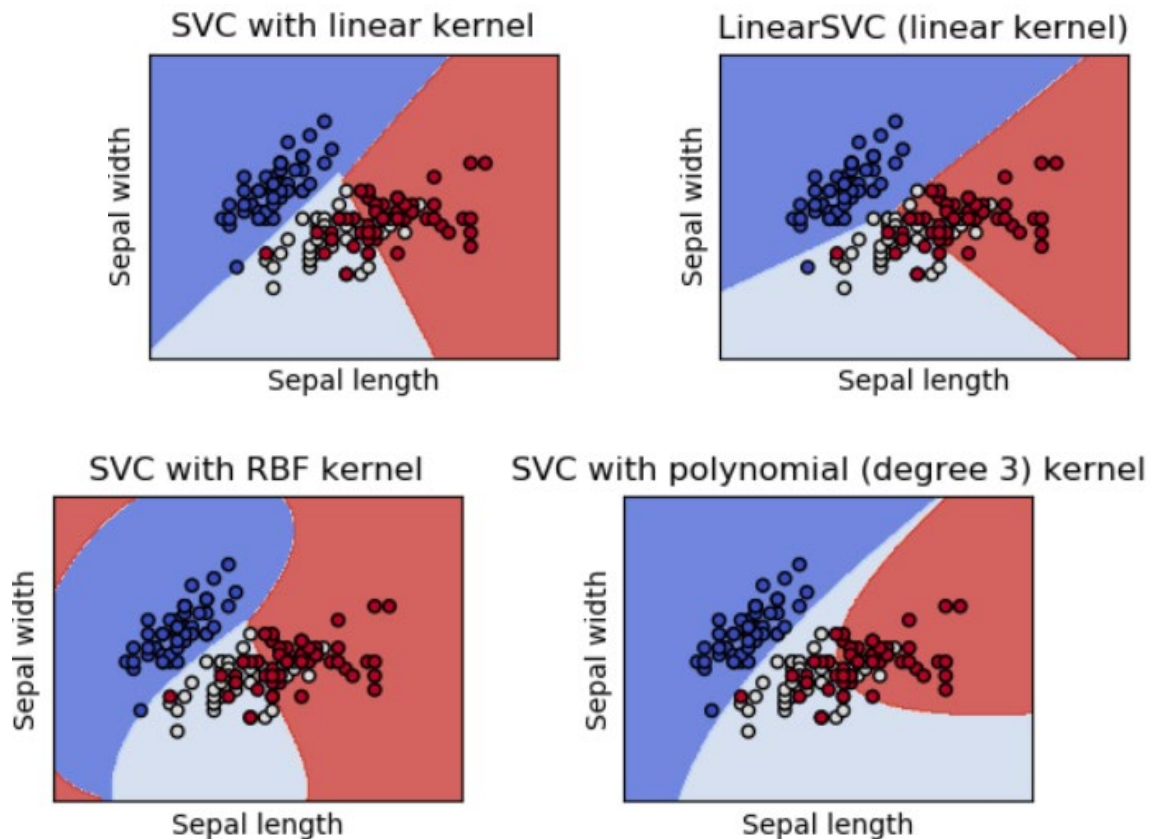


**Low Gamma**  
Far away points are also considered.



# 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://scikit-learn.org/stable/auto_examples/svm/plot_iris_svc.html)  
[https://en.wikipedia.org/wiki/Iris\\_flower\\_data\\_set](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

