

# PFA

28 MAI 2019

« A study research about the algorithms allowing  
us to detect and identify sign language »



A



G



N



T



U



V



W



X



Y



F



M



S

**Realised by :**

MOUKHLIS Yahiya  
NAIT HAMMOU Mehdi

**Supervised By :**

M.Berrahou Issam


# Acknowledgement

We would like to express our special thanks of gratitude to our teacher Mr.Berrahou Issam who gave us the opportunity to work on this project about studying the algorithms that allows us to detect and identify sign language, which also helped us in doing a lot of research that brought us a lot of knowledge.

Secondly, we would also like to thank Ms.Tamou Nasser for judging our project.

Lastly, we would like to thank anybody who helped us, directly or indirectly, to realize this project.

# Abstract

This document is representing our final project which is about identifying sign language using machine learning.

Our goal is to study the difference between the various classification algorithms coupled with different descriptors.

This project is about studying the accuracy of all these algorithms, and how it changes with different descriptors. In order to deduct the optimal combination for such a use.

# Abbreviations list

Abbreviation	Meaning
LBP	Local Binary Pattern
HOG	Histogram of Oriented Gradients
KNN	K Nearest Neighbor
SVM	Support Vector Machine
CNN	Convolutional Neural Network
RAM	Random Access Memory
GB	Giga Byte
CPU	Central Processing Unit
GPU	Graphics Processing Unit
AI	Artificial Intelligence

# Figures List:

Figure number and name	Page
Figure 1: The relation between AI, ML and CV.	7
Figure 2: Illustration of the Hog algorithm mechanism	13
Figure 3: Input and output of HOG algorithm	13
Figure 4: LBP thresholding example	14
Figure 5: input and output of LBP descriptor	15
Figure 6: Illustration of the invariance of the hu moments	17
Figure 7: A representation of classification	18
Figure 8: KNN scale comparison	19
Figure 9: KNN explication 1	20
Figure 10: KNN explication 2	20
Figure 11: The effect of value “K” on the class boundaries.	21
Figure 12: Curve of training error rate with varying value of K	22
Figure 13: Curve of validation error rate with varying value of K	22
Figure 14: SVM concept representation.	23
Figure 15: Hyperplane identification scenario 1	24
Figure 16: Hyperplane identification scenario 2(1)	25
Figure 17: Hyperplane identification scenario 2(2)	25
Figure 18: Hyperplane identification scenario 3	26

Figure 19: Hyperplane identification scenario 4(1)	27
Figure 20: Hyperplane identification scenario 4(2)	27
Figure 21: Hyperplane identification scenario 5(1)	28
Figure 22: Hyperplane identification scenario 5(2)	28
Figure 23: Hyperplane identification scenario 6	29
Figure 24: Array of RGB Matrix	30
Figure 25: Neural network with many convolutional layers	31
Figure 26: Image matrix multiplies kernel or filter matrix	31
Figure 27: Image matrix multiplies kernel or filter matrix	32
Figure 28: 3 x 3 Output matrix	32
Figure 29: Some common filters	33
Figure 30: Stride of 2 pixels	33
Figure 31: ReLU operation	34
Figure 32: Max Pooling	35
Figure 33: After pooling layer, flattened as FC layer	35
Figure 34: Complete CNN architecture	36
Figure 35: Python Logo	38
Figure 36: Colab interface	39
Figure 37: Scikit Learn logo	40
Figure 38: matplotlib logo	41
Figure 39: OpenCV logo	41
Figure 40: train_test_split() use	42
Figure 41: Results of the implementation of the KNN algorithm	43
Figure 42: accuracy rate of different SVM/Descriptors combinations	43
Figure 43: Confusion matrix layout	44
Figure 44: Confusion matrix for KNN / HOG with k=7	45
Figure 45: Confusion matrix for SVM /Hu Moments	46

Figure 46: Confusion matrix for KNN / Hu Moments with k=1

47



# Contents table

Abbreviation List .....	3
Figures List .....	4
General introduction .....	7
Chapter I: The general context of the project .....	9
1 – Introduction .....	10
2 – Project presentation: .....	11
2 - 1 Project description .....	11
2 - 2 The problematic and the objective .....	11
Chapter II: Descriptors and Classifiers .....	12
1 – Feature Extractors .....	13
1 - 2 Histogram of Oriented Gradients (HOG) .....	13
1 - 2 Local Binary Patterns (LBP) .....	15
1– 3 HU Moments .....	16
2 – Classifiers .....	19
2 – 1 KNN algorithm .....	20
2 – 2 SVM algorithm .....	24
2 – 3 CNN algorithms .....	31
Chapter III: Realization of the project .....	38
1 – Workspace .....	39
1 - 1 Python .....	39
2 - 2 Colab .....	39
2 - 3 Libraries used .....	40
A – Scikit Learn .....	40
B – matplotlib .....	42
C – Open CV .....	42
2 – Analyzing the results .....	44
2 – 1 Dataset Preparation .....	44
2 – 2 Implementation of the classifiers with the different descriptors .....	45
A – Results and interpretations .....	45
B – Cross Validation .....	45
3 – Conclusion .....	50
Bibliography .....	51

# General introduction

In order to detect the sign that is going to be in the picture, the computer must detect it first, it must “see” it. The computer must have two human traits to do that, which is vision and the ability to differentiate between things in those images. Luckily, researchers in computer science developed what is called Computer Vision.

Computer Vision, often abbreviated as CV, is defined as a field of study that seeks to develop techniques to help computers “see” and understand the content of digital images such as photographs and videos.

The problem of computer vision appears simple because it is trivially solved by people, even very young children. Nevertheless, it largely remains an unsolved problem based both on the limited understanding of biological vision and because of the complexity of vision perception in a dynamic and nearly infinitely varying physical world.

The internet is comprised of text and images. It is relatively straightforward to index and search text, but in order to index and search images, algorithms need to know what the images contain. For the longest time, the content of images and video has remained opaque, best described using the meta descriptions provided by the person that uploaded them.

To get the most out of image data, we need computers to “see” an image and understand the content.

This is a trivial problem for a human, even young children.

- A person can describe the content of a photograph they have seen once.
- A person can summarize a video that they have only seen once.
- A person can recognize a face that they have only seen once before.

We require at least the same capabilities from computers in order to unlock our images and videos.

Computer vision is a multidisciplinary field that could broadly be called a subfield of artificial intelligence and machine learning, which may involve the use of specialized methods and make use of general learning algorithms.

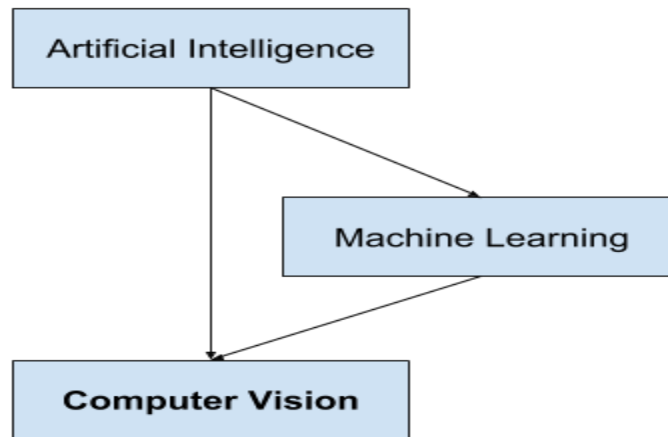


Figure 1: The relation between AI, ML and CV.

As a multidisciplinary area of study, it can look messy, with techniques borrowed and reused from a range of disparate engineering and computer science fields.

One particular problem in vision may be easily addressed with a hand-crafted statistical method (descriptors), whereas another may require a large and complex ensemble of generalized machine learning algorithms (Classifiers).

# Chapter I:

**The general context of the project**

## **1 – Introduction:**

Being still in our first, our knowledge about the subject, machine learning and computer vision was very limited. This pushed us into doing a lot of researches about all the subjects mentioned above, in order to know how to structure our work and which aspects of machine learning and computer vision to focus on.

In doing so We found 2 important notions that will be the heart of this project: Descriptors and Classifiers.

Visual descriptors or image descriptors are descriptions of the visual features of the contents in images, videos, or algorithms or applications that produce such descriptions. They describe elementary characteristics such as the shape, the color, the texture or the motion, among others.

Classification is the process of predicting the class of given data points. These data points are given to the Classifier by the use of a Descriptor. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function ( $f$ ) from input variables ( $X$ ) to discrete output variables ( $y$ ).

## **2 – Project presentation:**

### **2 - 1 Project description:**

People can debate all they want about what's the hardest language to learn, but it is for a fact the hardest one will always be sign language. As sign language isn't a go to language for people wanting to discover a new one, in fact it is hardly considered. So, it is natural that, compared to other languages, documentation and other way to learn it are few and hard to be interpreted alone.

Waiting for more documentations and ways of studying it will take a long time. So, meanwhile, we can use the next best thing, which is using a computer to translate that language. This is of course possible using computer vision and deep machine learning.

The study of such a possibility is what's our project about.

### **2 -2 The problematic and the objective:**

Computers alone cannot interpret an image and define what's in it. As images are for computers only matrixes of different values that differs with from a pixel to another. These matrixes contain multiple information, some which are important and will make the computer able to differentiate between images and others that are not. Examples for said information are: texture, color variation, edges etc....

We need then to extract these useful data. We'll do that using descriptors. They describe elementary characteristics such as the shape, the color, the texture or the motion, among others. The ones that we'll use are the local binary pattern "LBP", the histogram of oriented gradients "HOG" and the "Hu moment".

After extracting these features, we'll need classify them. We'll need the computer to learn how to classify each picture depending on its features. This is made through the classification method. Classifiers will be trained to recognize the class of a picture depending on its feature using a dataset divided into two parts: one for training, and the other one for testing the results of the training, to know the accuracy of detection.

Coupling the different descriptors and different algorithms will provide us with various accuracies. Our goal is to find the optimal combination, and to try to explain why that choice is the optimal one, while displaying the variance of said accuracies throughout the process.

# Chapter II:

## **Descriptors and Classifiers**

## **1 – Feature Extractors:**

As mentioned before , images are a sequence of ones and zeros , often associated with different color models such as RGB (Red ,green , bleu ) where every pixel of an image is represented through a three dimension array where each value indicates the level of the RGB colors , going from 0 to 255 , and by recognizing the variations of these values we can recognize the image characteristics through algorithms that we call feature extractors.

A feature extractor is an algorithm destined to recognize and extract patterns from a big set of data, through the elimination of redundant and useless information

In this project the objective is the recognition of the hand shape to match it with the corresponding letter. Since the shape is not color variant, we chose to convert the images to grayscale since the RGB model is three times the size of the grayscale one, especially since we used a considerably sized dataset.

In the following pages we will discuss the three descriptors we used which are:

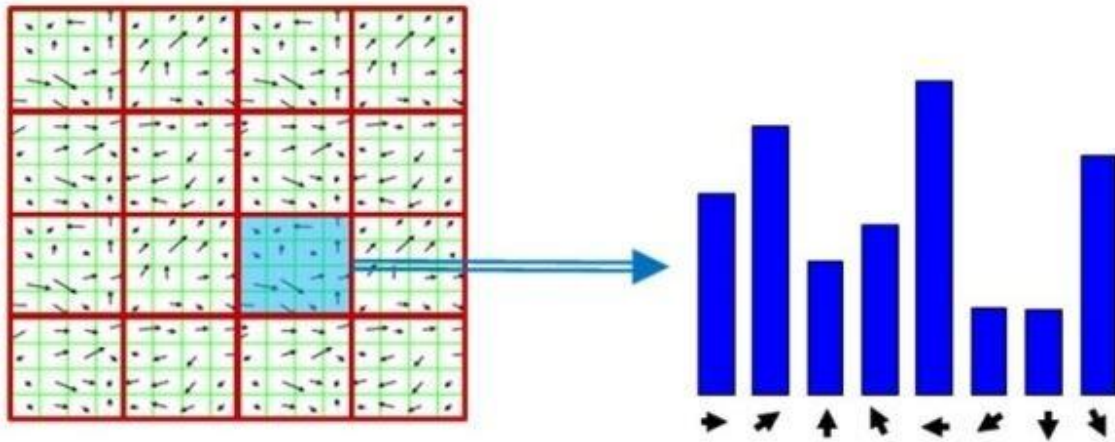
- HOG (Histogram of Oriented Gradients)
- LBP (Local Binary Patterns)
- HU Moments

### **1 -2 Histogram of Oriented Gradients (HOG)**

HOG is a form feature extractor used in computer vision and image processing applications mainly used in object detection. As it is scale invariant, and it uses the occurrences of gradient orientation in localized portions of an image in a dense grid of uniformly spaced cells.

