2020-2021

הפקולטה להנדסה קמפוס דהאן
ע"ש אלכסנדר קופקין
אוניברסיטת בר־אילן

Gesture Detection And Analysis

Instructors:

Professor Tzvika Lotker

Doctor Adi Makmal

Autor's:

Ron Kandelshein

Elihyo Etin

Abstract

On the following paper we will present our final project we undertook as a part of Bar-Ilan's University Computer Engineering bachelor's degree. We will present the problem we aimed to solve, our goals and the tools we implemented in-order to achieve them.

The project's idea developed from firsthand experience on the hardships people with physical disabilities experience in their every day's life, in several aspects. In general, as the disability is more severe the cost of any well-being Improvement is more expensive. We aimed to use the knowledge from our study years to develop a product that will ease the life's of people, without extensive costs.

As both of us are seeking to enrich our knowledge on machine learning area of study, we chose to use the tools it contributes to tackle the problem and enrich our knowledge - theoretical and practical.

The paper will introduce and define the problems we faced and will review our research. Following the intro, we will review the basic relevant concepts of the field that were developed and optimized during the years. After the theory review, we will display our main work, divided to the 3 main areas we focused: Architecture, Image processing algorithm and application development. The paper ends with future work ideas concluded from the results.

TABLE OF CONTENTS

Chapter 1 - Introduction

1. First Steps

Our main idea approaching the project was the following question:
How can we access people in need of technology we got familiar with during our time at the university?
That question along with our interest in the Machine and Deep learning fields is what drove us during the past year to further study and experiment on the subject.
The people we aimed to help in our first draft of the idea were people who suffer from severe physical disabilities. More specifically, people who cannot operate technological products with their hand alone i.e., phone, tablet, pc, etc.

2. The Problem

Following our established guidelines, we conducted research to assess the following:

a. How large is the population of people we aimed to help?
b. What are the difficulties they are facing in everyday life?
c. Where can we contribute?

Our basic research was based on The Equal Rights Commission for Persons with Disabilities along with the Department of Justice. Every year the two departments conduct research that addresses the state of the people suffering from disabilities in a vary of fields. The main results regarding our interest where:

The How:

As mentioned, 30% of the disabled population suffer from sever disability. We contacted several organizations to help us map the population who suffer from physical disability and could benefit from our work.

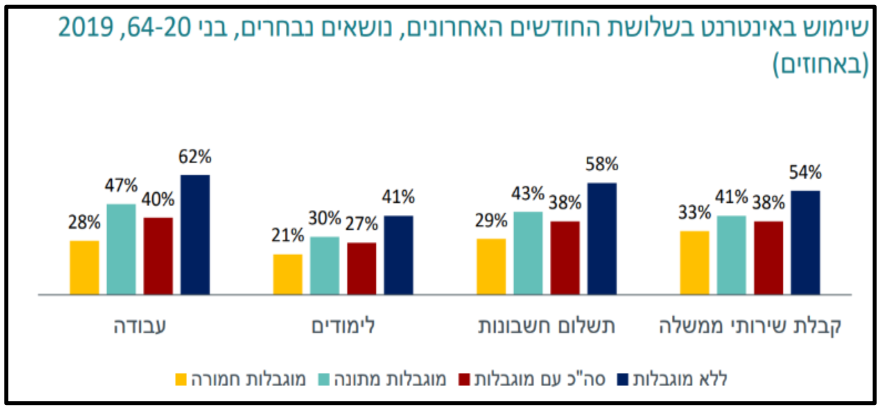
First, we contacted the Myers-JDC-Brookdale Institute to draw more information from their past study. After we've contacted the lead researcher, we received more data that helped us later, but we'd been told that there was no mapping of the disability's types in the national level and therefore, we could not quantify our target population.

Later, we contacted the IDF Disabled Veterans Organization. In several meetings with the Chairman of the Shefela District, so to establish communication with contact who could potentially benefit from our development, and we've received information on those contacts. Of course, individual adaptations will have to take place, as different ranges of motions apply to each case.

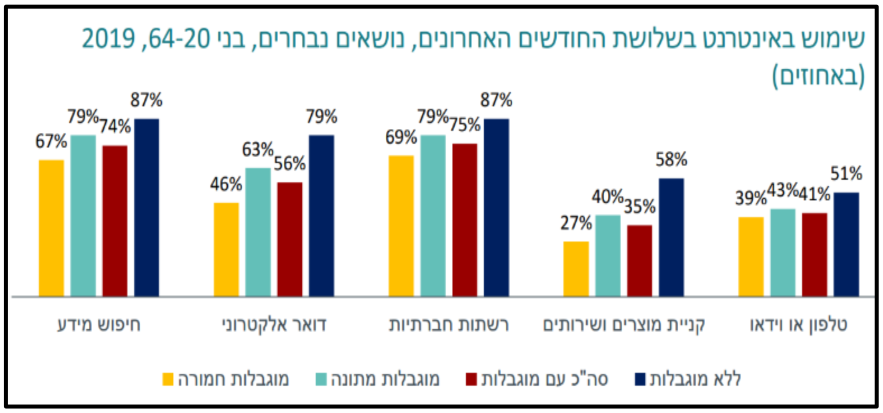
The What:

There is a significant difference in the use of internet via PC versus cellphone between disabled people and the general population in Israel. The data shows that only 51% of the people with medium disability and 21% with severe disability use PC compared to the general population, which was estimated in 71%.

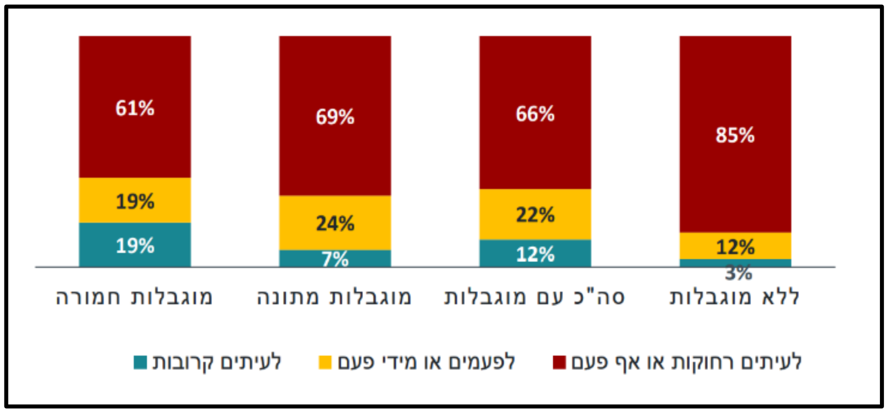
As the following graphs demonstrate, people with severe disability use the internet less frequently for various reasons.



In this graph we see the difference in internet use between people without disability (Blue) and disabled people (Red) in 4 subjects: Work, Studies, bill payments and receiving government services.



In this graph we see the difference in internet use between people without disability (Blue) and disabled people (Red) in 4 subjects: Searching information in the net, E-mail uses, Social networks and using phone or video.



In this graph we see the difference in loneliness feeling between people without disability (the right graph) and disabled people (3- left graphs) in when the colors categorized as: red-rarely or never, yellow-sometimes or occasionally, blue-often

Furthermore, 14% of population with sever disability report that they have no one to trust in a time of need. The following graph represent the answer people gave to the question: How often do you feel lonely?

The Where:

From the personal experience and the data at hand we saw the gap in the use on the internet and a phone\Pc in general. We drafted an idea that aims to help a smaller part of the population mentioned, people with severe disabilities. To do that we aimed to develop a system which can monitor basic movements and integrating it to a phone\Pc to achieve the ability to do basic functions at first and later using the internet as we all do.

As mentioned, 30% of the disabled population suffer from sever disability. We contacted several organizations to help us map the population who suffer from physical disability and could benefit from our work.

First, we contacted the Myers-JDC-Brookdale Institute to draw more information from their past study. After we've contacted the lead researcher, we received more data that helped us later, but we'd been told that there was no mapping of the disability's types in the national level and therefore, we could not quantify our target population.

Later, we contacted the IDF Disabled Veterans Organization. In several meetings with the Chairman of the Shefela District, so to establish communication with contact who could potentially benefit from our development, and we've received information on those contacts. Of course, individual adaptations will have to take place, as different ranges of motions apply to each case.

3. Main Goal

Develop a model that will allow users that cannot operate a cellphone \ computer, due to their disability, to perform trivial tasks to improve their well-being without extensive costs. Furthermore, we aim to establish a basic infrastructure that will allow future work to enhance and optimize the final product.

4. Sub Goals

a. Design and train a Neural Network that classifies a variety of pre-determined movements. i.e., robust arm movements, head movements etc. (to do so we are using TensorFlow and Keras libraries via python.)
b. Assimilate the network in a smartphone application that will utilize its ability to activate certain pre-determined protocols on a variety of software. We aim to develop a cross-platform app that will support our objective with an emphasis on simplicity.
c. Design and apply image-processing algorithms and tools via OpenCV & NumPy libraries to handle our data and optimize it to our needs.
d. Conduct research on the relevant part of the population in order to assess the types of movement we should apply.

Chapter 2 - Scientific background

This chapter is an introduction to the main topics in our project- Machine learning and image processing subjects.

2.1 Machine learning

Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns, and make decisions with minimal human intervention. Machine learning algorithms build a model based on sample data, known as "training data", to make predictions or decisions without being explicitly programmed to do so.

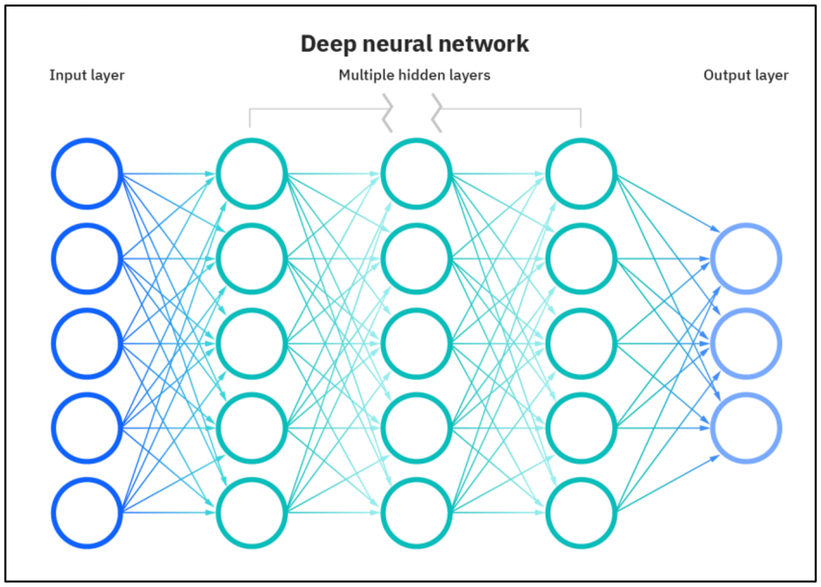
2.2 Deep learning

Deep learning is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised.
The adjective "deep" in deep learning refers to the use of multiple layers in the network. Deep learning is a class of machine learning algorithms that uses multiple layers to progressively extract higher-level features from the raw input. For example, in image processing, lower layers may identify edges, while higher layers may identify the concepts relevant to a human such as digits or letters or faces.