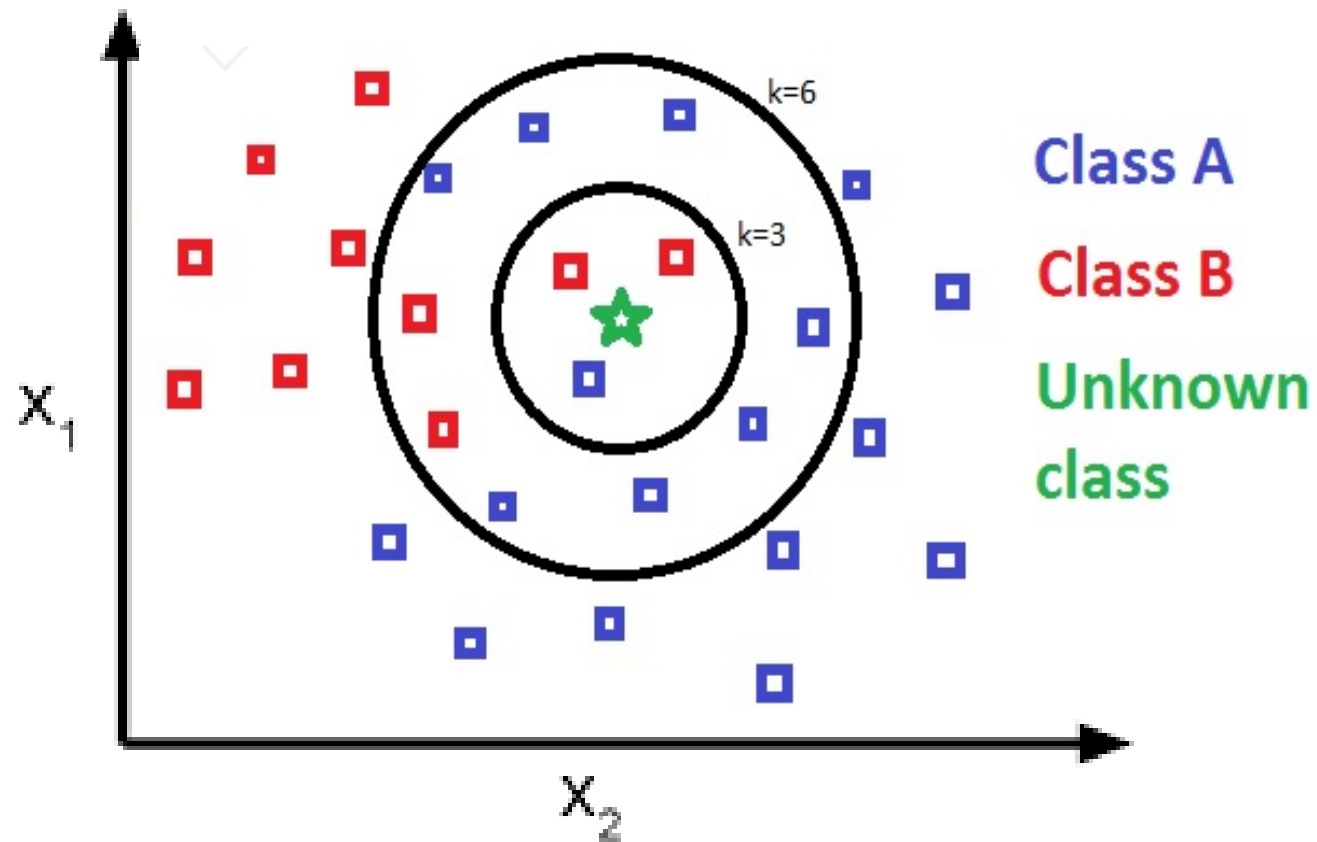
# K-Nearest Neighbour (KNN)

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



# K-Nearest Neighbour (KNN)



# 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:  $TP / \text{Cancer Diagnoses}$

	Diagnoses (Predicted)	
	No Cancer	Cancer
True state (Actual)	No Cancer	FP
	Cancer	TP

Recall:  $TP / \text{Cancer True States}$

# Evaluation Metrics

## • Precision & Recall

what are the “error” cells:

- *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

		Diagnoses (Predicted)	
		No Cancer	Cancer
True state (Actual)	No Cancer	TN	FP
	Cancer	FN	TP