Image gradients are simply intensity changes across pixels in an image (or a small patch of it). Given a pixel arrangement, we can find out how much the intensity values change (from lighter tones to darker shades, or the converse) along any direction. it can be done for different directions and compute a histogram. The bins of the histogram would be the different directions you picked, and the value of each bin corresponds to the number (magnitude) of intensity changes you noted in that direction. So, it essentially represents a distribution of intensity fluctuations along different orientations (directions).





(Image from <http://gilscvblog.wordpress.com/...>)

Figure 2: Illustration of the Hog algorithm mechanism

The following figures illustrate the input and the output images of the HOG algorithm, which comes with the “skimage.feature” library.

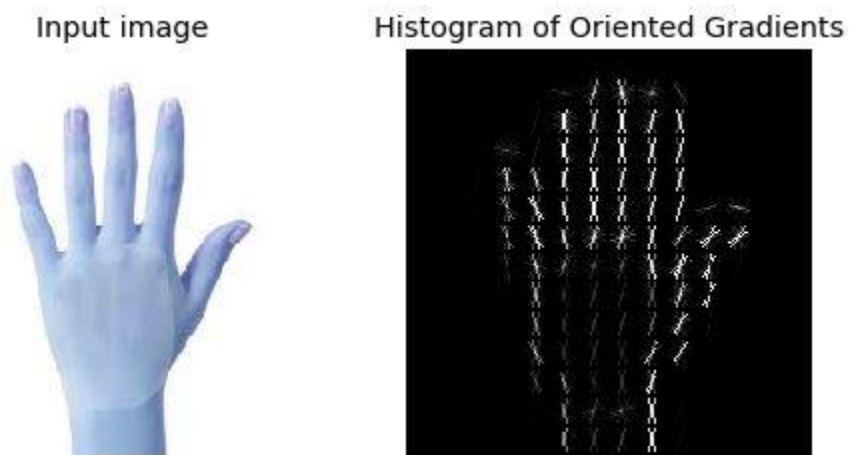


Figure 3: Input and output of HOG algorithm

## **1 - 2 Local Binary Patterns (LBP):**

Local Binary Pattern (LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number.

The first step in constructing the LBP texture descriptor is to convert the image to grayscale. For each pixel in the grayscale image, we select a neighborhood of size  $r$  surrounding the center pixel. A LBP value is then calculated for this center pixel and stored in the output 2D array with the same width and height as the input image.

Then we calculate the LBP value of each pixel and threshold it against its neighborhood of 8 pixels. If the intensity of the center pixel is greater-than-or-equal to its neighbor, then we set the value to 1; otherwise, we set it to 0. With 8 surrounding pixels, we have a total of  $2^8 = 256$  possible combinations of LBP codes. Then we calculate the LBP value using the combination obtained above starting from a pixel and working clockwise or anti-clockwise (generally we start from the top left pixel and we work clockwise).

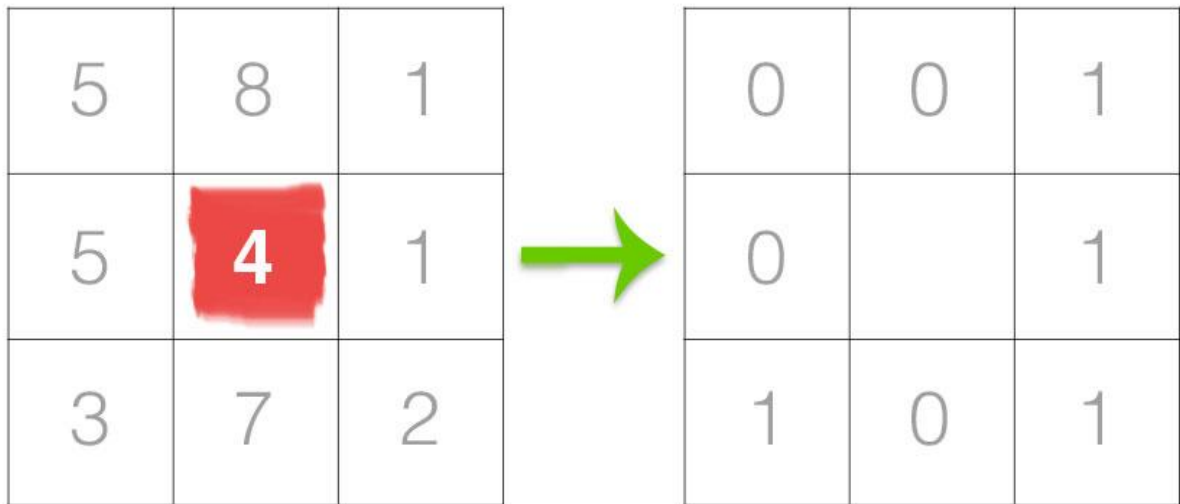


Figure 4: LBP thresholding example

In this example the sequence obtained is 010111000 which is 184 in decimal values.

There is also an extension to the standard LBP where the radius of the thresholding  $r$  and the number of point  $p$  are variables, thus this variation can solve scaling details as the standard LBP is sensitive to this variation

The following figure presents the input and the output images of the LBP descriptor which is also available in the “skimage.feature” library:

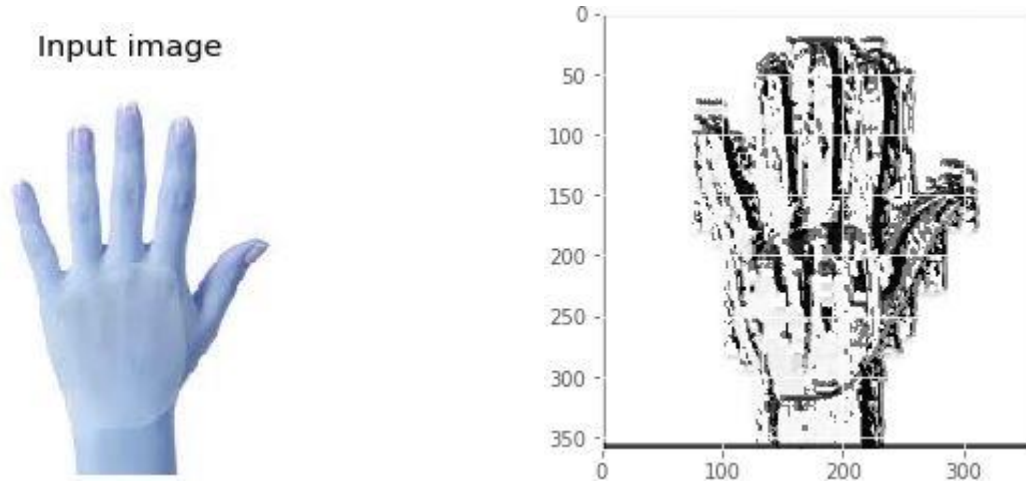


Figure 5: input and output of LBP descriptor

### **1– 3 HU Moments:**

In image processing, computer vision and related fields, an image moment is a certain particular weighted average (moment) of the image pixels' intensities, or a function of such moments, usually chosen to have some attractive property or interpretation.

From image moments we can distinguish two types of moments, which are raw moments and central moments given by the following formulas:

**Raw moments:**

$$M_{ij} = \sum_x \sum_y x^i y^j I(x, y)$$

where  $i$  and  $j$  are integers (e.g. 0, 1, 2 ....).

Note the above moments depend on the intensity of pixels and their location in the image. So intuitively these moments are capturing some notion of shape.

### Central moments:

Central moments are very similar to the raw image moments we saw earlier, except that we subtract off the centroid from the  $x$  and  $y$  in the moment formula.

$$\mu_{ij} = \sum_x \sum_y (x - \bar{x})^i (y - \bar{y})^j I(x, y)$$

Where:

$$\bar{x} = \frac{M_{10}}{M_{00}}$$

$$\bar{y} = \frac{M_{01}}{M_{00}}$$

Notice that the above central moments are **translation invariant**. In other words, no matter where the blob is in the image, if the shape is the same, the moments will be the same.

### Normalized central moments:

$$\eta_{ij} = \frac{\mu_{i,j}}{\mu_{00}^{(i+j)/2+1}}$$

**Central moments** are translations invariant, and **normalized central moments** are both translation and scale invariant.

### HU Moments:

They are a set of seven moments which are scale, translation, rotation invariant. They're given in the following formulas:

$$h_0 = \eta_{20} + \eta_{02}$$

$$h_1 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$h_2 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$h_3 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

$$h_4 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

$$h_5 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$

$$h_6 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

NB: the Hu Moment function can be obtained from the OpenCV (cv2) library.

The following image is an illustration of the invariance of the HU Moments towards the previously mentioned geometrical transformations.







id	Image	H[0]	H[1]	H[2]	H[3]	H[4]	H[5]	H[6]
K0		2.78871	6.50638	9.44249	9.84018	-19.593	-13.1205	19.6797
S0		2.67431	5.77446	9.90311	11.0016	-21.4722	-14.1102	22.0012
S1		2.67431	5.77446	9.90311	11.0016	-21.4722	-14.1102	22.0012
S2		2.65884	5.7358	9.66822	10.7427	-20.9914	-13.8694	21.3202
S3		2.66083	5.745	9.80616	10.8859	-21.2468	-13.9653	21.8214
S4		2.66083	5.745	9.80616	10.8859	-21.2468	-13.9653	-21.8214

Figure 6: Illustration of the invariance of the hu moments

## 2 – Classifiers:

Classification is a technique for determining class the dependent belongs to, based on one or more independent variables.

In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation.