2.3 Neural networks

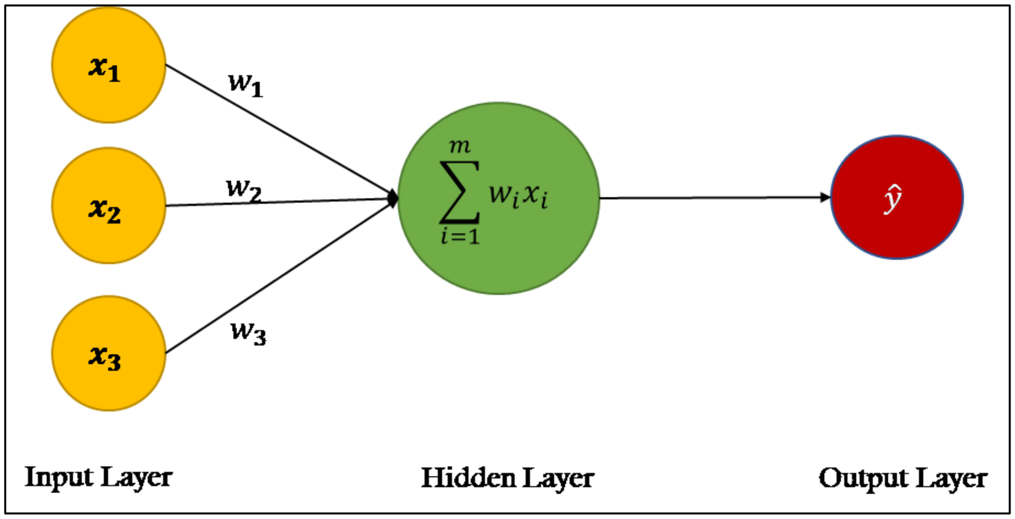
A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature

Artificial neural networks (ANNs) are comprised of a node layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.



Neural networks rely on training data to learn and improve their accuracy over time. However, once these learning algorithms are fine-tuned for accuracy, they are powerful tools in computer science and artificial intelligence, allowing us to classify and cluster data at a high velocity. Tasks in speech recognition or image recognition can take minutes versus hours when compared to the manual identification by human experts. One of the most well-known neural networks is Google's search algorithm.

2.3.1 Basic architecture



The output value can be continuous, binary, or categorical. For the neural network to easier work with the input we must standardize it, etc.: Normalization. W-weights- Essential for network functioning because the network performs the learning by adjusting the weights and by that it decides what information passes, how much and when. The criterion for the output error is the mean square error: $C = \frac{1}{2}(\hat{Y} - Y)^2$,when $\hat{Y}$ is the output value we get from the neural network and Y is the real value. If Y is a vector size n, we can write the error as:

$$C = MSE = \frac{1}{n}\sum_{i=1}^{n}(\widehat{Y}_i - Y_i)^2$$

2.3.2 **Basic elements of the network**

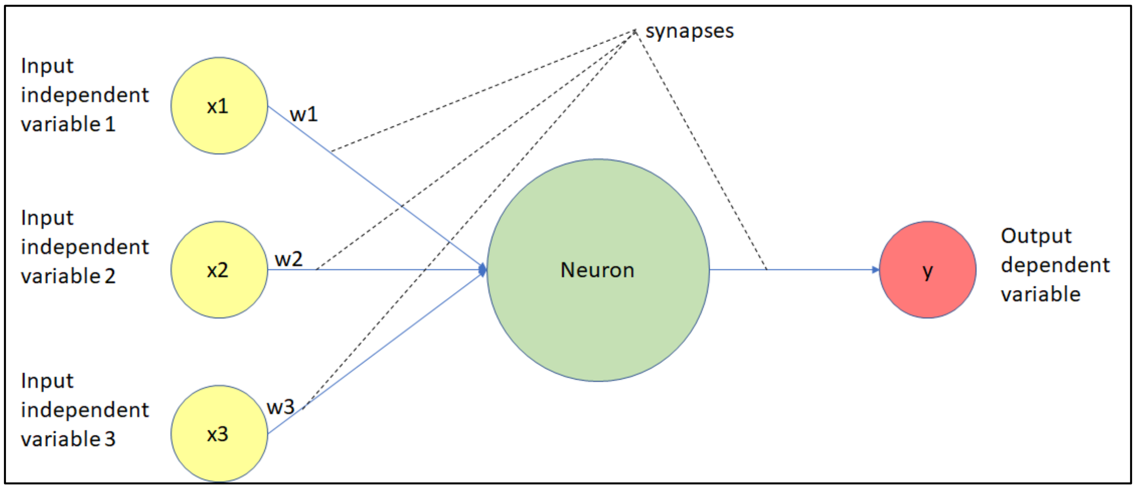
After gaining macro understanding of the network, we reviewed the basic elements.

- **Neuron**

Neurons in deep learning models are nodes through which data and computations flow.

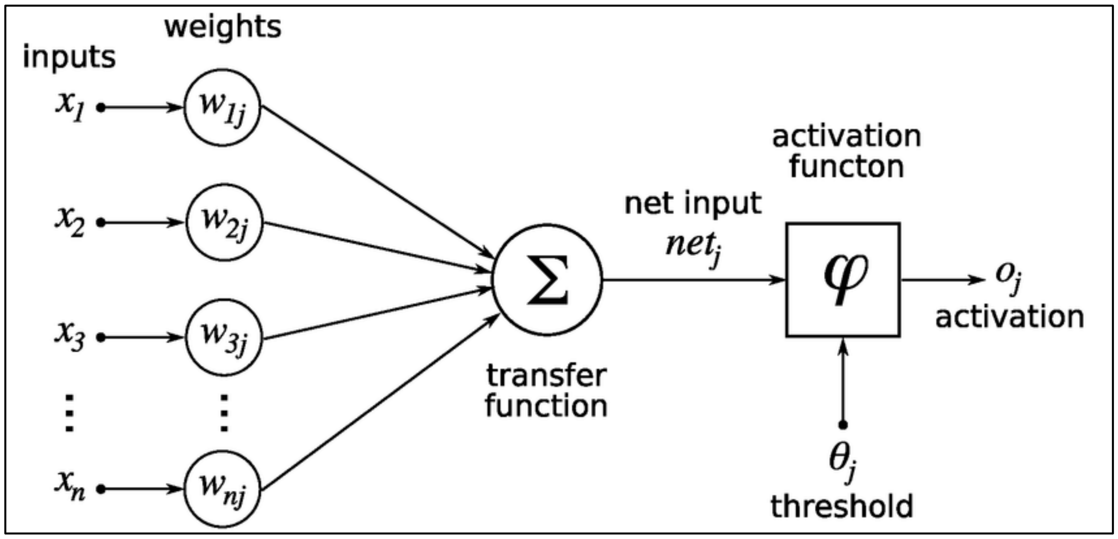
Neurons work like this:

- They receive one or more input signals. These input signals can come from either the raw data set or from neurons positioned at a previous layer of the neural net.
- They perform some calculations.
- They send some output signals to neurons deeper in the neural net through a synapse.


synapses
Input independent variable 1
Input independent variable 2
Input independent variable 3
x1
x2
x3
w1
w2
w3
Neuron
y
Output dependent variable

Functionality diagram of the neuron in a deep learning neural network

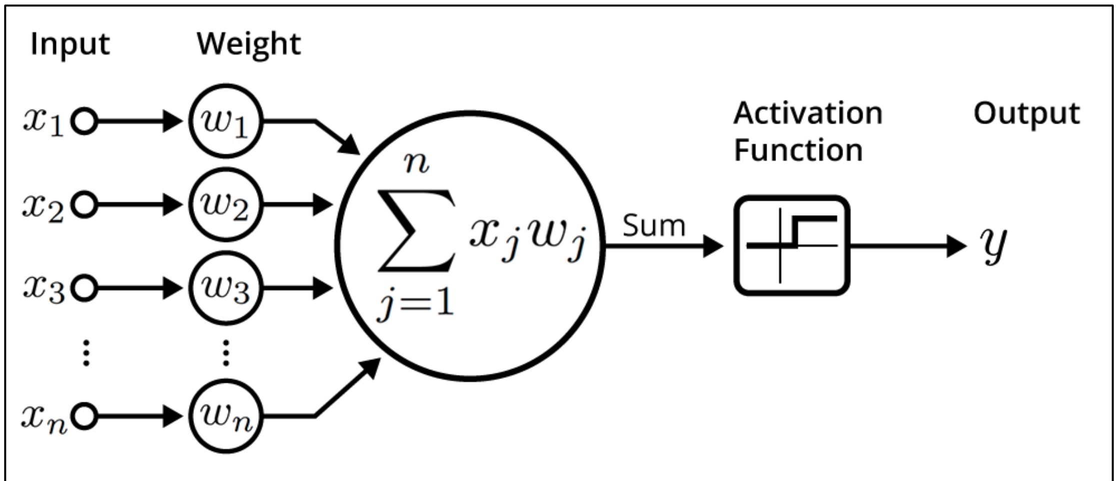
- **Network Calculation:**



The Activation function is applied on the following sum:

$$\varphi(\sum_{i=1}^{n} w_i x_i)$$

Activation Function adds non-linearity to the calculation and defines how the weighted sum of the input is transformed into an output from a node or nodes in a layer of the network. Basically, it decides whether the neuron should be activated or not.

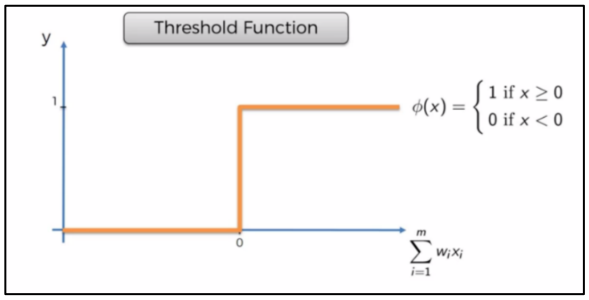


An illustration of an artificial neuron. Source: Becoming Human.

- **Activation Functions:**

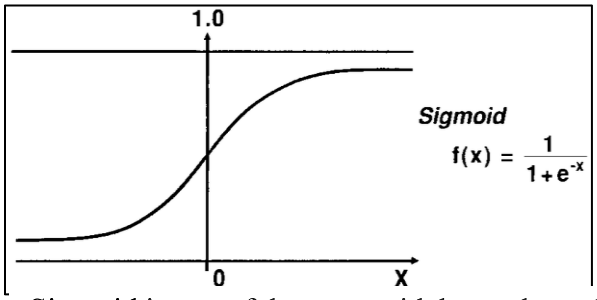
The following activation functions are the most common:

1) Threshold's function:



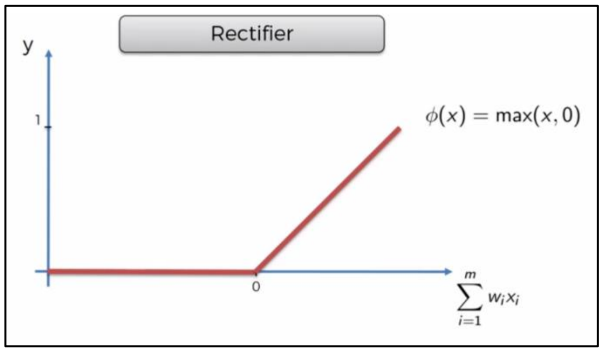
If the input to the activation function is greater than a threshold, then the neuron is activated, else it is deactivated, i.e. its output is not considered for the next hidden layer.