Recall: TP/Cancer True States


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

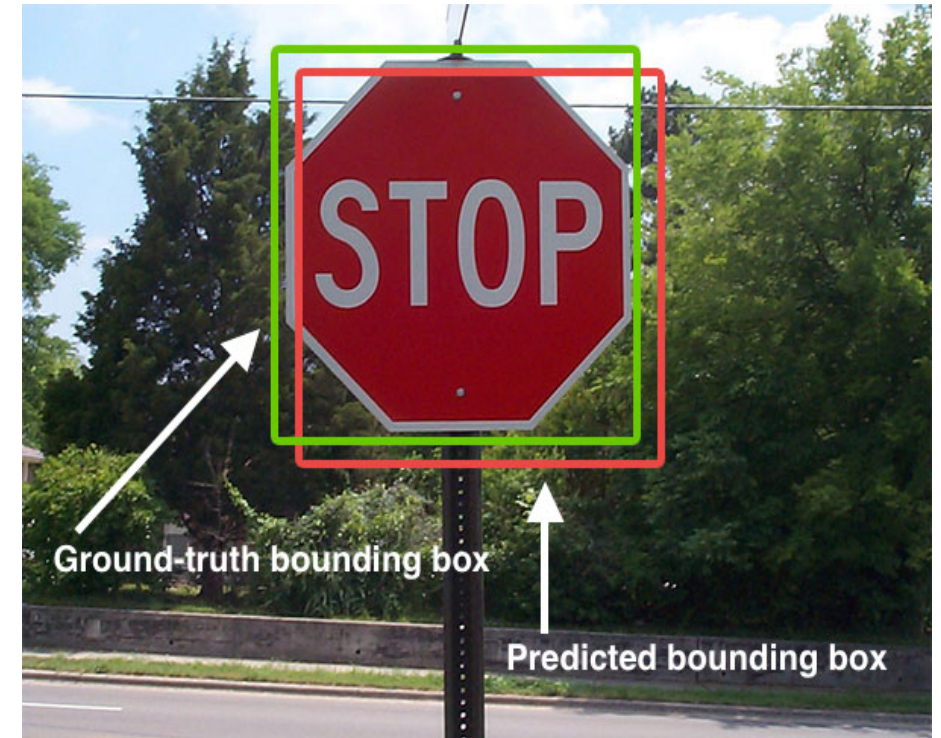
$$\text{Recall} = (TP) / (TP + FN)$$

# 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

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




# 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*** arranged 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

Blue	170	170	55	170	170			
Green	170	170	55	170	170	70		
Red	170	170	55	170	170	70	70	
	170	55	170	55	170	70	55	
	170	55	170	55	170	55	55	
	55	140	140	140	55	55		
	55	170	170	170	55			

**5 X 5 X 3 colour image (24 bit)**



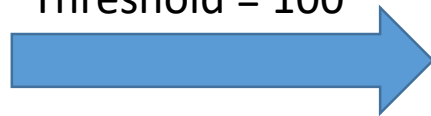
# Image Thresholding

- Easiest method for image segmentation!
- Converts gray-scale image into a binary image  
If  $f(x,y) > \text{Threshold}$ , then  $f(x,y) = 0$  else  $f(x,y) = 255$

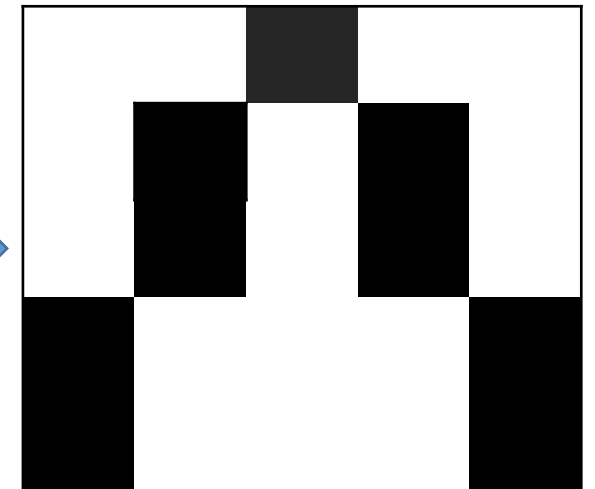
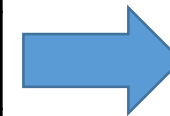
*Binary Image (8-bit) has only two possible values of pixel intensity ( 0 and 1, or B & W)*