*Classification is used for predicting discrete responses.*

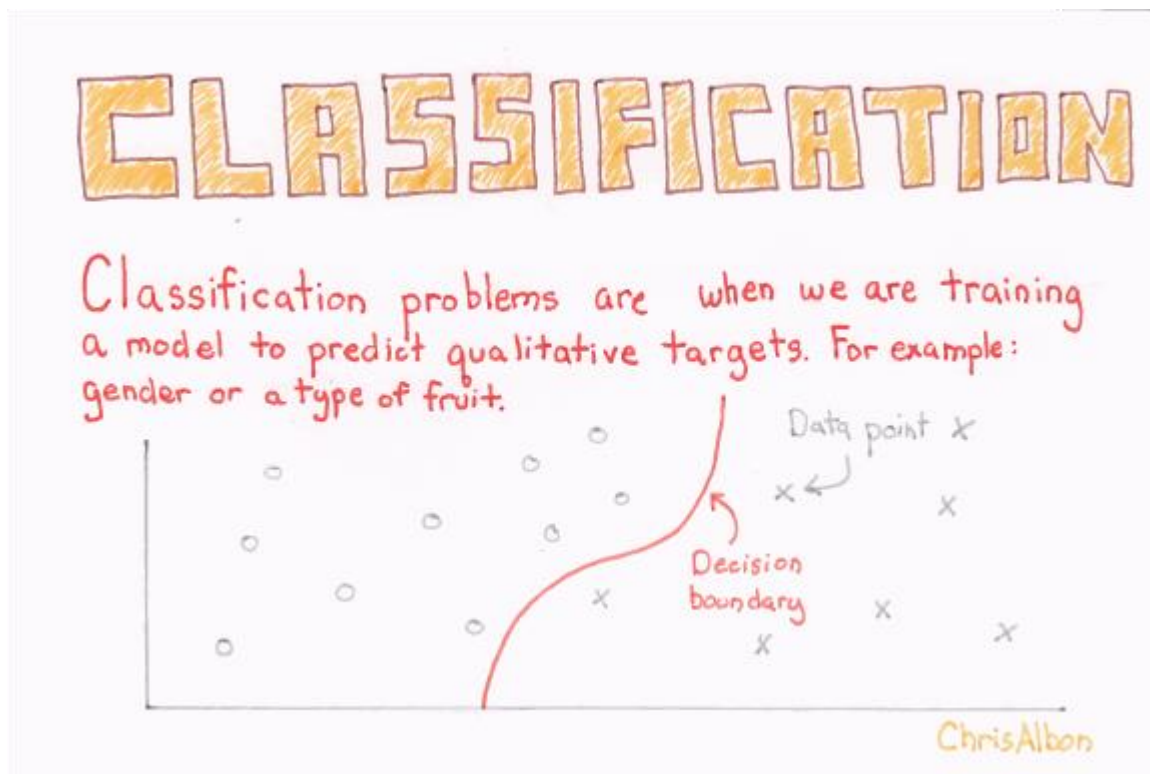


Figure 7: A representation of classification

## **2 – 1 KNN algorithm:**

### **When do we use KNN?**

KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems. To evaluate any technique, we generally look at 3 important aspects:

1. Ease to interpret output
2. Calculation time
3. Predictive Power

Let us take a few examples to place KNN in the scale:

	Logistic Regression	CART	Random Forest	KNN
1. Ease to interpret output	2	3	1	3
2. Calculation time	3	2	1	3
3. Predictive Power	2	2	3	2

Figure 8: KNN scale comparison

KNN algorithm fares across all parameters of considerations. It is commonly used for its easy of interpretation and low calculation time.

### **How does the KNN algorithm work?**

Let's take a simple case to understand this algorithm. Following is a spread of red circles (RC) and green squares (GS):

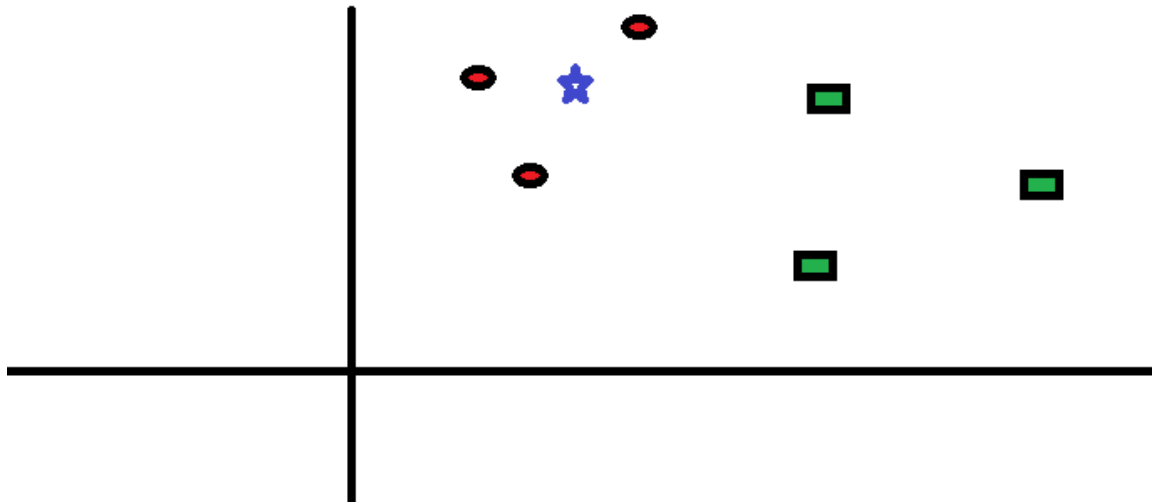


Figure 9: KNN explication 1

You intend to find out the class of the blue star (BS). BS can either be RC or GS and nothing else. The “K” in KNN algorithm is the nearest neighbors we wish to take vote from. Let’s say  $K = 3$ . Hence, we will now make a circle with BS as center just as big as to enclose only three datapoints on the plane. Refer to following diagram for more details:

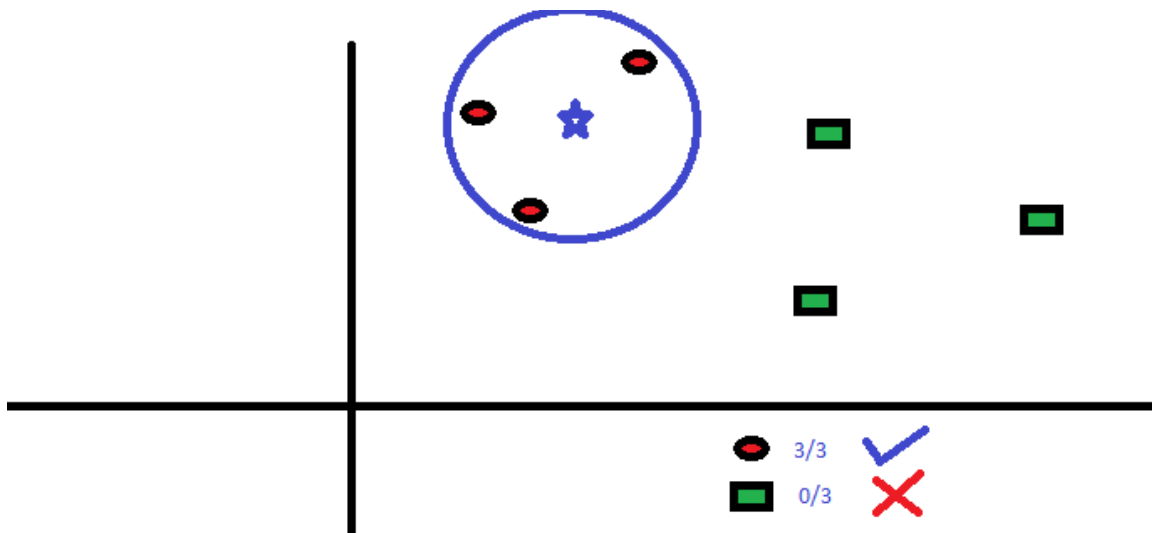


Figure 10: KNN explication 2

The three closest points to BS are all RC. Hence, with good confidence level we can say that the BS should belong to the class RC. Here, the choice became very obvious as all three votes from the closest neighbor went to RC. The choice of the parameter  $K$  is very crucial in this algorithm. Next, we will understand what are the factors to be considered to conclude the best  $K$ .



### How do we choose the factor K?

First let us try to understand what exactly does K influence in the algorithm. If we see the last example, given that all the 6 training observation remain constant, with a given K value we can make boundaries of each class. These boundaries will segregate RC from GS. The same way, let's try to see the effect of value "K" on the class boundaries. Following are the different boundaries separating the two classes with different values of K.

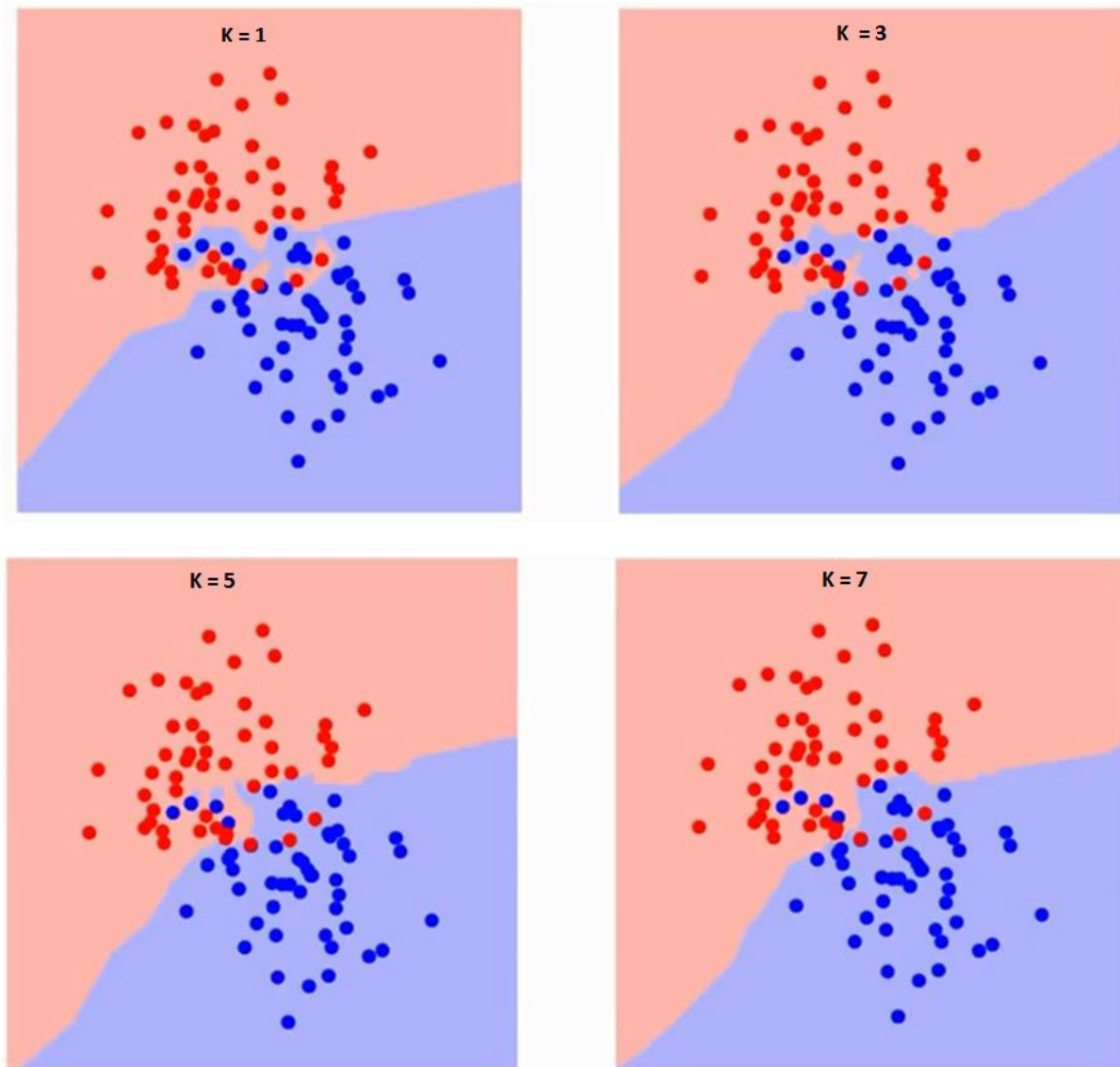
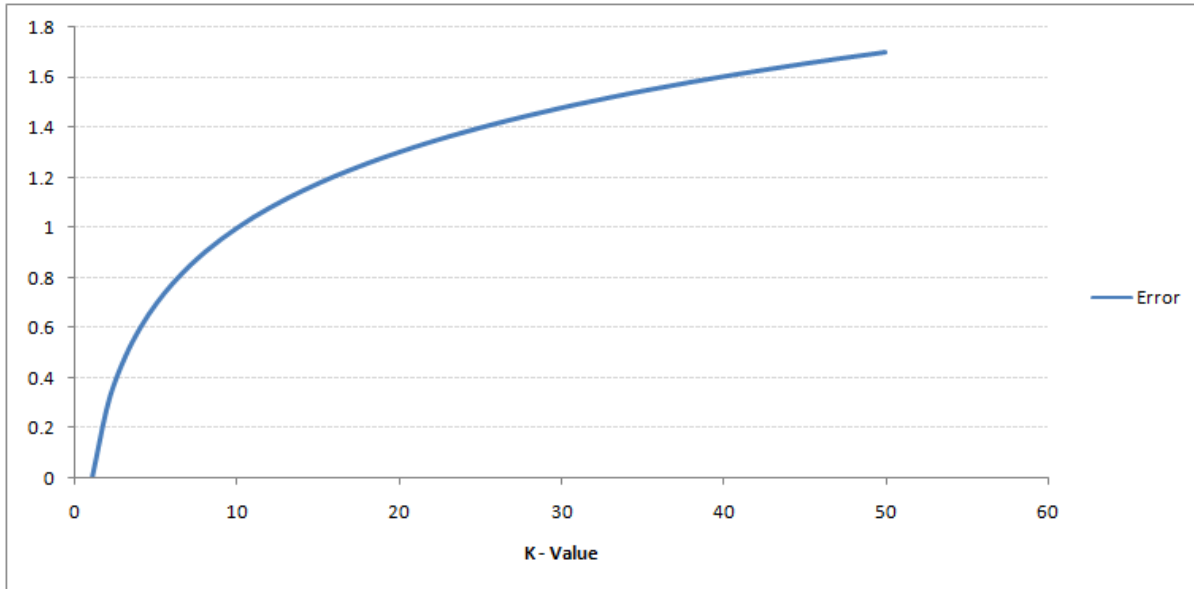