2) Sigmoid function:



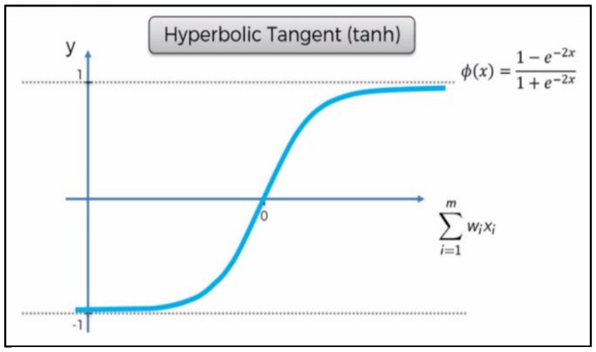
Sigmoid is one of the most widely used non-linear activation function. Sigmoid transforms the values between the range 0 and 1.

3) ReLU:

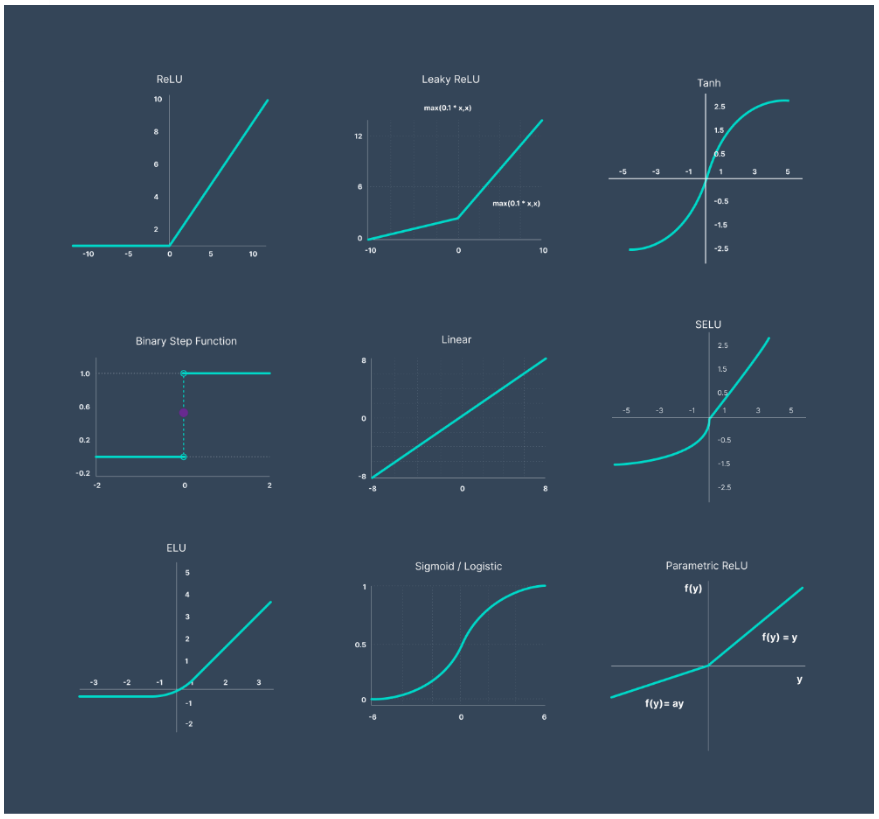


ReLU stands for Rectified Linear Unit. The main advantage of using the ReLU function over other activation functions is that it does not activate all the neurons at the same time. This means that the neurons will only be deactivated if the output of the linear transformation is less than 0.

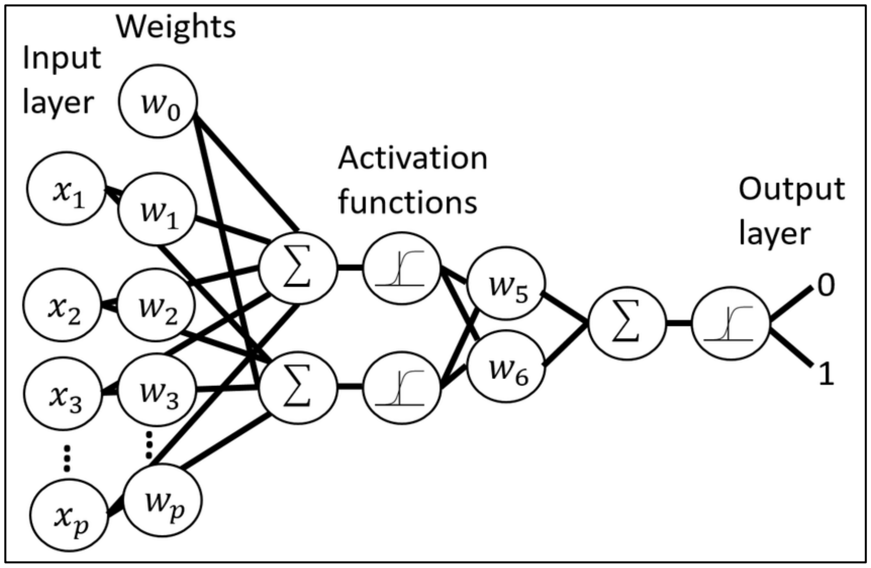
4) Hyperbolic Tangent:



The tanh function is very similar to the sigmoid function. The only difference is that it is symmetric around the origin. The range of values in this case is from -1 to 1. Thus the inputs to the next layers will not always be of the same sign.



More examples of activation functions



Multiple activation functions can be used in

a network architecture as shown above

- **Hidden layers**

  Important aspect in building the neural network is using the Hidden layers- which are the middle layers we should add between the input and the output. Part of the weights W are 0 and others are not it's all depended on the importance of the information in the neurons connected to the weight. Each neuron is connected to specific output type and specific inputs, each input will affect the final calculation.

2.3.3 Working Method:

In machine learning neural networks we build architecture that contains the program that learns what it needs to do using the inputs we provide, compared to normal code in which we give all the instructions, make sure all the cases, and set all the rules.

So how it works?

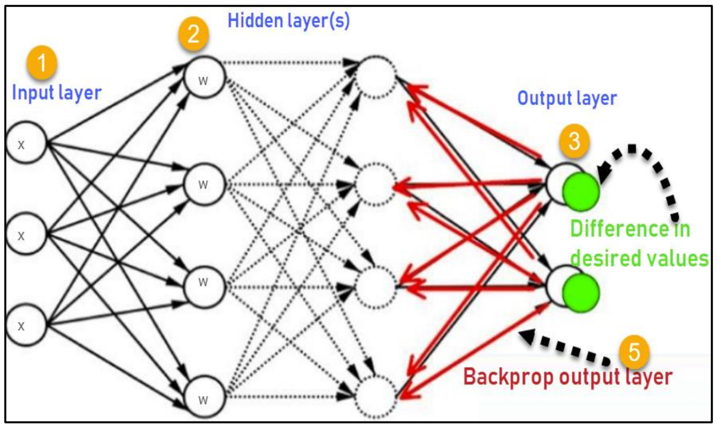
C= is the error variable between the actual value in the output-Y and the output value that the neural net extract.

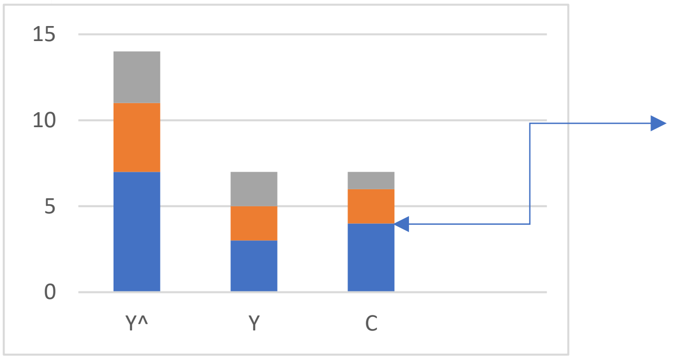
For guided learning in ML, we need to know the right output values that fit the inputs, so we will enter the inputs wait for the output of the neural network and then we will give it the true output (which we know in advance) so the neural network could calculate the error between the prediction and the true output value.

this will happen in the training step and each epoch the neural net will try to adjust and update the W-weights so the calculate error will be minimal, when we will reach low error, it means the network working well and the net extract output that close enough to the output we expect.

The weights are the only adjustable variables in the net, after the error calculation the information is backpropagated for the weights update. Each weight updated to a value corresponded to it influence on the error. Some weights will have low values some higher.

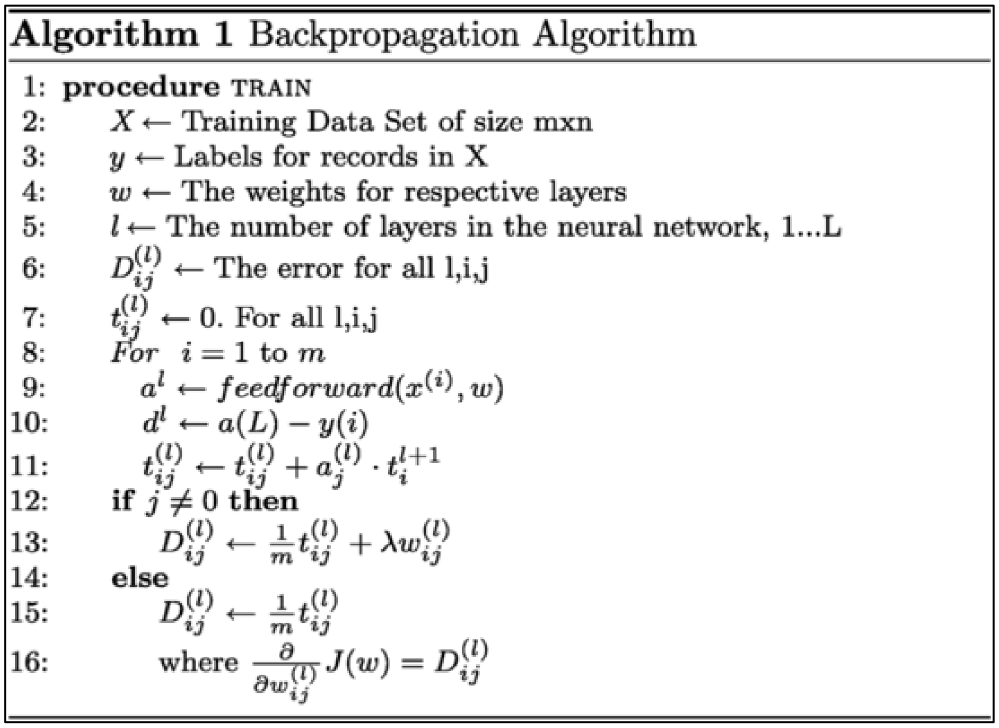
All This process called Back-propagation. And its important idea and method of the machine learning nets system of learning.





After each weight update the error-C will decrease and we will strive to reach a minimum C.