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 Thresholding methods

- **Histogram shape:**

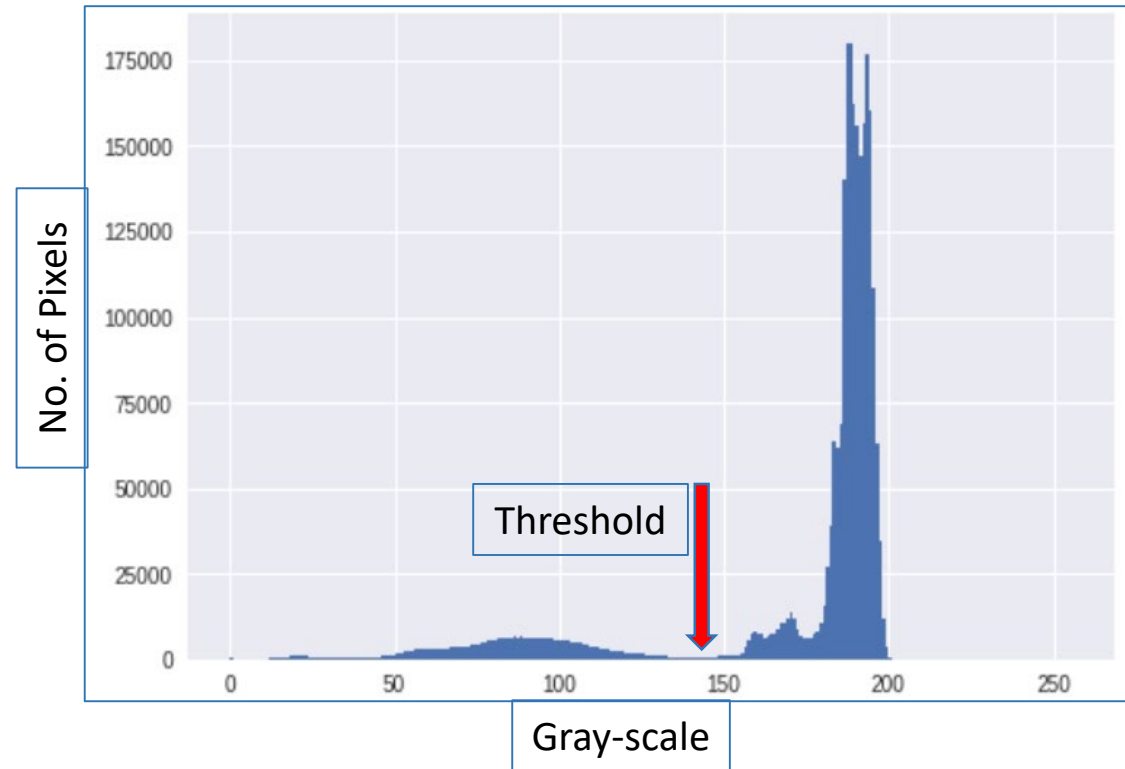
Peaks, valleys and curvature of the histogram are analysed.

- **Clustering based:**

The <sup>1</sup>Otsu method, very good for bimodal distribution

- **Adaptive thresholding:**

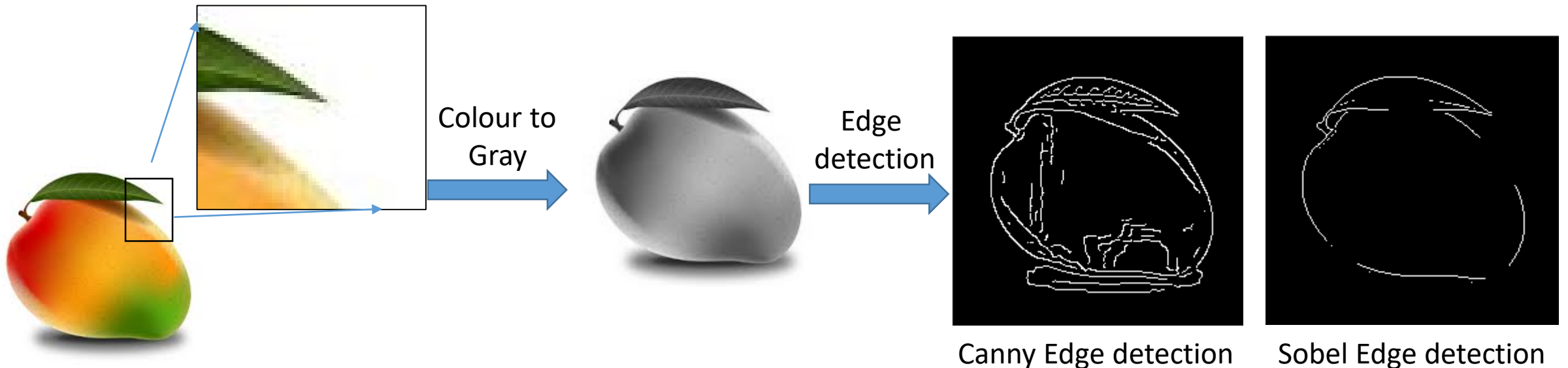
Instead of a single threshold, have thresholds for different regions in the image