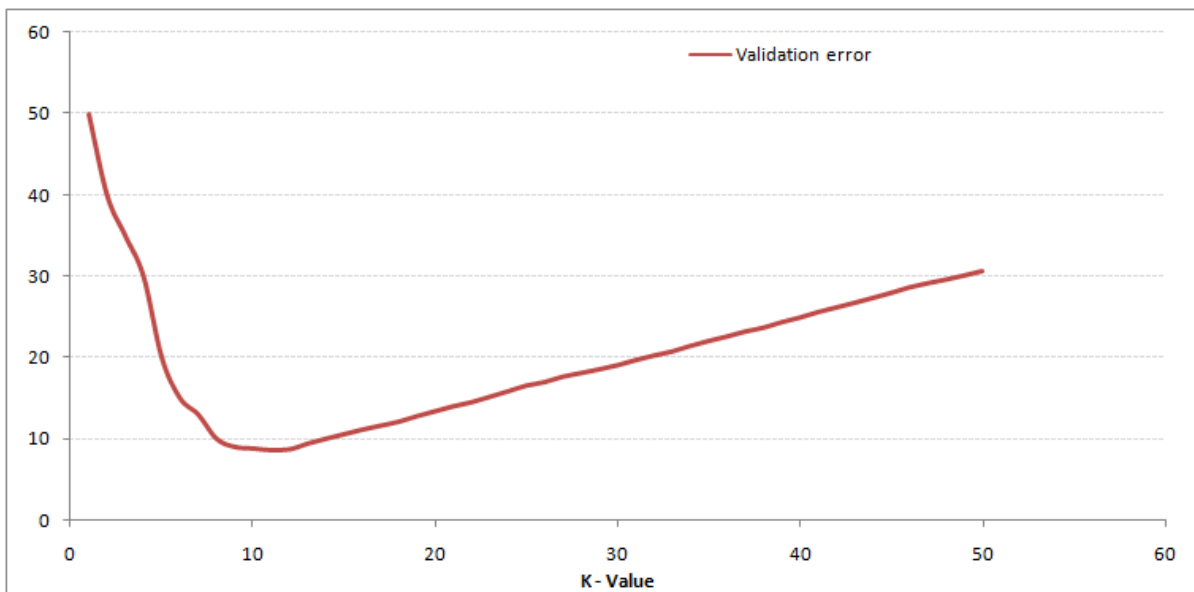
Figure 11: The effect of value "K" on the class boundaries.

If you watch carefully, you can see that the boundary becomes smoother with increasing value of K. With K increasing to infinity it finally becomes all blue or all red depending on the total majority. The training error rate and the validation error rate are two parameters we need to access on different K-value. Following is the curve for the training error rate with varying value of K:



**Figure 12:** Curve of training error rate with varying value of K

As you can see, the error rate at  $K=1$  is always zero for the training sample. This is because the closest point to any training data point is itself. Hence the prediction is always accurate with  $K=1$ . If validation error curve would have been similar, our choice of  $K$  would have been 1. Following is the validation error curve with varying value of  $K$ :



**Figure 13:** Curve of validation error rate with varying value of K

This makes the story clearer. At  $K=1$ , we were overfitting the boundaries. Hence, error rate initially decreases and reaches a minima. After the minima point, it then increases with increasing  $K$ . To get the optimal value of  $K$ , you can segregate the training and validation from the initial dataset.

Now plot the validation error curve to get the optimal value of  $K$ . This value of  $K$  should be used for all predictions.

### Breaking it Down – Pseudo Code of KNN

We can implement a KNN model by following the below steps:

1. Load the data.
2. Initialize the value of  $k$ .
3. For getting the predicted class, iterate from 1 to total number of training data points
  1. Calculate the distance between test data and each row of training data. Here we will use Euclidean distance as our distance metric since it's the most popular method. The other metrics that can be used are Chebyshev, cosine, etc.
  2. Sort the calculated distances in ascending order based on distance values.
  3. Get top  $k$  rows from the sorted array.
  4. Get the most frequent class of these rows.
  5. Return the predicted class.

## 2 – 2 SVM algorithm:

### What is Support Vector Machine?

“Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in  $n$ -dimensional space (where  $n$  is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well (look at the below snapshot).

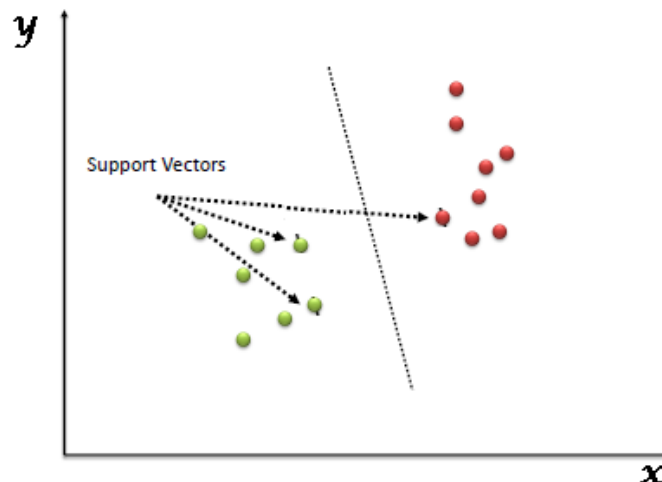


Figure 14: SVM concept representation.

Support Vectors are simply the co-ordinates of individual observation. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/ line).

You can look at support vector machines and a few examples of its working here.

### How does it work?

Above, we got accustomed to the process of segregating the two classes with a hyper-plane. Now the burning question is “How can we identify the right hyper-plane?”. Don’t worry, it’s not as hard as you think!

Let’s understand:

- **Identify the right hyper-plane (Scenario-1):**

Here, we have three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle.

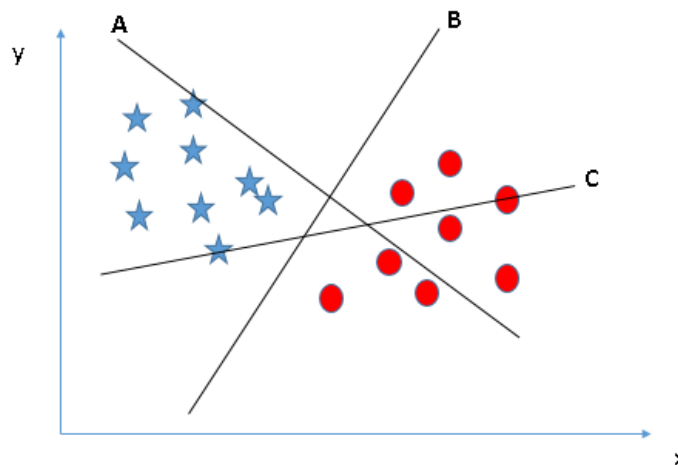


Figure 15: Hyperplane identification scenario 1

You need to remember a thumb rule to identify the right hyper-plane: “Select the hyper-plane which segregates the two classes better”. In this scenario, hyper-plane “B” has excellently performed this job.

- **Identify the right hyper-plane (Scenario-2):**

Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now, how can we identify the right hyper-plane?

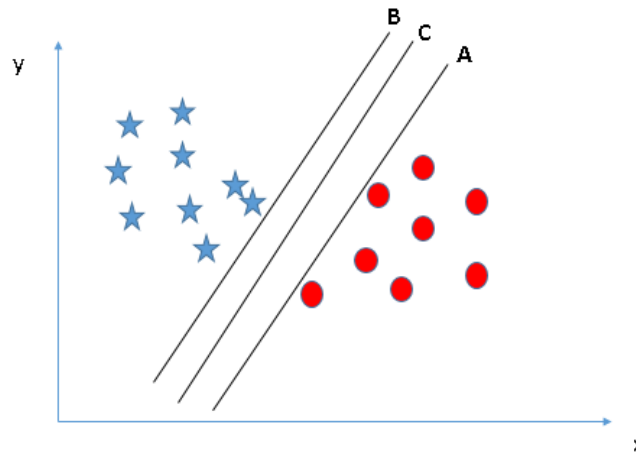


Figure 16: Hyperplane identification scenario 2(1)

Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as **Margin**. Let's look at the below snapshot:

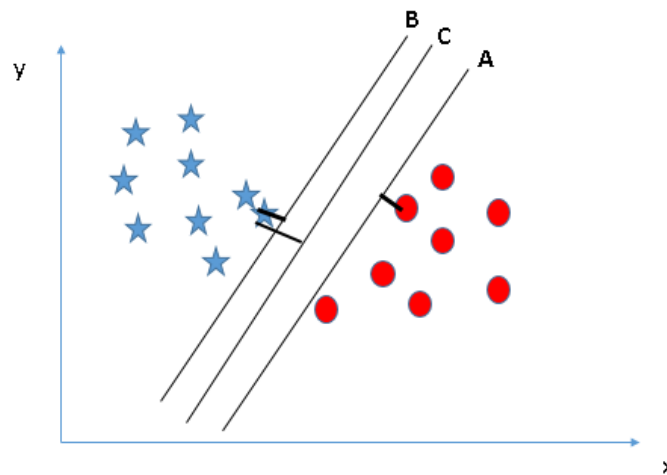


Figure 17: Hyperplane identification scenario 2(2)

Above, you can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is high chance of miss-classification.

- **Identify the right hyper-plane (Scenario-3):**

Hint: Use the rules as discussed in previous section to identify the right hyper-plane

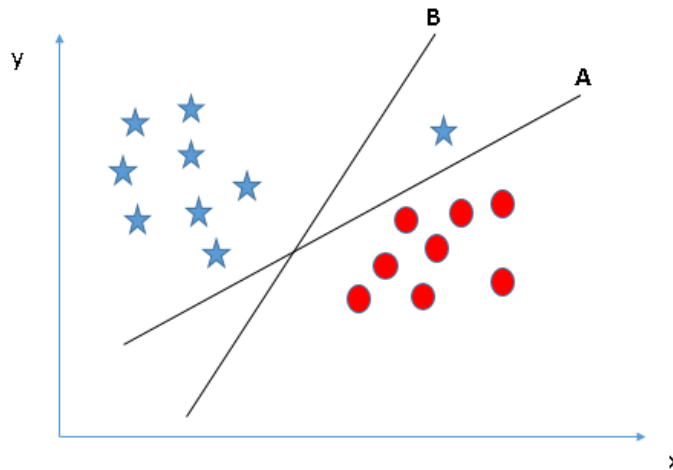


Figure 18: Hyperplane identification scenario 3

Some of you may have selected the hyper-plane B as it has higher margin compared to A. But here is the catch, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is A.