Back-propagation algorithm pseudo code:



Algorithm 1 Backpropagation Algorithm

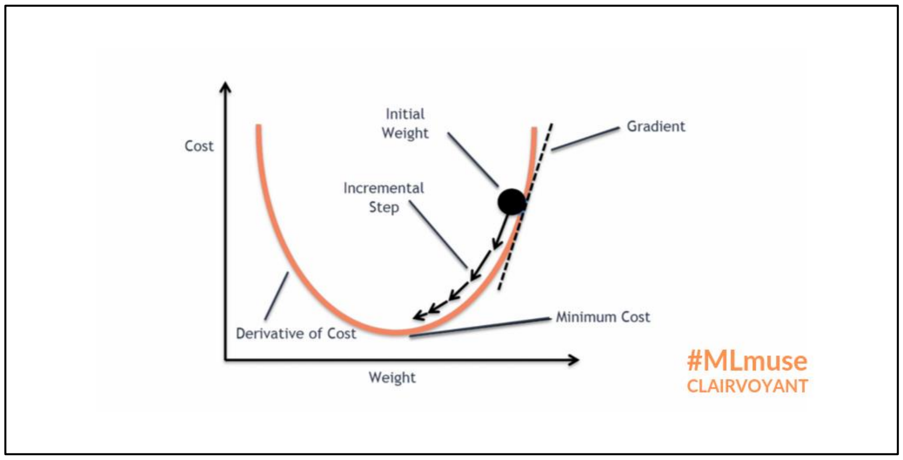
```
 1: procedure TRAIN
 2:     $X \leftarrow$ Training Data Set of size mxn
 3:     $y \leftarrow$ Labels for records in X
 4:     $w \leftarrow$ The weights for respective layers
 5:     $l \leftarrow$ The number of layers in the neural network, 1...L
 6:     $D_{ij}^{(l)} \leftarrow$ The error for all l,i,j
 7:     $t_{ij}^{(l)} \leftarrow 0$. For all l,i,j
 8:     $For \;\; i = 1$ to $m$
 9:       $a^l \leftarrow feedforward(x^{(i)}, w)$
10:       $d^l \leftarrow a(L) - y(i)$
11:       $t_{ij}^{(l)} \leftarrow t_{ij}^{(l)} + a_j^{(l)} \cdot t_i^{l+1}$
12:     if $j \neq 0$ then
13:        $D_{ij}^{(l)} \leftarrow \frac{1}{m} t_{ij}^{(l)} + \lambda w_{ij}^{(l)}$
14:     else
15:        $D_{ij}^{(l)} \leftarrow \frac{1}{m} t_{ij}^{(l)}$
16:        where $\frac{\partial}{\partial w_{ij}^{(l)}} J(w) = D_{ij}^{(l)}$
```

The algorithm of the weight's updates for minimum error-Also called the optimizer that optimize the error lose.

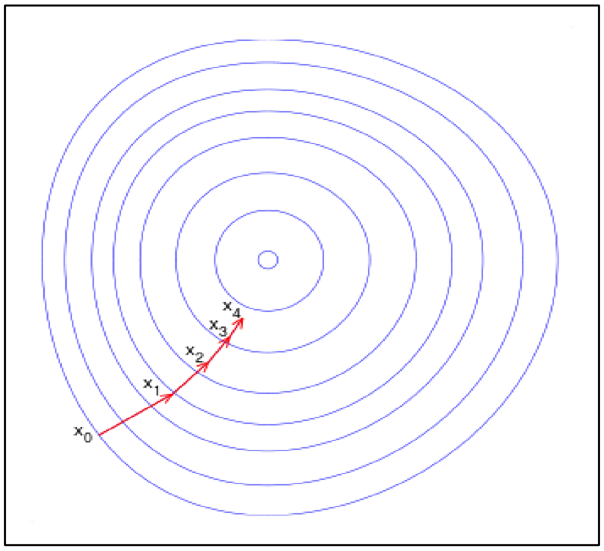
Gradient descent

Gradient descent is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function. The idea is to take repeated steps in the opposite direction of the gradient (or approximate gradient) of the function at the current point, because this is the direction of steepest descent.

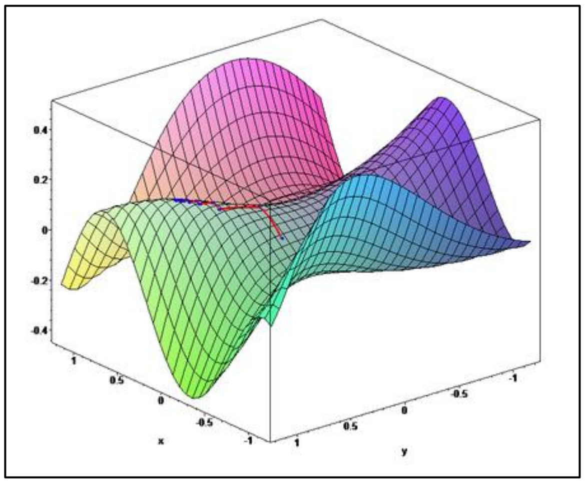
We chose a point on the graph calculate the derivate in the point and keep moving down until we arrive to minimal derivate (derivate->0).

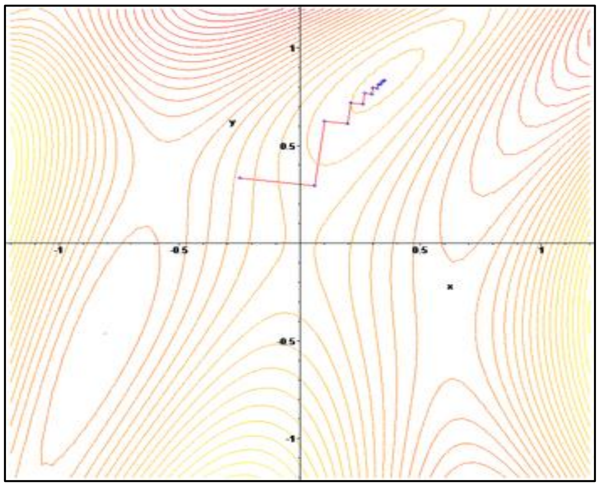


In 2D and 3D it will look like this:



Minimum point



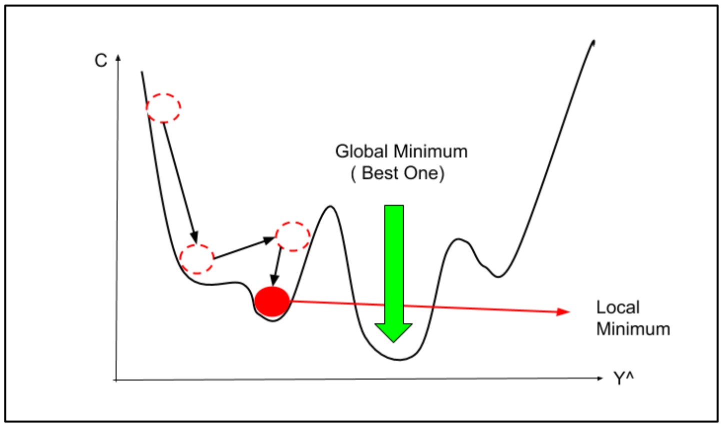


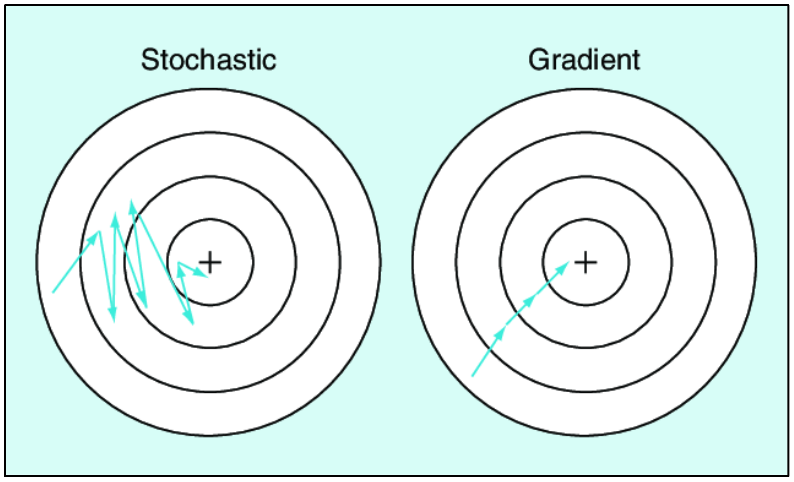
Stochastic gradient descent

In Gradient Descent, we consider all the points in calculating loss and derivative, while in Stochastic gradient descent, we use single point in loss function and its derivative randomly.

This algorithm finds the local minimum points and then takes the minimum between them as the global minimum.

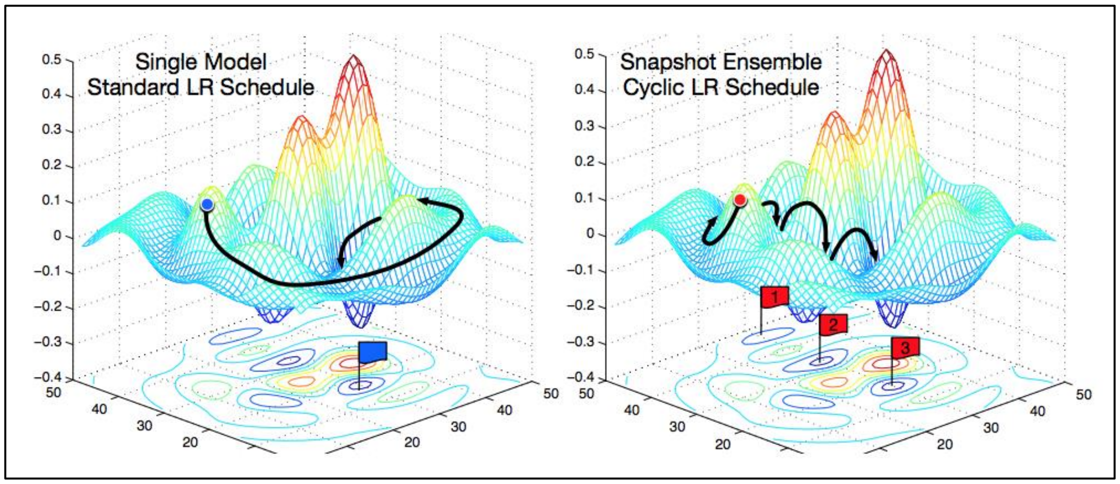
C curve (error curve) does not have to be parabolic.





Illustrating the difference between the cases

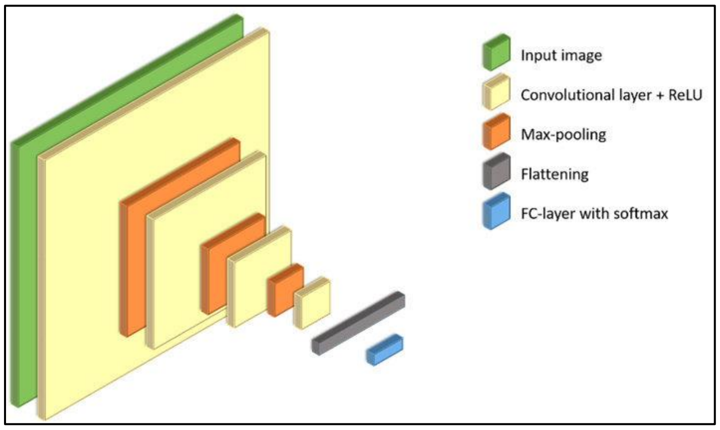
In this algorithm we perform each line (batch of information/data that permeates throw the net layers) and then we update the weights (with back propagation algorithm), compared to performing all the lines (all the batch of data) and in the end update the weights (in this way we won't find the local minimums). -This algorithm is also faster.