# 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 find 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

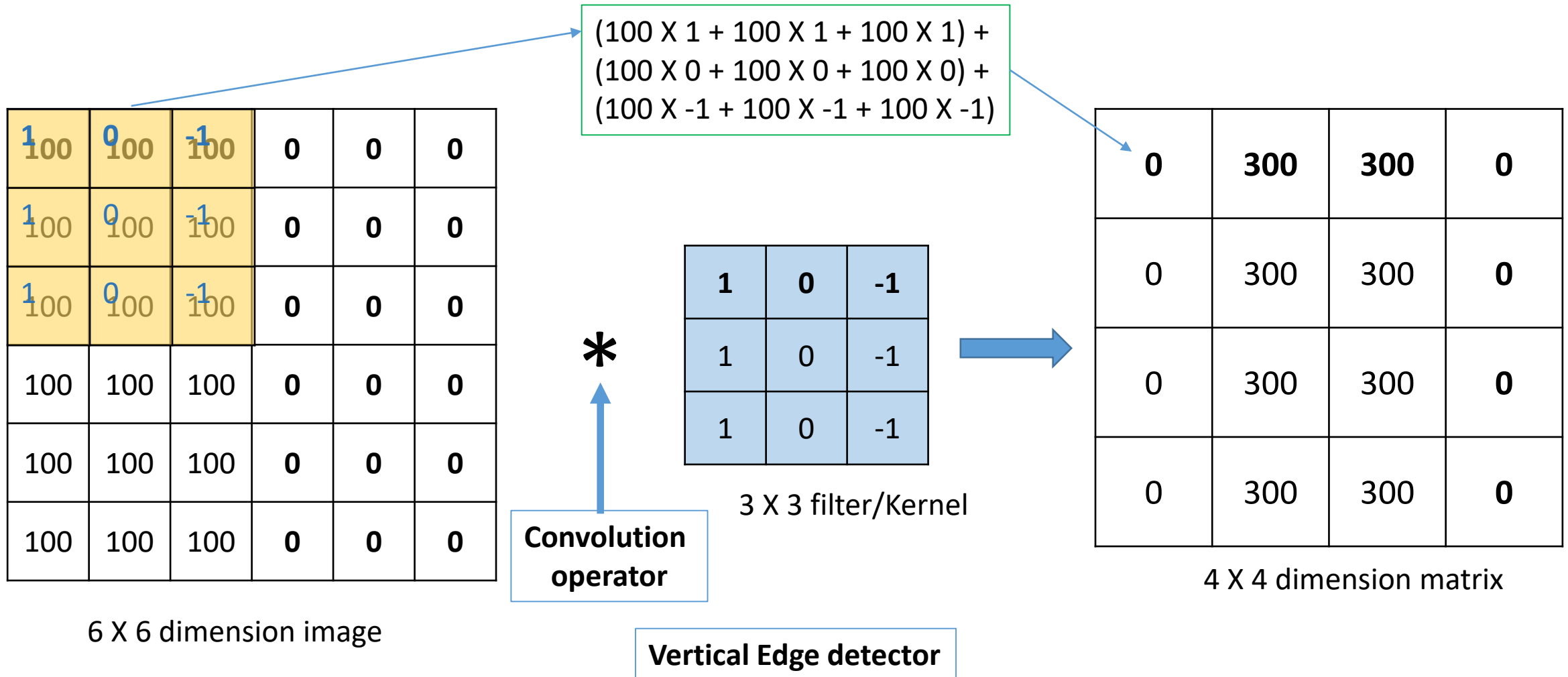
3 X 3 filter/Kernel



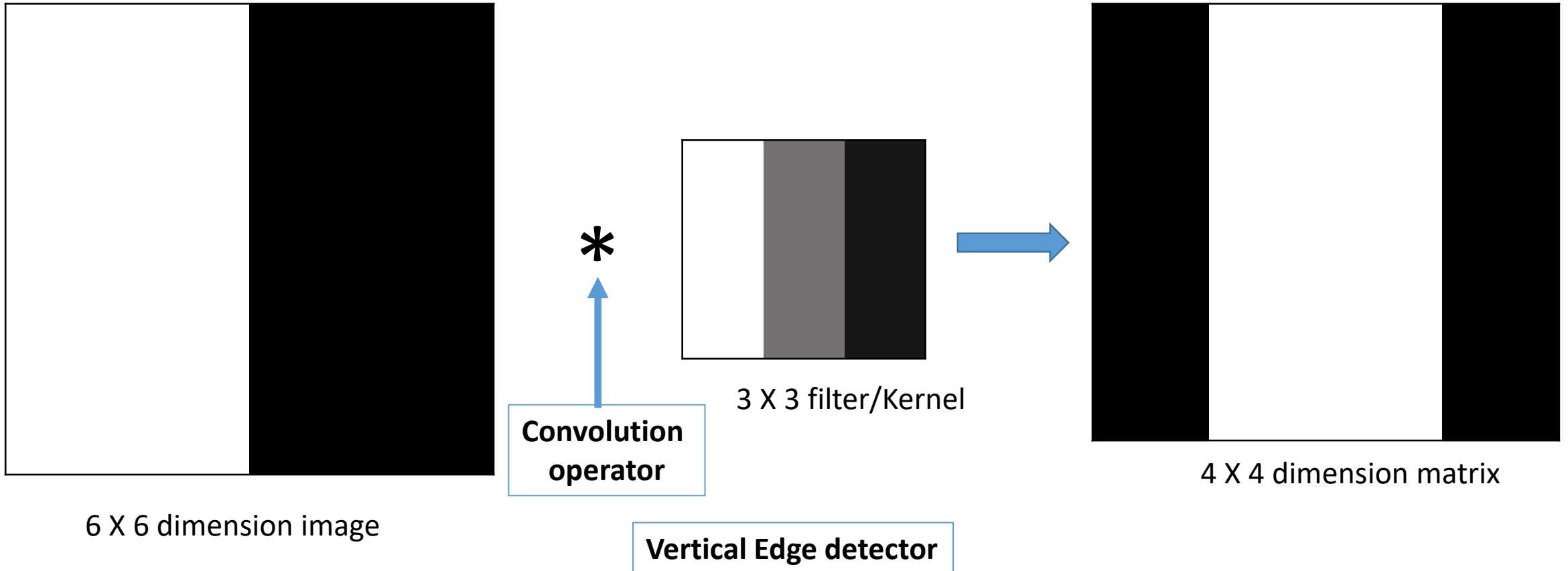
0	300	300	0
0	300	300	0
0	300	300	0
0	300	300	0