- **Can we classify two classes (Scenario-4):**

Below, I am unable to segregate the two classes using a straight line, as one of star lies in the territory of other(circle) class as an outlier.



Figure 19: Hyperplane identification scenario 4(1)

As mentioned earlier, one star at another end is like an outlier for star class. SVM has a feature to ignore outliers and find the hyper-plane that has maximum margin. Hence, we can say, SVM is robust to outliers.

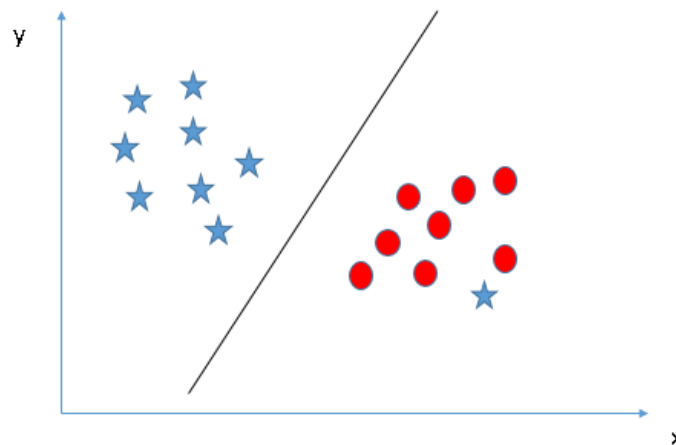


Figure 20: Hyperplane identification scenario 4(2)

- **Find the hyper-plane to segregate to classes (Scenario-5):**

In the scenario below, we can't have linear hyper-plane between the two classes, so how does SVM classify these two classes? Till now, we have only looked at the linear hyper-plane.

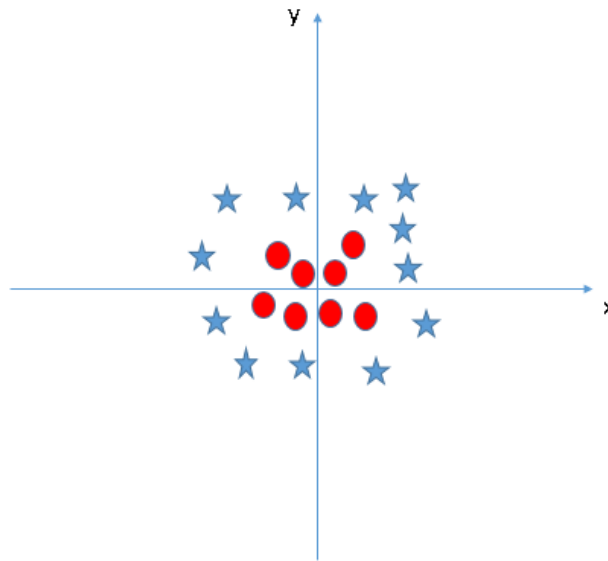


Figure 21: Hyperplane identification scenario 5(1)

SVM can solve this problem. Easily! It solves this problem by introducing additional feature. Here, we will add a new feature  $z = x^2 + y^2$ . Now, let's plot the data points on axis  $x$  and  $z$ :

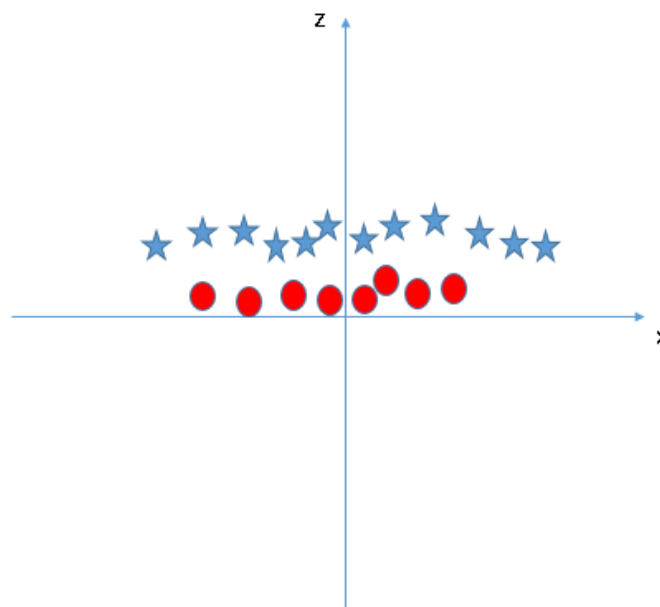


Figure 22: Hyperplane identification scenario 5(2)



In above plot, points to consider are:

- All values for  $z$  would be positive always because  $z$  is the squared sum of both  $x$  and  $y$
- In the original plot, red circles appear close to the origin of  $x$  and  $y$  axes, leading to lower value of  $z$  and star relatively away from the origin result to higher value of  $z$ .

In SVM, it is easy to have a linear hyper-plane between these two classes. But another burning question which arises is, should we need to add this feature manually to have a hyper-plane. No, SVM has a technique called the **kernel trick**. These are functions which takes low dimensional input space and transform it to a higher dimensional space i.e. it converts not separable problem to separable problem, these functions are called kernels. It is mostly useful in non-linear separation problem. Simply put, it does some extremely complex data transformations, then find out the process to separate the data based on the labels or outputs you've defined.

When we look at the hyper-plane in original input space it looks like a circle:

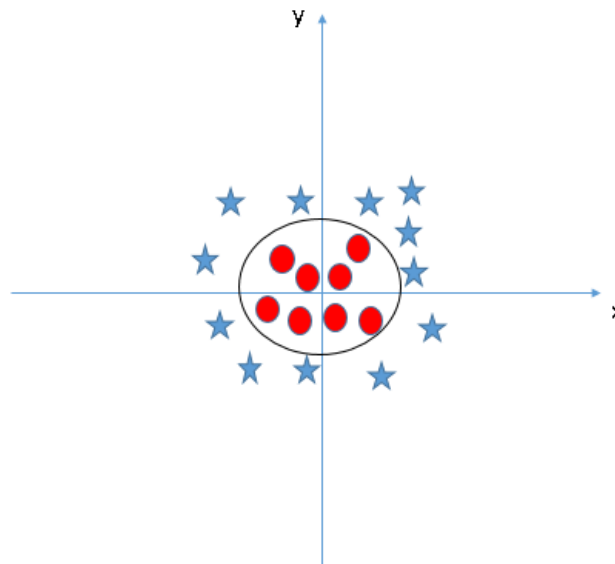


Figure 23: Hyperplane identification scenario 6

## 2 – 3 CNN algorithms:

In neural networks, Convolutional neural network (ConvNets or CNNs) is one of the main categories to do images recognition, images classifications. Objects detections, recognition faces etc., are some of the areas where CNNs are widely used.

CNN image classifications takes an input image, process it and classify it under certain categories (Eg., Dog, Cat, Tiger, Lion). Computers sees an input image as array of pixels and it depends on the image resolution. Based on the image resolution, it will see  $h \times w \times d$  (  $h$  = Height,  $w$  = Width,  $d$  = Dimension ). Eg., An image of  $6 \times 6 \times 3$  array of matrix of RGB (3 refers to RGB values) and an image of  $4 \times 4 \times 1$  array of matrix of grayscale image.

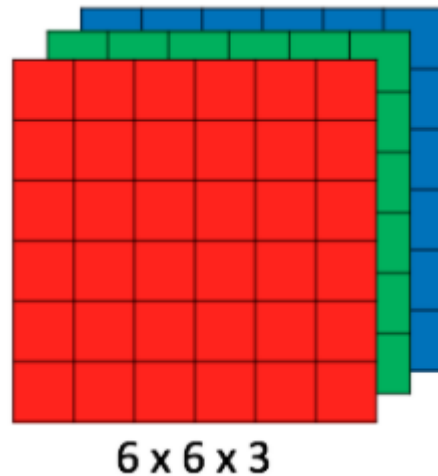


Figure 24: Array of RGB Matrix

Technically, deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernels), Pooling, fully connected layers (FC) and apply Softmax function to classify an object with probabilistic values between 0 and 1. The below figure is a complete flow of CNN to process an input image and classifies the objects based on values.

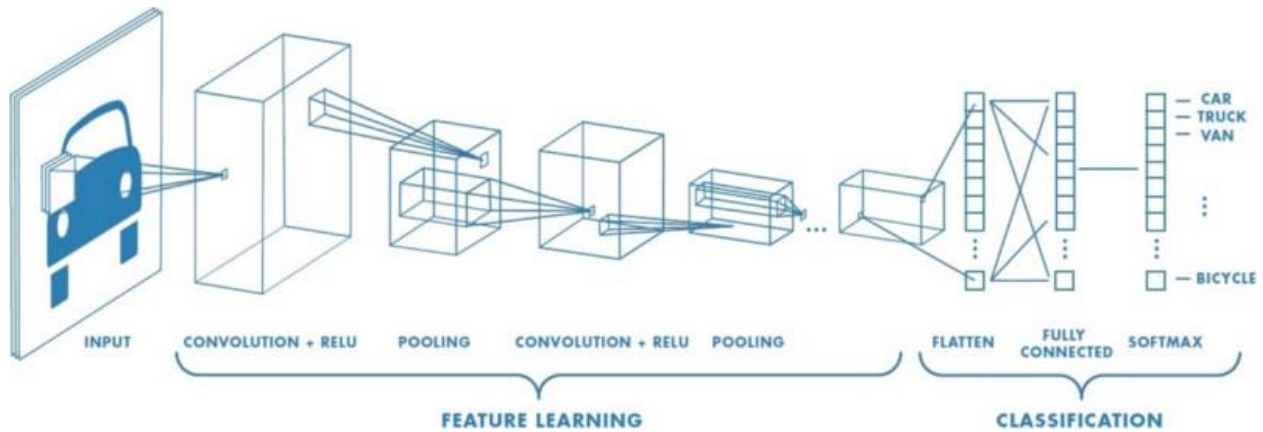


Figure 25: Neural network with many convolutional layers

### Convolution Layer

Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel

- An image matrix (volume) of dimension  **$(h \times w \times d)$**
- A filter  **$(f_h \times f_w \times d)$**
- Outputs a volume dimension  **$(h - f_h + 1) \times (w - f_w + 1) \times 1$**



Figure 26: Image matrix multiplies kernel or filter matrix

Consider a 5 x 5 whose image pixel values are 0, 1 and filter matrix 3 x 3 as shown in below

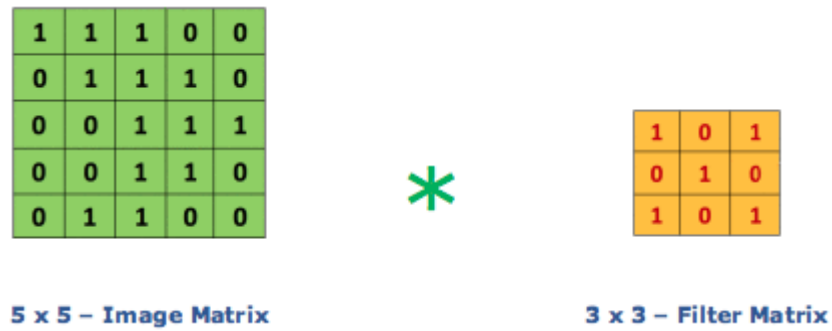


Figure 27: Image matrix multiplies kernel or filter matrix

Then the convolution of 5 x 5 image matrix multiplies with 3 x 3 filter matrix which is called “**Feature Map**” as output shown in below

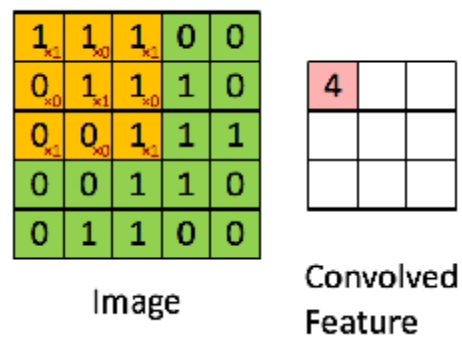


Figure 28: 3 x 3 Output matrix

Convolution of an image with different filters can perform operations such as edge detection, blur and sharpen by applying filters. The below example shows various convolution image after applying different types of filters (Kernels).