2.4 CNN- Convolutional neural networks

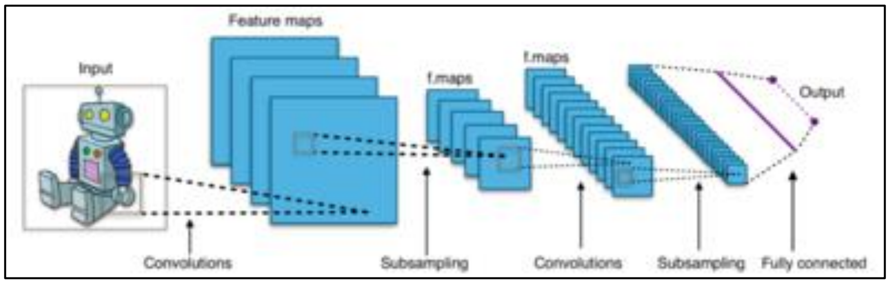
CNN networks primarily used to analyze visual images and used for recognition in images and videos. CNN networks use convolution instead of normal matrix multiplication in at least one of their layers. CNN networks based on the shared-weight architecture of the convolution kernels or filters that slide along input features and provide translation equivariant responses known as feature maps. Architecture: input layer, output layer and hidden layers that contains convolution layers. The mostly used activation function is Relu. Also, it is common to use pooling layers (description below). CNN networks are fully connected networks which means every neuron in one layer connected to all the neurons in the next layer- each neuron gets information from all the neurons in the previous layer. Each neuron activates specific function on the inputs from the previous layer, the function that activated determined by the "weights vector"- also called the "filter".

We wish to elaborate on the basic elements of the CNN by the following overview:

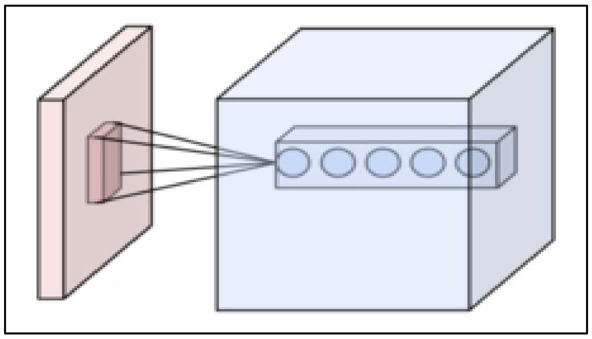


2.4.1 The convolution layers

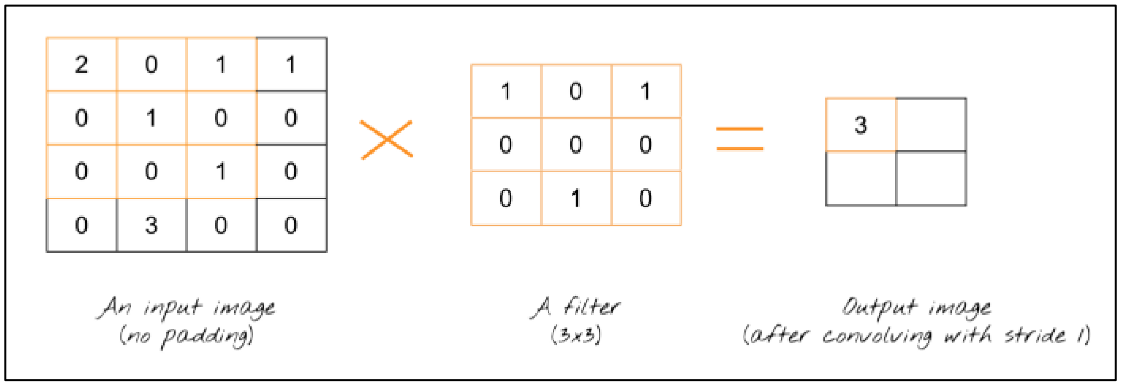
The convolution layer is the base layer of the CNN network. The layer contains set of filters with learning ability, the absorption field of the filters grow bigger as the depth of the input in the net. Each filter influences the input and apply his own changes upon it. Etc.: changes on specific high value areas in a frame for focusing on the specific area- for recognition tasks. Because images can be high number dimensional inputs each neuron in the network belong to specific area in the image (or frame) and the information permeate throw the neurons according to their area of effect. The convolution layer gives us a "features map" of the image.



CNN layers and the working method of the CNN architecture.



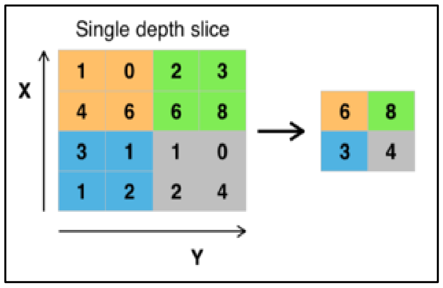
Neuron in the network connected to his field of affect

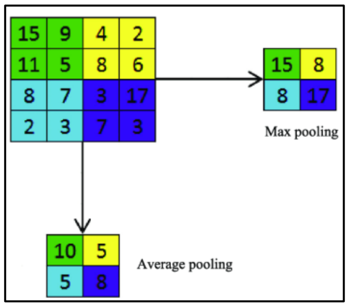


The convolution function between the pixels in the image and the convolution filter values.

2.4.2 Pooling layer

Let's review the idea behind the "pooling layer". Pooling is a way of taking sample for reduction the size of the data and passing only the high value data (max Pooling), or reduction of the data and passing the average of the values in a specific area (average pooling).

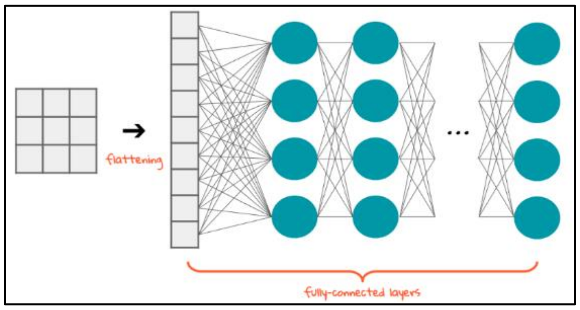


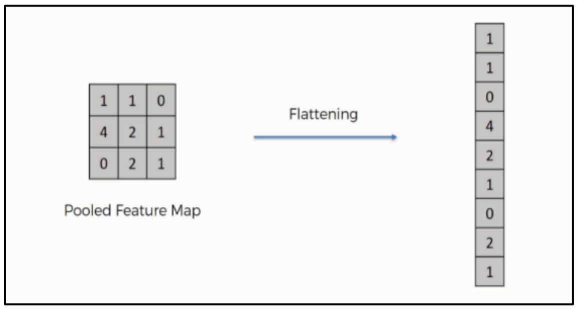


We determine the dimensions of the filter and the filter pass on the image and gives as the pooling results this is how we get the data reduction- taking only the important features of the image and passing them on. There are variety of activation functions that we can apply on the data, as mentioned in the activation functions, in the different layers to cancel the linearity in the image for extracting better features. Linearity Cancelation can be used for transition between dull color and dark color will be absolute and not gradual (linear), in this way we can extract the features of the image better.

2.4.3 Flattening layer

Flattening is the operation of converting the data into a 1-dimensional array. It is done to insert the output of a certain layer to next one. We flatten the output of the convolutional layers to create a single long feature vector. And it is connected to the final classification model, which is called a fully connected layer.

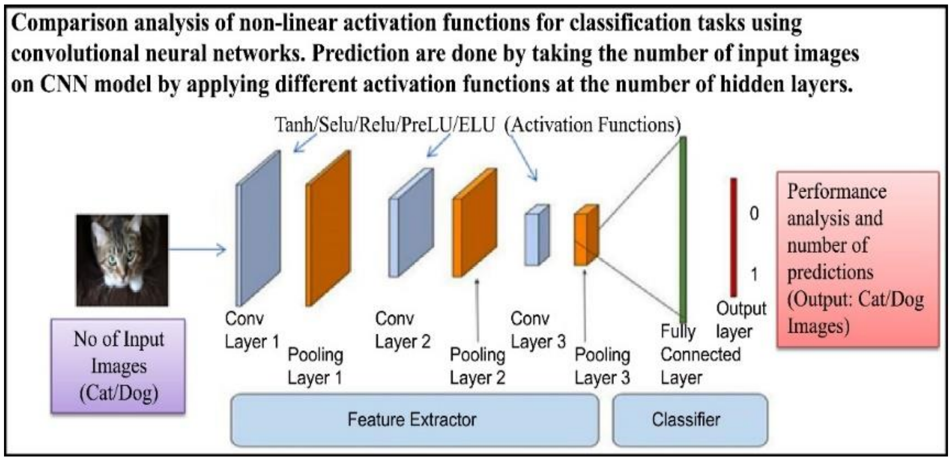




We transfer the output data of the convolution layers to 1D vector of data.

2.4.4 CNN models used for recognition

If for example, we have images of 2 different classes dogs and cats and we want to train neural network using deep learning to recognize if an image we give as a input is a cat or a dog (image the net didn't process before in the training step). we will use convolutional neural network and we will feed the net with the images of both classes mixed and labeled with 0-cat and 1-dog the convolutional layers will extract the features of each class images (features of a dog and features of a cat). For each image that we will insert as input in the training step the net will try to identify to which class it belongs, using the error calculation and updating of the weights in the neural network (as we explained before) the net will learn and improve in the specific recognition assignment she was given. In CNN the emphasis is what areas, pixels of the image are important, and which features belong to each class.


Comparison analysis of non-linear activation functions for classification tasks using convolutional neural networks. Prediction are done by taking the number of input images on CNN model by applying different activation functions at the number of hidden layers.
Tanh/Selu/Relu/PreLU/ELU (Activation Functions)
No of Input Images (Cat/Dog)
Conv Layer 1
Pooling Layer 1
Conv Layer 2
Pooling Layer 2
Conv Layer 3
Pooling Layer 3
Fully Connected Layer
0
1
Output layer
Performance analysis and number of predictions (Output: Cat/Dog Images)
Feature Extractor
Classifier

Example of classification CNN architecture- trained to identify if image is a cat image or not.

Usually in the CNN architectures, we will use Flatten layer in the end after the convolution layers and before the final classification layer

2.4.5 Loss functions

For the network to succeed in learning and improving it have to minimize the loss function- the difference between the prediction/classification for some input and the real output using this calculation the network is back propagate and updating the weights correspond to the error between the network output and the real output in machine learning we can use different loss functions for different tasks:

- Categorical Cross entropy -for classification tasks for number of classes.
- Mean Squared Error
- Binary cross entropy-used for binary prediction yes/no.

Because in our project we want the neural network to know how to identify number of movements, we will use Categorical Cross entropy as the loss function, we want to minimize.

In addition, there is different activation functions we can use in our last fully connected classification layer (we mentioned some of them previously).