4 X 4 dimension matrix

# How to find Edges in images? (Convolution)



# How to find Edges in images? (Convolution)



# How to find 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



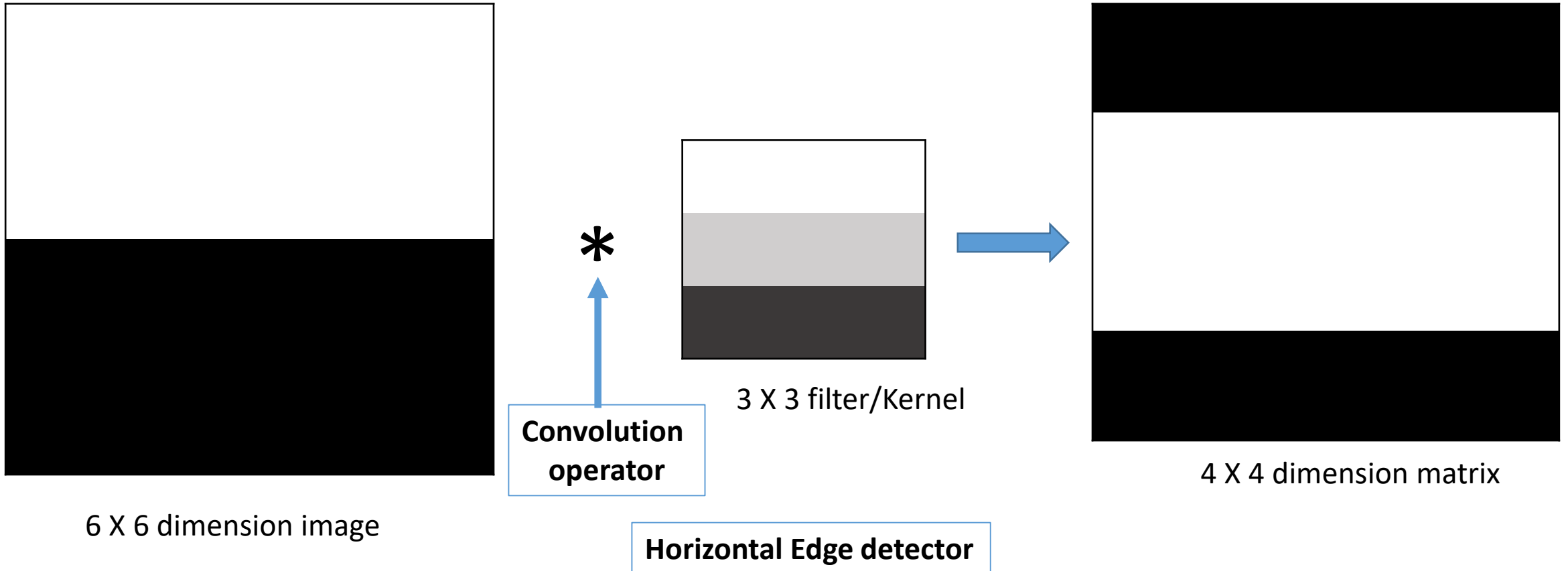
0	0	0	0
300	300	300	300
300	300	300	300
0	0	0	0