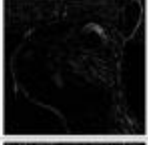




Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Figure 29: Some common filters

### Strides

Stride is the number of pixels shifts over the input matrix. When the stride is 1 then we move the filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a time and so on. The below figure shows convolution would work with a stride of 2.

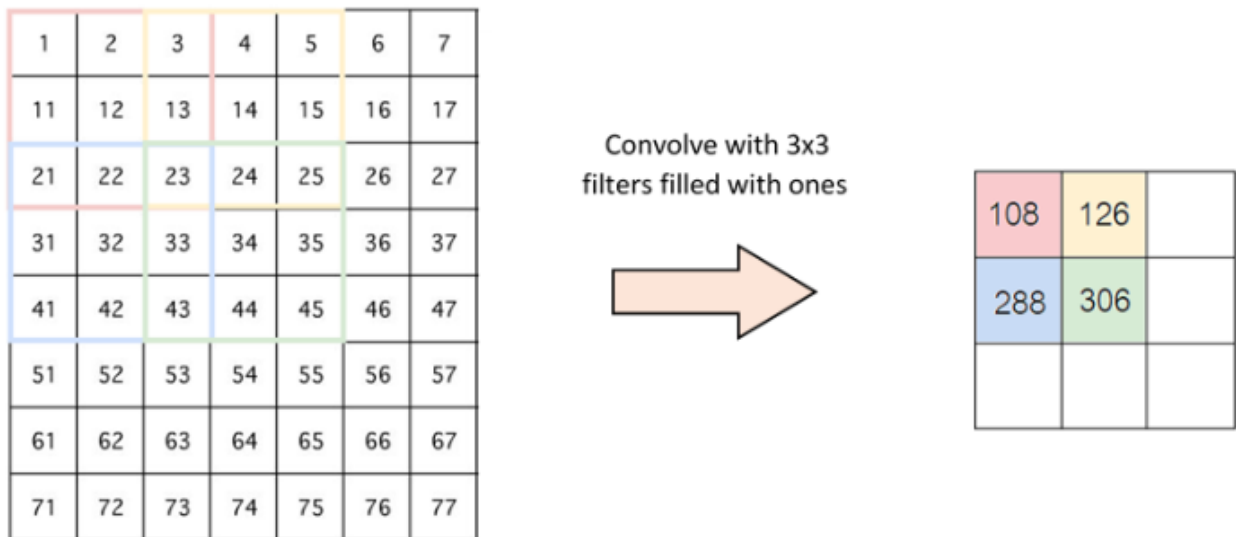


Figure 30: Stride of 2 pixels

## Padding

Sometimes filter does not fit perfectly fit the input image. We have two options:

- Pad the picture with zeros (zero-padding) so that it fits
- Drop the part of the image where the filter did not fit. This is called valid padding which keeps only valid part of the image.

## Non Linearity (ReLU)

ReLU stands for Rectified Linear Unit for a non-linear operation. The output is  $f(x) = \max(0, x)$ .

Why ReLU is important : ReLU's purpose is to introduce non-linearity in our ConvNet. Since, the real world data would want our ConvNet to learn would be non-negative linear values.

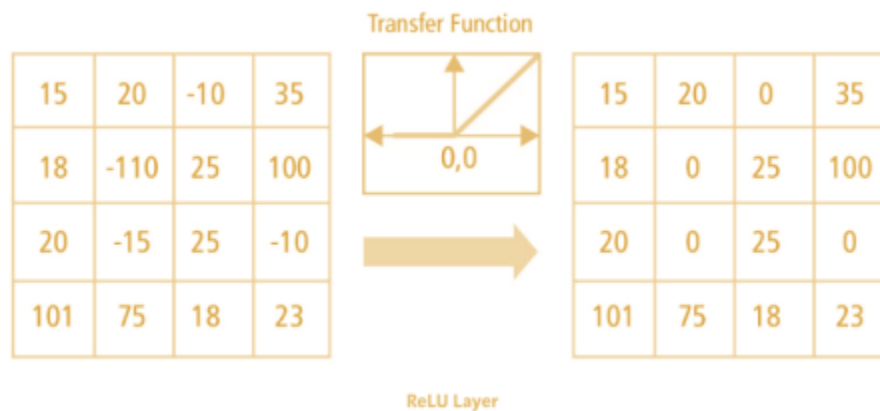


Figure 31: ReLU operation

There are other non linear functions such as tanh or sigmoid can also be used instead of ReLU. Most of the data scientists uses ReLU since performance wise ReLU is better than other two.

## Pooling Layer

Pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling also called subsampling or downsampling which reduces the dimensionality of each map but retains the important information. Spatial pooling can be of different types:

- Max Pooling
- Average Pooling
- Sum Pooling

Max pooling take the largest element from the rectified feature map. Taking the largest element could also take the average pooling. Sum of all elements in the feature map call as sum pooling.

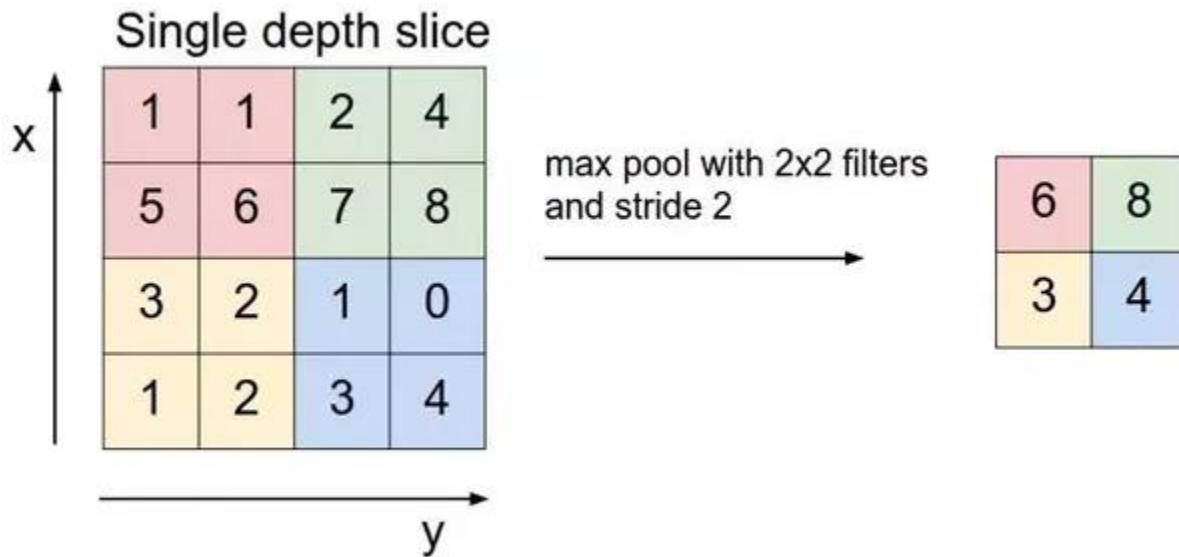


Figure 32: Max Pooling

### Fully Connected Layer

The layer we call as FC layer, we flattened our matrix into vector and feed it into a fully connected layer like neural network.

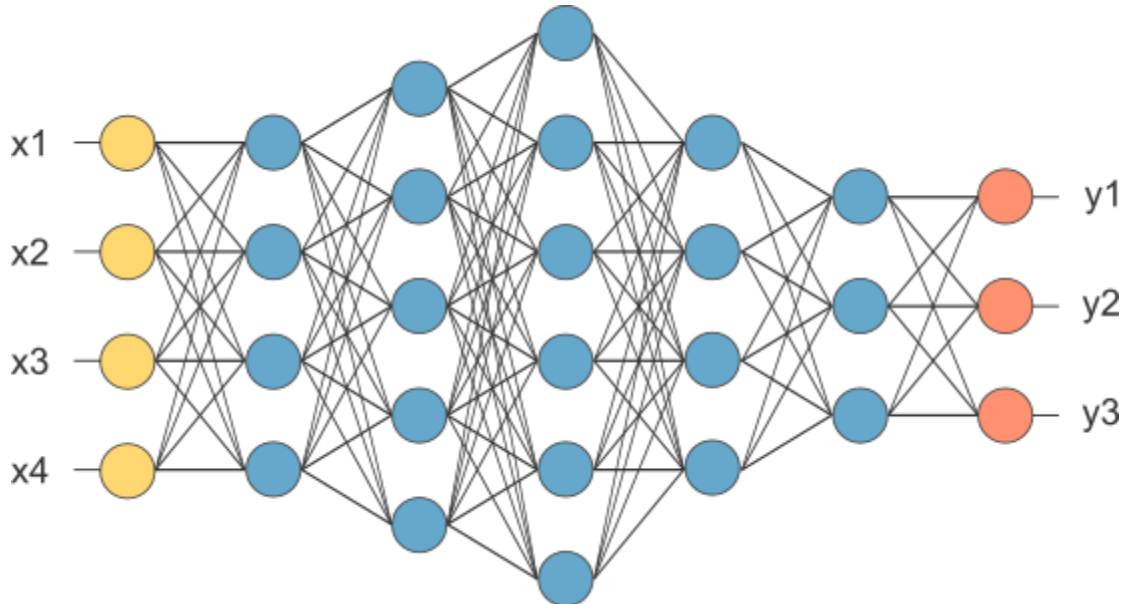


Figure 33: After pooling layer, flattened as FC layer

In the above diagram, feature map matrix will be converted as vector ( $x_1, x_2, x_3, \dots$ ). With the fully connected layers, we combined these features together to create a model. Finally, we have

an activation function such as softmax or sigmoid to classify the outputs as cat, dog, car, truck etc.,

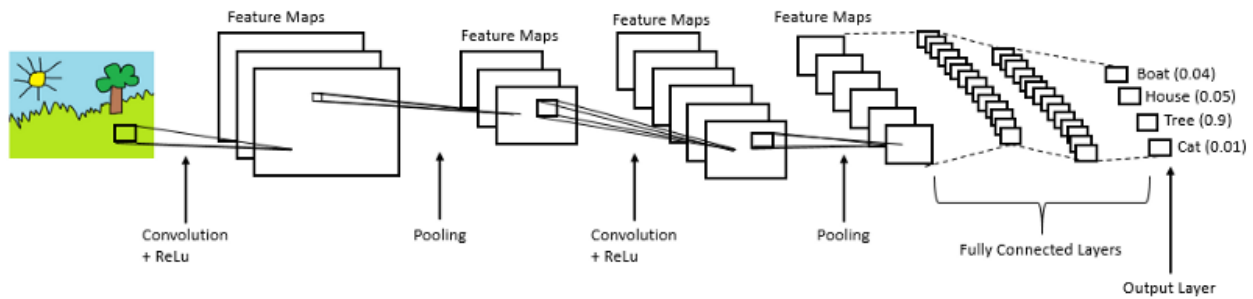


Figure 34: Complete CNN architecture

### Summary

- Provide input image into convolution layer
- Choose parameters, apply filters with strides, padding if requires. Perform convolution on the image and apply ReLU activation to the matrix.
- Perform pooling to reduce dimensionality size
- Add as many convolutional layers until satisfied
- Flatten the output and feed into a fully connected layer (FC Layer)
- Output the class using an activation function (Logistic Regression with cost functions) and classifies images.