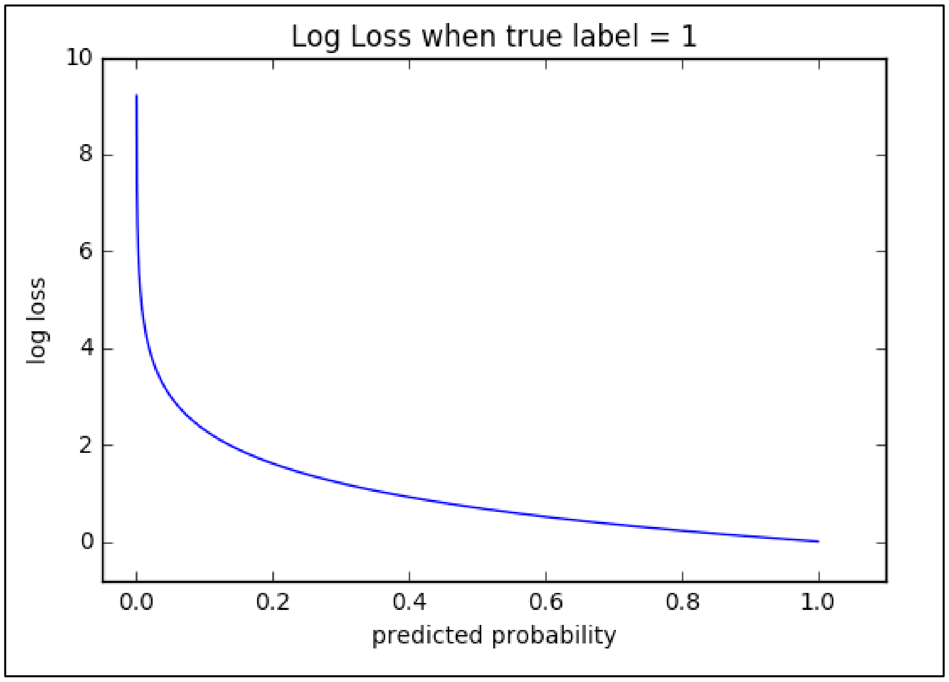
For categorical classification task, we will use SoftMax, which is the most common for this kind of tasks because it suitable for multi class tasks.

Because in our project we want the neural network to know how to identify number of movements, we will use SoftMax as the **activation function in the output layer**.

So, the idea is to insert sequence of images input to the net apply the same filters on each image using the convolution layers and extract the features of each class.

But because we try to classify movements- videos which are sequence of images we need to use another neural network system for sequence recognition-RNN epically the LSTM layers.

For our loss function we expect to see an exponential descending graph like this:

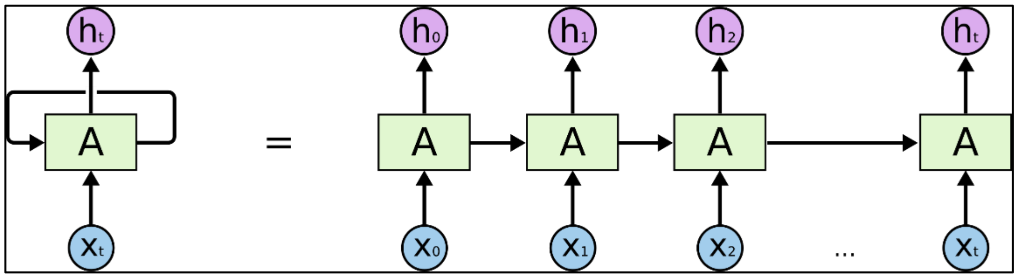


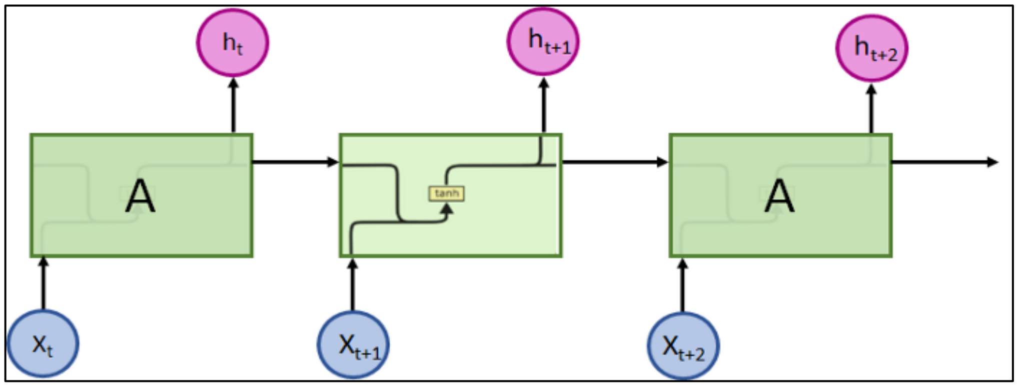
2.5 RNN-recurrent neural networks

A **recurrent neural network (RNN)** is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. Derived from feedforward neural networks, RNNs can use their internal state (memory) to process variable length sequences of inputs.

The term "recurrent neural network" is used indiscriminately to refer to two broad classes of networks with a similar general structure, where one is finite impulse and the other is infinite impulse. Both classes of networks exhibit temporal dynamic behavior.

As we have seen, CNNs do not have any kind of memory, RNNs can go beyond this limitation of 'starting to think from scratch' each time because they have memory.



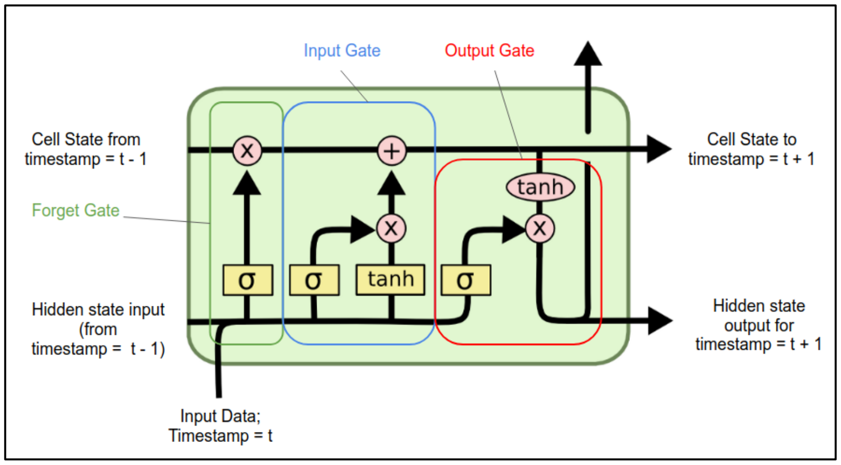


2.5.1 LSTM layers

Long short-term memory is a RNN architecture and compared to normal feed networks the LSTM has feedback connections and its capable to process not just single data points but entry data sequences (like image sequences). Every LSTM unit/cell contains entry gate, exit gate, and forget gate. The unit remember values at arbitrary time intervals (i.e., there is a dimension of time in this layer) and the 3 gates control the flow of data into and out of the cell. LSTM layers are especially suitable for prediction and identification tasks based on sequence of data. LSTM also designed to avoid the vanishing gradient problem that can occur in training of regular RNN networks. Vanishing gradient problems: the networks find only the local minimum for her loss function and stack on it value, the net can't find the global minimum which we want. The advantage of using the LSTM units compared to standard RNN unit is in the memory abilities of each cell which can forget some of the data or add new information to the data that seeps between the different time intervals-past information affects current information.

The cell-responsible for monitoring the dependences between the various elements in the input sequence. Input gate- controls how much of new data will flow into the cell. Forget gate- controls the degree of data retention in the cell.

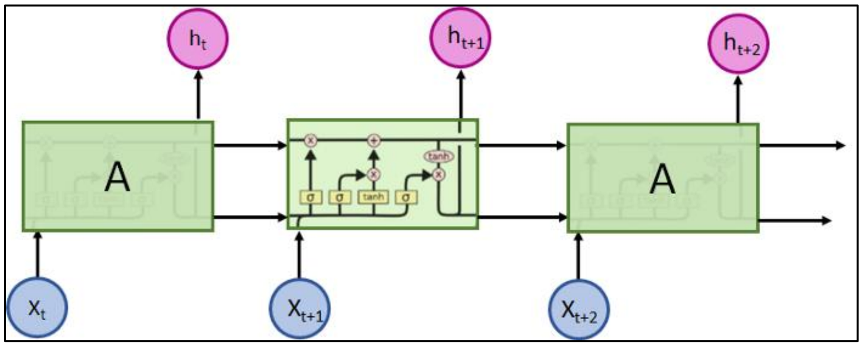
Output gate- controls how much data is used in the cell to perform the calculation for the output of the LSTM unit. There are also connections inside and outside the LSTM gates, some of which are recurrent connections.


Input Gate
Output Gate
Cell State from timestamp = t - 1
Forget Gate
Cell State to timestamp = t + 1
σ
σ
tanh
σ
tanh
Hidden state input (from timestamp = t - 1)
Hidden state output for timestamp = t + 1
Input Data; Timestamp = t

An overview of the LSTM unit

2.5.2 Sequence of LSTM units:

The input in each time cycle is represented by x_t. The activation functions are usually tanh or the sigmoid function that operates on the current input (input of the current cell) and the output from the previous cell. The output of each cell or time unit is represented by h when part of it is permeated to the next LSTM cell and part of it is the output for that time unit (for the current time cycle) h_t.



Sequence of connected LSTM units

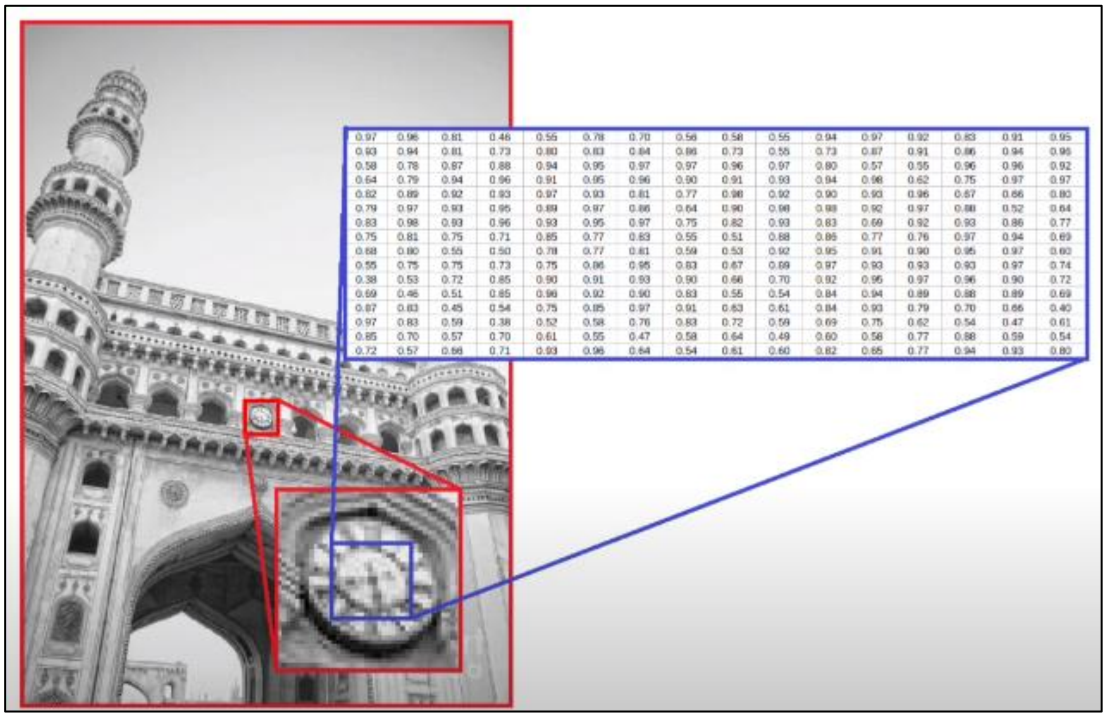
2.6 <u>**Image processing**</u>

Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image. Nowadays, image processing is among rapidly growing technologies. It forms core research area within engineering and computer science disciplines too.