4 X 4 dimension matrix

Horizontal Edge detector



# How to find Edges in images? (Convolution)



# Sobel edge detection

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

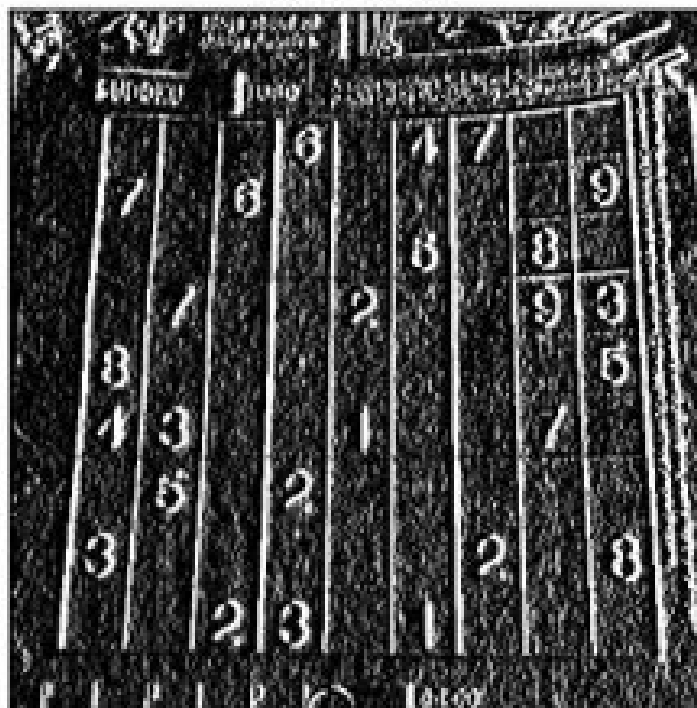
Sobel Filters

# Sobel edge detection

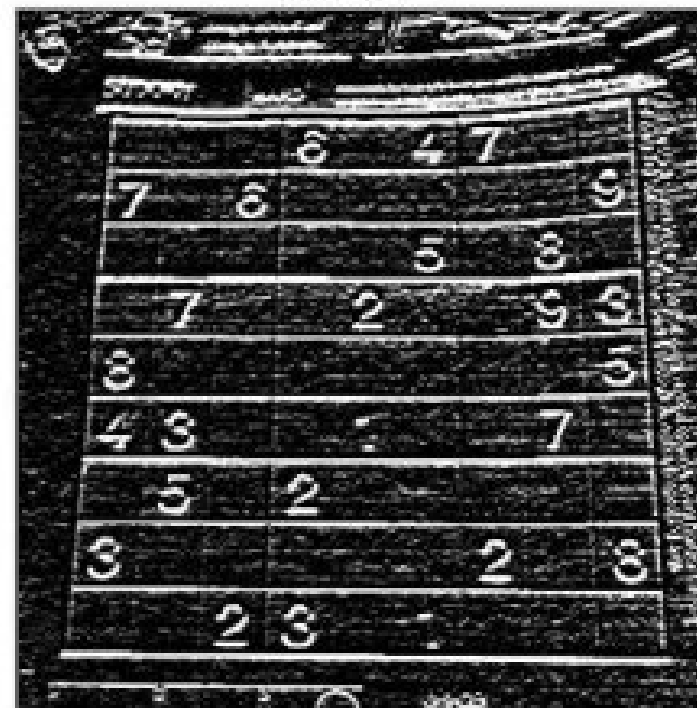
Original



Sobel X



Sobel Y



# 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