# Chapter III:

**Realization of the project**

## **1 – Workspace:**

Here will be stated some of the programs used during our work as well as the programming language used:

### **1 - 1 Python:**

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aims to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

We used python because of how user friendly it is, but mainly because of python's strong presence in machine learning, AI, and computer vision.



Figure 35: Python logo.

### **2 - 2 Colab:**

Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud.

With Colaboratory you can write and execute code, save and share your analyses, and access powerful computing resources, all for free from your browser.

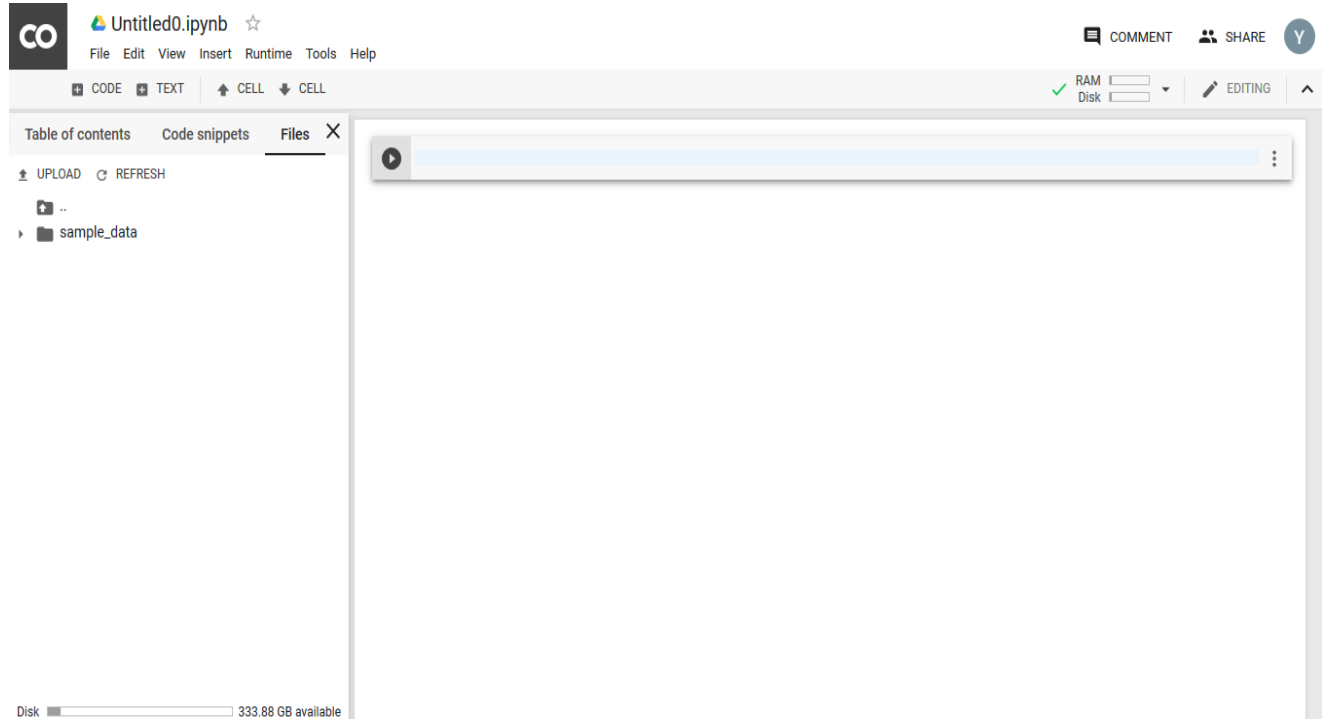


Figure 36: Colab interface

In order to do all the computing related with machine learning, computer vision and AI, we had to have a solid computer, which none of us had, so we used Google's Colab as instructed by Mr.Berrahou. Colab provided us with more than 300 GB of stockage, 12 GB of RAM, a strong GPU/CPU as well as fast download speeds. Thanks to that, downloading and setting the database was a breeze. And it enabled us to execute algorithms that wouldn't be possible, or would've took way too long on our humble computers.

## **2 - 3 Libraries used:**

Here shall be named some of the most important libraries used:

### **A – Scikit Learn:**

Scikit Learn is based about Machine Learning in Python. Some of its properties:

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

It can be used for:

- **Classification:**  
Identifying to which category an object belongs to.  
Applications: Spam detection, Image recognition.  
Algorithms: SVM, nearest neighbors, random forest, ...
- **Regression:**  
Predicting a continuous-valued attribute associated with an object.  
Applications: Drug response, Stock prices.  
Algorithms: SVR, ridge regression, Lasso, ...
- **Clustering:**  
Automatic grouping of similar objects into sets.  
Applications: Customer segmentation, Grouping experiment outcomes  
Algorithms: k-Means, spectral clustering, mean-shift, ...
- **Dimensionality reduction:**  
Reducing the number of random variables to consider.  
Applications: Visualization, Increased efficiency  
Algorithms: PCA, feature selection, non-negative matrix factorization.
- **Model selection:**  
Comparing, validating and choosing parameters and models.  
Goal: Improved accuracy via parameter tuning  
Modules: grid search, cross validation, metrics.
- **Preprocessing:**  
Feature extraction and normalization.  
Application: Transforming input data such as text for use with machine learning algorithms.  
Modules: preprocessing, feature extraction.



Figure 37: Scikit Learn logo

### **B – matplotlib:**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.



Figure 38: matplotlib logo

### **C – Open CV:**

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

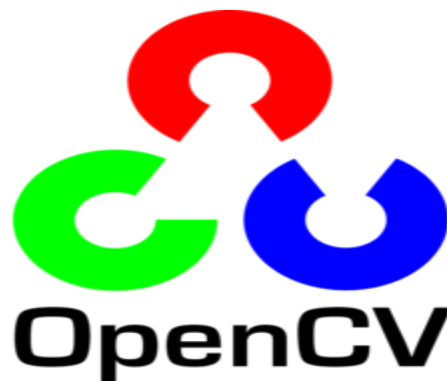


Figure 39: OpenCV logo

## **2 – Analyzing the results:**

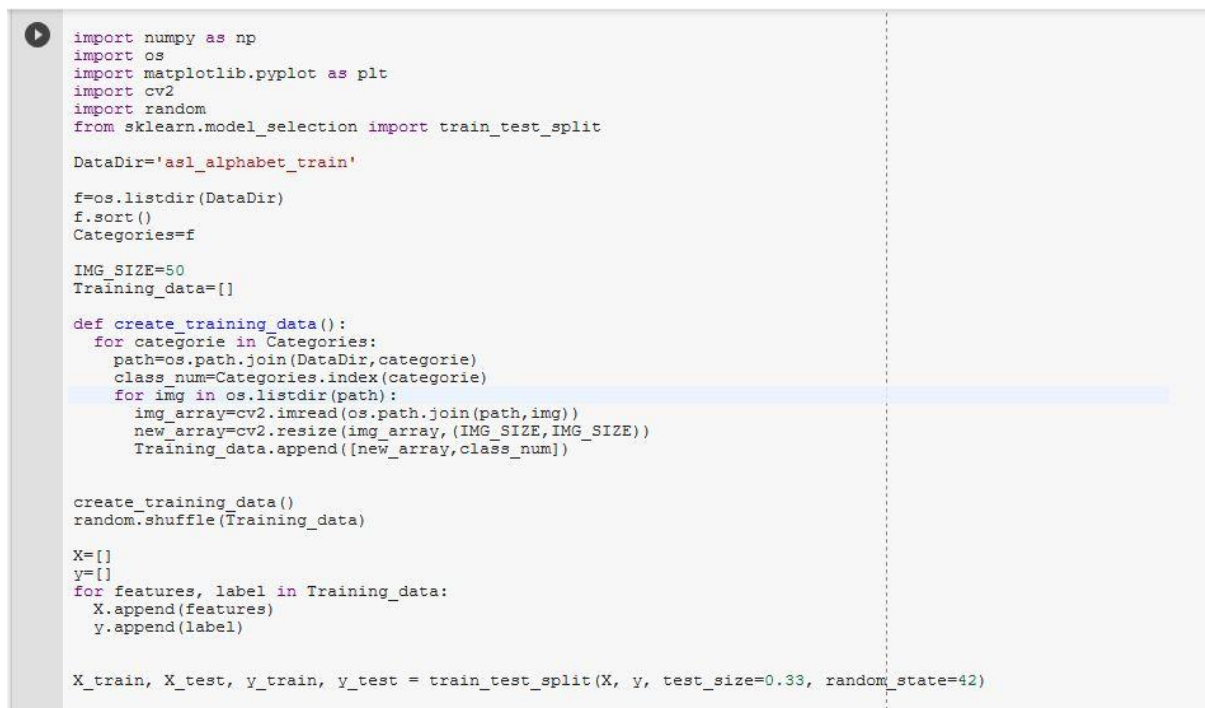
### **2 – 1 Dataset Preparation:**

We chose to use the ASL sign language dataset from Kaggle, as it offers a big number of raw images which helped us understand more the functioning of the algorithms. The important size of the dataset lead to big waiting times during processing phases, and sometimes, even stopped the Google Colab virtual machine due to surpassing the usable amount of ram of the machine.

The dataset contained two folders; `asl_alphabet_train` and `asl_alphabet_test`. But for the sake of simplicity we only used the training dataset which we split to training and validation since we used cross validation method s to conclude on the precision of the algorithms.

We used the function “`train_test_split()`” of the “`sklearn.model_selection`” library , which splits our features and labels sets into two subsets train and test.