Image processing basically includes the following three steps:

- Importing the image via image acquisition tools.
- Analyzing and manipulating the image.
- Output in which result can be altered image or report that is based on image analysis.

using digital computers. Its use has been increasing exponentially in the last decades. Its applications range from Digital image processing consists of the manipulation of images medicine to entertainment, passing by geological processing and remote sensing. <u>Multimedia systems</u>, one of the pillars of the modern information society, rely heavily on digital image processing.
The discipline of digital image processing is a vast one, encompassing digital signal processing techniques as well as techniques that are specific to images. An image can be regarded as a function *f(x, y)* of two continuous variables *x* and *y*. To be processed digitally, it has to be **sampled** and transformed into a matrix of numbers. Since a computer represents the numbers using finite precision, these numbers have to be **quantized** to be represented digitally. Digital image processing consists of the manipulation of those finite precision numbers.

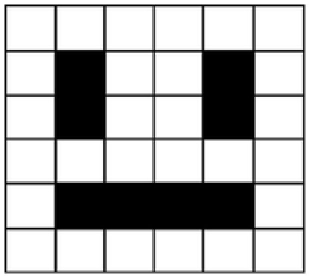


0.97	0.96	0.81	0.46	0.55	0.78	0.70	0.56	0.58	0.55	0.94	0.97	0.92	0.83	0.91	0.95
0.93	0.94	0.81	0.73	0.80	0.83	0.84	0.86	0.73	0.55	0.73	0.87	0.91	0.86	0.94	0.96
0.58	0.78	0.87	0.88	0.94	0.95	0.97	0.97	0.96	0.97	0.80	0.57	0.55	0.96	0.96	0.92
0.64	0.79	0.94	0.96	0.91	0.95	0.96	0.90	0.91	0.93	0.94	0.98	0.62	0.75	0.97	0.97
0.82	0.89	0.92	0.93	0.97	0.93	0.81	0.77	0.98	0.92	0.90	0.93	0.96	0.67	0.86	0.80
0.79	0.97	0.93	0.95	0.89	0.97	0.86	0.64	0.90	0.98	0.98	0.92	0.97	0.88	0.52	0.64
0.83	0.98	0.93	0.96	0.93	0.95	0.97	0.75	0.82	0.93	0.83	0.69	0.92	0.93	0.86	0.77
0.75	0.81	0.75	0.71	0.85	0.77	0.83	0.55	0.51	0.88	0.86	0.77	0.76	0.97	0.94	0.69
0.68	0.80	0.55	0.50	0.78	0.77	0.81	0.59	0.53	0.92	0.95	0.91	0.90	0.95	0.97	0.60
0.55	0.75	0.75	0.73	0.75	0.86	0.95	0.83	0.67	0.69	0.97	0.93	0.93	0.93	0.97	0.74
0.38	0.53	0.72	0.85	0.90	0.91	0.93	0.90	0.66	0.70	0.92	0.95	0.97	0.96	0.90	0.72
0.69	0.46	0.51	0.85	0.96	0.92	0.90	0.83	0.55	0.54	0.84	0.94	0.89	0.88	0.89	0.69
0.87	0.83	0.45	0.54	0.75	0.85	0.97	0.91	0.63	0.61	0.84	0.93	0.79	0.70	0.66	0.40
0.97	0.83	0.59	0.38	0.52	0.58	0.76	0.83	0.72	0.59	0.69	0.75	0.62	0.54	0.47	0.61
0.85	0.70	0.57	0.70	0.61	0.55	0.47	0.58	0.64	0.49	0.60	0.58	0.77	0.88	0.59	0.54
0.72	0.57	0.66	0.71	0.93	0.96	0.84	0.54	0.61	0.60	0.82	0.65	0.77	0.94	0.93	0.80

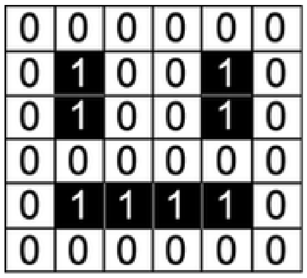
<u>Image pixel's value mapping example</u>

2.6.1 **Image representation**

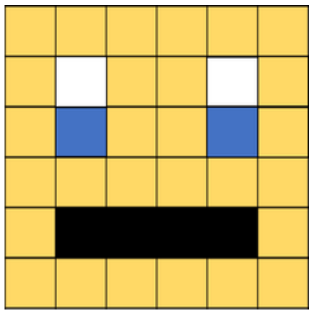
- Data in computers is stored and transmitted as a series of ones and zeros (also known as Binary). To store an image on a computer, the image is broken down into tiny elements called pixels. A pixel (short for picture element) represents one color. An image with a resolution of 1024 by 798 has 817,152 pixels.
- For the computer to store the image, each pixel is represented by a binary value. We call this representation of colors a "bit-plane". Each bit doubles the number of available colors i.e., 1-bit would give us 2 colors, 2-bits would give us 4 colors and 3-bits would give us 8 colors etc.
- In a monochrome (two color) image, like the example below, just 1 bit is needed to represent each pixel e.g., 0 for white and 1 for black.



Images are stored in scan lines. Each line is encoded from left to right, top to bottom. The image here would receive the following binary values:

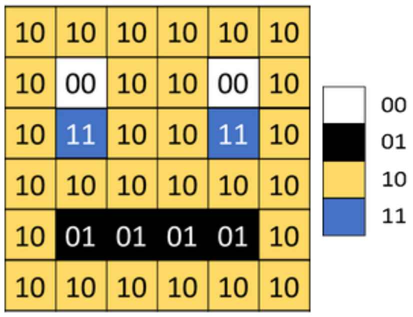


0	0	0	0	0	0
0	1	0	0	1	0
0	1	0	0	1	0
0	0	0	0	0	0
0	1	1	1	1	0
0	0	0	0	0	0



In an image that uses 4 colors, 2 bits are needed for each pixel. The following example uses two bits to store the following colors:

00 – White; 01 – Black; 10 – Yellow; 11 – Blue



10	10	10	10	10	10
10	00	10	10	00	10
10	11	10	10	11	10
10	10	10	10	10	10
10	01	01	01	01	10
10	10	10	10	10	10

⬜	00
⬛	01
🟨	10
🟦	11

Chapter 3 - Methods

On this chapter we will review our used tech and methods and describe each step of our system operation – from macro to micro.

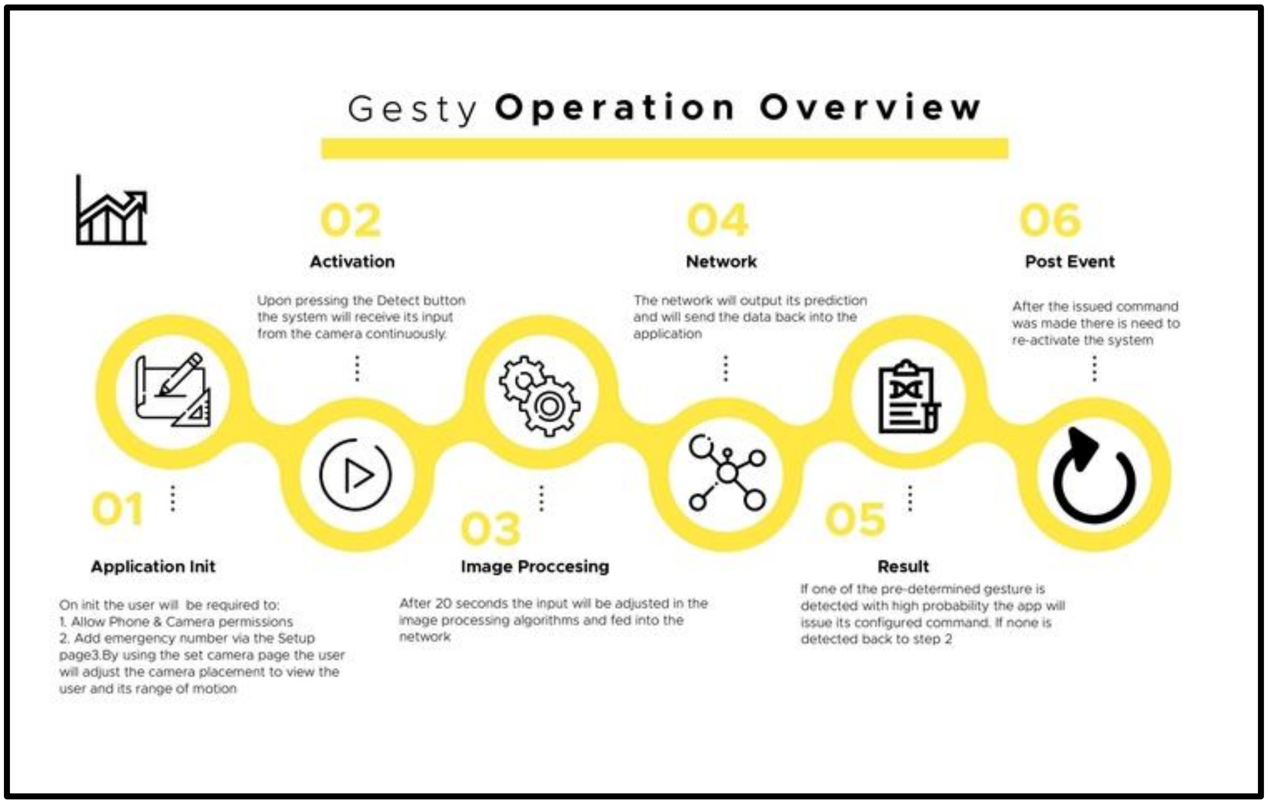
3.1 Overview –

3.1.1 To implement our model design, we used **Python** environment on Windows OS. Every sub problem we tackled with existing libraries that suits our needs. As python considered by many and in our opinion a very intuitive program language it was our best pick.

3.1.2 First, to implement our network architecture, we used **TensorFlow** and **Keras** libraries. TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications. Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Secondly, the basic data type we handle on our model is images. We searched for a way we can analyze, manipulate, and use our data to our goals. For that we used **OpenCV** library supported on Python environment. OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV provides a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

3.1.3 We aimed to mount our model on a system the designated population can use with ease and without excessive costs. According to these guidelines we chose to develop an application with a simple interface for the users to operate via **Java** programming language on **Android Studio**. Android Studio is the official integrated development environment for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. In the future we are aiming to deploy our system on all platforms.

These 3 main components are the backbone of our model implementation. The workflow operates in the following manner:



On init the user will be required to:
1. Allow Phone & Camera permissions
2. Add emergency number via the Setup page3.By using the set camera page the user will adjust the camera placement to view the user and its range of motion

Upon pressing the Detect button the system will receive its input from the camera continuously.

After 20 seconds the input will be adjusted in the image processing algorithms and fed into the network

The network will output its prediction and will send the data back into the application

If one of the pre-determined gesture is detected with high probability the app will issue its configured command. If none is detected back to step 2

After the issued command was made there is need to re-activate the system

We will now elaborate on the 3 main components of the network to gain better understanding of the displayed operation in each stage.

3.2 Network Architecture:

In this section we will present the networks we have conducted tests that helped us to achieve our final design. We will specify the architecture and the conclusions we have drawn from each.