The following figure illustrates the code used to split the dataset:



```
import numpy as np
import os
import matplotlib.pyplot as plt
import cv2
import random
from sklearn.model_selection import train_test_split

DataDir='asl_alphabet_train'

f=os.listdir(DataDir)
f.sort()
Categories=f

IMG_SIZE=50
Training_data=[]

def create_training_data():
    for categorie in Categories:
        path=os.path.join(DataDir,categorie)
        class_num=Categories.index(categorie)
        for img in os.listdir(path):
            img_array=cv2.imread(os.path.join(path,img))
            new_array=cv2.resize(img_array, (IMG_SIZE,IMG_SIZE))
            Training_data.append([new_array,class_num])

create_training_data()
random.shuffle(Training_data)

X=[]
y=[]
for features, label in Training_data:
    X.append(features)
    y.append(label)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

**Figure 40:** `train_test_split()` use

## **2 – 2 Implementation of the classifiers with the different descriptors:**

In order to compare the efficiency of the different combinations of the classifiers and descriptors we tried the following combinations:

- KNN with HOG
- KNN with LBP
- KNN with Hu Moments
- SVM with HOG
- SVM with LBP
- SVM with Hu Moments

### **A - Results and interpretations:**

The following table illustrates the accuracy rates of the KNN algorithm using different values of K and different descriptors

Descriptor \K value	K=3	K=5	K=7
HOG	X	98.88%	98.52%
LBP	97.99%	95.36%	95.36%
Hu Moments	6.93%	7.32%	7.43%

Figure 41: Results of the implementation of the KNN algorithm

The execution of the SVM with the different descriptors we came out with the following results:

Descriptor	Accuracy
HOG	X
LBP	3.27412 %
HU Moments	3.2532 %

Figure 42: accuracy rate of different SVM/Descriptors combinations

### **B - Cross validation:**

To validate the model we had to use cross validation methods to make sure that our model is capturing most of the patterns from the data correctly that's why we tried shuffling the data before inputting it to the descriptors the results of the execution were pretty much the same .

We also used the confusion matrix to make sure that our accuracy rate were correct, the resulting matrixes confirmed the results obtained above.

**Confusion Matrix:**

A confusion matrix is a summary of prediction results on a classification problem.

The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix.

The confusion matrix shows the ways in which your classification model is confused when it makes predictions.

It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made as shown in the following table.

	<i>Class 1 Predicted</i>	<i>Class 2 Predicted</i>
<b>Class 1 Actual</b>	TP	FN
<b>Class 2 Actual</b>	FP	TN

Figure 43: Confusion matrix layout

The following figures illustrate the confusion matrixes of some of the classifier/descriptor combinations.



##### KNN AVEC HOG : K=7 ACCURACY : 98.52% #####

Actual \ predicted	[A]	[B]	[C]	[D]	[E]	[F]	[G]	[H]	[I]	[J]	[K]	[L]	[M]	[N]	[O]	[P]	[Q]	[R]	[S]	[T]	[U]	[V]	[W]	[X]	[Y]	[Z]	[SP]	[DEL]	[NOT]
[A]	[[ 980	0	0	0	11	3	0	0	0	0	1	2	0	0	0	0	0	1	1	4	0	2	0	1	0	0	0	0	0]
[B]	[ 0	996	0	0	1	0	0	0	1	0	2	0	0	0	0	0	0	0	0	1	0	1	0	2	0	0	0	0	0]
[C]	[ 4	0	1006	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0]
[D]	[ 0	2	0	977	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0]
[E]	[ 16	18	0	9	966	0	0	0	3	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0]
[F]	[ 0	0	0	0	0	1038	0	0	0	0	0	0	0	0	0	0	0	1	0	4	2	0	0	0	0	0	0	0]	
[G]	[ 0	0	0	0	0	0	985	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0]	
[H]	[ 0	0	0	0	0	0	0	1	984	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0]	
[I]	[ 1	1	0	2	5	2	0	0	982	0	3	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0]	
[J]	[ 0	0	0	0	0	0	0	0	0	998	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0]	
[K]	[ 0	2	0	0	1	0	0	0	2	0	975	0	1	0	0	0	0	1	0	1	5	2	0	0	0	0	0	0]	
[L]	[ 1	0	0	0	0	1	0	0	0	0	0	1001	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0]	
[M]	[ 0	0	0	0	0	0	0	0	0	0	0	963	6	0	0	0	0	0	1	2	0	0	0	1	0	0	0	0]	
[N]	[ 0	0	0	0	0	0	0	0	0	0	0	6	994	2	0	0	0	1	0	0	0	0	0	0	2	0	0	1]	
[O]	[ 5	0	0	1	0	0	0	0	0	0	0	5	6	942	0	0	0	0	1	0	0	0	0	0	1	0	0	0]	
[P]	[ 0	0	0	0	0	0	0	1	0	0	0	0	0	0	963	0	0	0	0	0	0	0	0	0	0	0	0	0]	
[Q]	[ 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	974	0	0	0	0	0	0	0	0	0	0	0	0]	
[R]	[ 0	0	0	0	0	0	0	0	4	0	3	0	0	0	0	0	974	0	0	10	0	2	0	0	0	0	0	0]	
[S]	[ 0	2	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	949	0	0	0	0	0	3	1	0	0	0]	
[T]	[ 2	0	0	2	0	0	0	0	0	0	0	5	0	0	1	0	0	0	3	940	0	0	0	0	0	0	0	0]	
[U]	[ 0	0	0	2	0	0	0	0	1	0	0	0	0	0	0	0	12	0	2	933	6	9	1	0	0	0	0	0]	
[V]	[ 0	0	0	0	1	0	0	0	5	0	8	0	0	0	0	0	5	2	0	38	949	6	0	0	0	0	0	0]	
[W]	[ 0	1	0	0	0	0	0	0	0	0	16	0	1	0	0	0	3	0	0	7	24	922	4	0	0	0	0	0]	
[X]	[ 0	0	0	0	4	0	0	0	7	0	0	0	1	0	1	0	1	5	0	5	1	0	936	0	0	0	0	0]	
[Y]	[ 0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	948	0	0	0]		
[Z]	[ 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1017	0	0	0]	
[SP]	[ 0	0	0	0	0	0	0	1	0	0	3	0	1	0	0	0	0	1	0	0	0	0	0	0	0	2	1038	0]	
[DEL]	[ 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	982	0]	
[NOT]	[ 0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	973]]]	

Figure 44: Confusion matrix for KNN / HOG with k=7

```
##### SVM w/T HU MOMENTS: ACCURACY: 3.253228739115293% #####
Actual\ predicted [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Y] [Z] [SP][DEL][NK]
[A] [[ 0 1033 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[B] [ 0 934 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[C] [ 0 966 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[D] [ 0 980 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[E] [ 0 997 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[F] [ 0 981 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[G] [ 0 1017 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[H] [ 0 1026 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[I] [ 0 1015 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[J] [ 0 965 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[K] [ 0 966 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[L] [ 0 983 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[M] [ 0 1011 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[N] [ 0 984 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[O] [ 0 995 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[P] [ 0 972 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[Q] [ 0 997 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[R] [ 0 992 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[S] [ 0 989 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[T] [ 0 1007 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[U] [ 0 960 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[V] [ 0 964 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[W] [ 0 1026 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[X] [ 0 1011 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[Y] [ 0 1007 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[Z] [ 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[SP] [ 0 975 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[DEL] [ 0 978 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[NK] [ 0 980 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Figure 45: Confusion matrix for SVM /Hu Moments

##### KNN W/T HU MOMENTS: k=1, accuracy=7.61% #####

Actual\ predicted	[A]	[B]	[C]	[D]	[E]	[F]	[G]	[H]	[I]	[J]	[K]	[L]	[M]	[N]	[O]	[P]	[Q]	[R]	[S]	[T]	[U]	[V]	[W]	[X]	[Y]	[Z]	[SP]	[DEL]	[NOT]
[A]	75	37	34	28	41	37	43	41	32	36	29	40	36	38	25	26	32	28	29	37	37	36	20	38	35	36	29	17	33]
[B]	32	67	31	34	37	44	31	35	29	41	42	34	44	41	25	36	39	22	23	38	17	26	30	26	30	25	28	16	24]
[C]	29	32	60	48	41	16	14	21	23	39	43	43	31	29	44	31	38	27	34	40	49	33	48	31	19	33	36	14	43]
[D]	30	42	52	56	38	26	26	33	26	27	45	31	45	24	43	29	24	43	31	39	42	37	39	42	25	26	38	22	36]
[E]	29	36	34	32	52	49	37	29	39	27	26	37	27	28	40	36	33	39	28	43	34	43	41	38	31	39	20	19	31]
[F]	40	40	34	38	35	65	43	41	36	32	22	20	37	39	29	28	30	21	24	35	32	29	33	42	20	22	33	21	31]
[G]	49	34	22	26	30	57	107	38	35	48	33	38	34	45	19	48	37	26	30	32	24	25	21	36	28	23	31	32	25]
[H]	49	37	21	31	35	37	40	56	39	38	33	39	31	47	25	42	22	31	23	39	22	28	23	41	26	25	44	21	27]
[I]	41	31	31	45	44	47	29	39	75	47	36	35	22	31	27	20	22	39	34	45	29	30	47	40	24	29	33	16	36]
[J]	42	37	29	28	31	34	35	44	46	39	29	35	33	41	27	34	30	31	20	32	44	27	24	33	35	44	35	9	27]
[K]	26	32	32	29	31	39	27	30	45	32	64	45	28	35	23	43	33	44	28	34	39	42	30	40	25	42	27	24	32]
[L]	33	37	30	39	42	32	21	40	28	32	47	64	26	39	39	28	38	38	38	39	39	39	37	28	15	44	14	28	36]
[M]	41	43	38	33	28	36	36	25	29	27	30	26	55	37	36	33	38	34	43	32	32	34	32	32	37	41	32	25	26]
[N]	41	52	30	22	27	48	31	43	29	42	32	18	36	72	21	37	39	20	31	32	34	25	24	28	35	21	55	11	29]
[O]	31	32	32	25	22	30	29	26	23	29	33	35	49	24	62	32	53	37	36	32	43	41	43	32	50	38	22	29	48]
[P]	22	32	22	38	33	48	44	48	33	47	44	32	35	32	37	76	44	24	24	44	31	33	36	22	28	17	40	17	19]
[Q]	21	30	26	30	26	28	35	21	33	32	29	36	36	33	39	51	52	28	40	37	34	46	34	46	35	31	32	22	34]
[R]	23	23	32	42	34	28	23	28	34	37	42	31	29	27	40	25	35	54	39	46	47	45	49	37	52	39	29	30	43]
[S]	18	30	36	26	26	31	11	27	23	34	39	35	35	31	50	26	32	38	61	22	44	41	34	35	41	57	30	22	42]
[T]	30	36	31	29	30	29	26	29	29	32	36	28	34	29	31	36	27	33	46	70	36	24	28	33	47	32	31	27	33]
[U]	26	28	34	36	36	19	26	24	24	29	49	39	31	17	50	29	43	29	46	37	52	42	35	36	36	37	20	37	42]
[V]	27	28	33	41	40	40	28	26	30	36	38	32	39	19	30	35	36	29	41	25	54	48	35	33	36	37	41	27	42]
[W]	19	38	39	34	35	28	19	36	41	30	41	38	25	38	39	34	33	32	32	34	41	39	73	47	30	46	19	21	33]
[X]	31	19	31	24	34	30	26	44	39	20	37	24	35	32	30	25	29	45	28	47	32	26	42	79	29	31	33	23	22]
[Y]	45	26	30	32	36	29	21	17	21	23	35	20	27	28	35	16	31	34	41	37	44	43	44	30	68	39	32	33	49]
[Z]	25	27	31	22	26	36	25	30	24	30	45	35	24	24	38	30	36	34	44	42	40	52	35	28	37	64	27	16	47]
[SP]	26	43	24	38	35	36	30	42	35	36	19	27	48	51	29	42	25	27	33	26	34	31	29	46	35	29	72	14	36]
[DEL]	15	16	18	11	19	21	23	20	17	18	18	34	17	17	22	20	18	41	34	19	26	27	22	26	35	23	17	384	26]
[NOT]	30	20	34	46	19	28	23	32	33	29	25	37	31	24	30	18	44	38	41	46	37	28	39	32	57	38	24	29	63]

Figure 46: Confusion matrix for KNN / Hu Moments with k=1



### **3 - Conclusion:**

In the first matrix we can notice that the values are distributed diagonally, which is explained by the fact that the combination had very good results in identifying characters, which can be seen in the high accuracy rate (exceeding 98%).

On the other hand, the second matrix presents a row distribution of values, which can be explained by the fact that the combination failed to recognize and differentiate the characters hence the 3% accuracy rate. The same thing can be said about the last matrix.

# Bibliography

[https://scikit-learn.org/stable/auto\\_examples/index.html#general-examples](https://scikit-learn.org/stable/auto_examples/index.html#general-examples)

<https://www.kaggle.com/grassknoted/asl-alphabet>

<https://colab.research.google.com/>

<https://stackoverflow.com/questions/49070242/converting-images-to-csv-file-in-python>

<https://www.geeksforgeeks.org/confusion-matrix-machine-learning/>

<https://towardsdatascience.com/supervised-machine-learning-classification-5e685fe18a6d>

<https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/>

<https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>

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

<https://opencv.org/about/>

<https://scikit-learn.org/>

<https://matplotlib.org/>

<https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/>

<https://www.learnopencv.com/shape-matching-using-hu-moments-c-python/>

<https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f>

<https://www.youtube.com/channel/UCdKG2JnvPu6mY1NDXYFfN0g>

References:

OpenCV-*Python Tutorials Documentation Release 1*- Alexander Mordvintsev & Abid K, Nov 05, 2017