3.2.1 **CNN Fist classification**

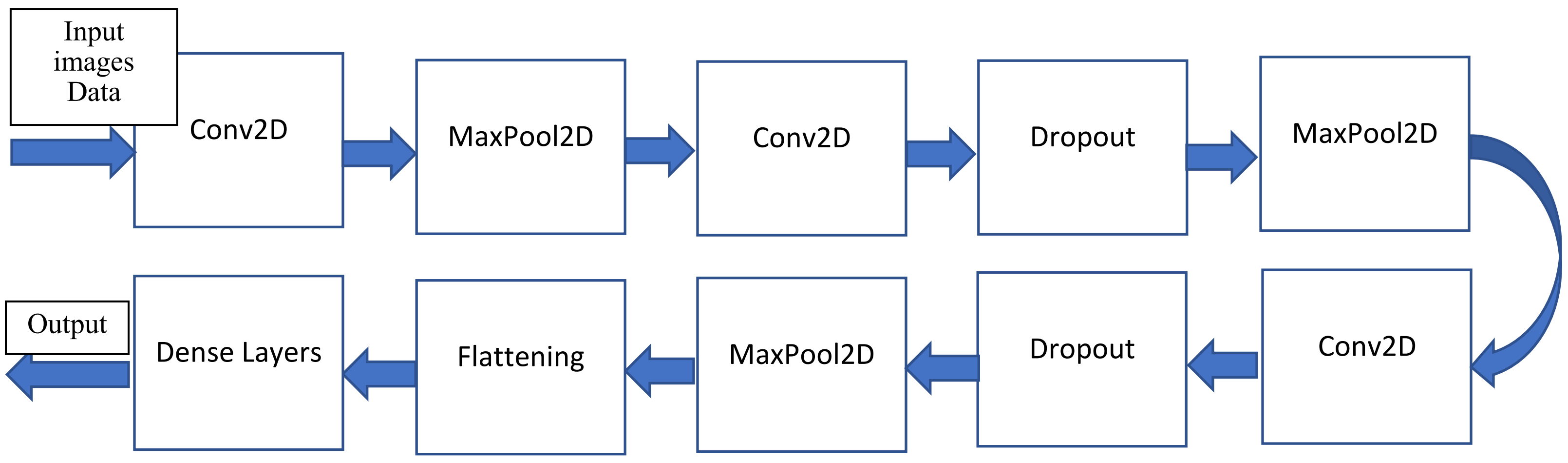
in this network we tried to train CNN to classify if a fist appeared in an image. This network served us as a preunderstanding of how CNN works and which parameters, we should use in image processing for the neural networks. The network is binary class network, which means it return 1 if fist was identified and 0 if not – meaning the output is defined by the following function:

$$Output = \begin{cases} 1\,, Fist\ detected \\ 0,\ \ Fist\ was\ not\ Detected \end{cases}$$

We used this network to determine the best parameters for the image-processing algorithm as described in sub-chapter 3.3. By implementing the network, we concluded the following parameter has the most significant effect on the results:

- Sequence length
  The size of the frames sequence we extract from each video.
- Image size
  The number of pixels in each image that we determine for each frame extracted from the video. Using this network, we tested different sizes for the images in the train size and checked which size gave as the best results in the prediction upon the test data.
- Color/black and white
  We tested for which images in the train set we get the best results in the test set- colored images or binary images.

The architecture of the simple CNN network:



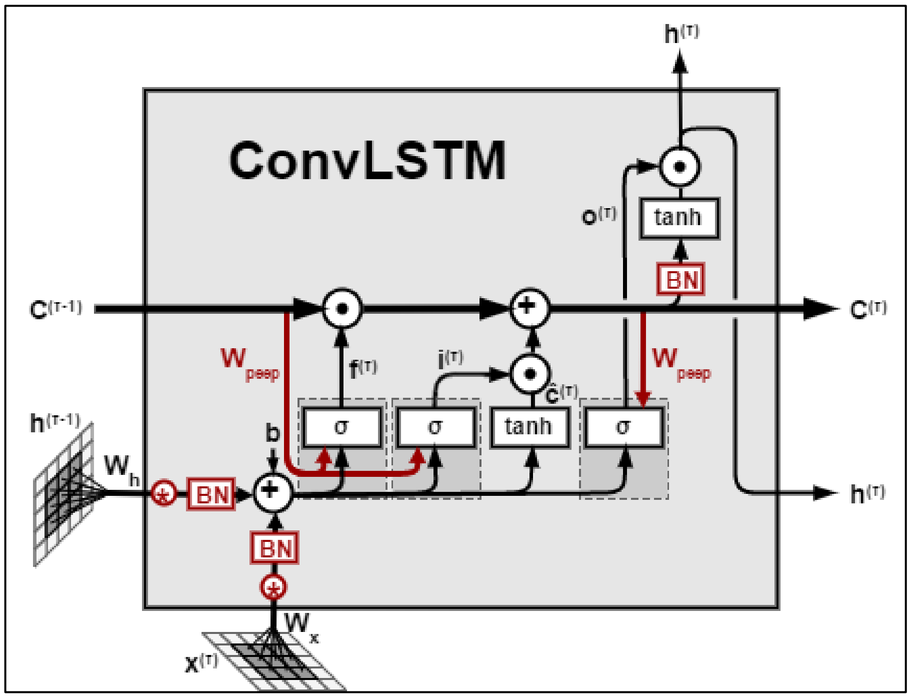
3.2.2 RNN Networks- gesture recognition in video

In this network train networks for recognizing gestures and specific movements that given in video files. This network architecture should have satisfied the gestures detection capabilities we use for our application.

The approach we chose is extract the frames from each movement video transmit each frame throw our image processing algorithm and finally insert the output images to the neural network as train data. Because we use videos as a sequence of frames/images we took the approach of using CNN+LSTM architecture combined, CNN for the image's analysis and features extraction and LSTM (Long short term memory) for dealing with sequences of data that have effect of the time dimension.

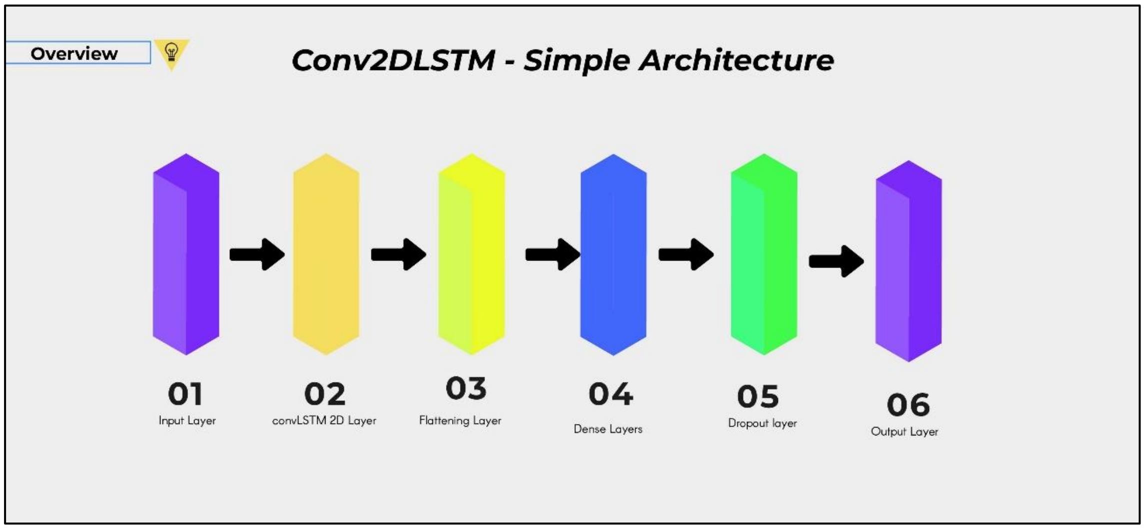
This combination of methods and architecture is implemented in ConvLSTM2D layers. ConvLSTM2D is **an implementation of paper Convolutional LSTM Network**: A Machine Learning Approach for Precipitation Nowcasting that introduces a special architecture that combines gating of LSTM with 2D convolutions.

ConvLSTM replaces matrix multiplication with convolution operation at each gate in the LSTM cell. By doing so, it captures underlying spatial features by convolution operations in multiple-dimensional data.



The calculations that each convLSTM cell apply on the data

3.2.1.1 Simple RNN architecture using convLSTM2D

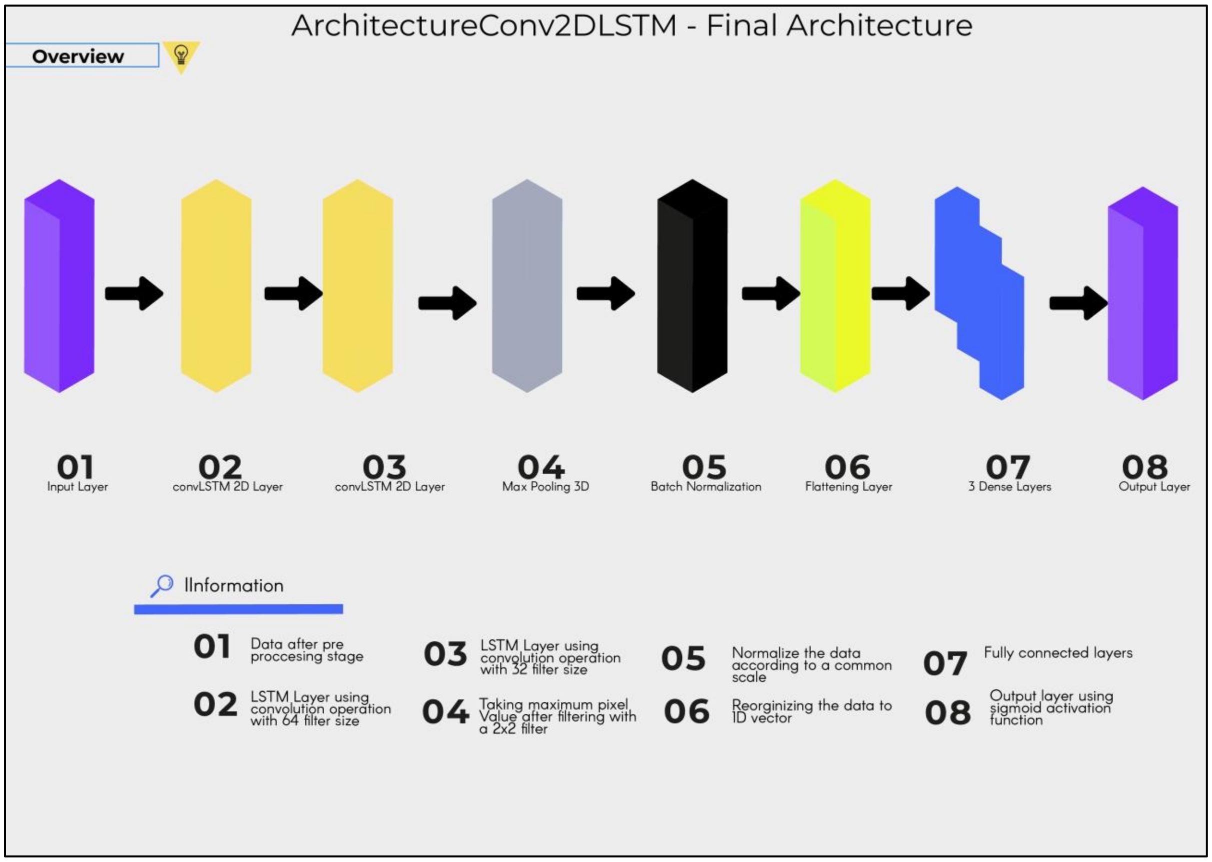


Training on this simple architecture caused the network to over fit: overfitting problem- big percentages of success in prediction during the training process on the validation set and low percentage of success on the test data. The network over fit on the training data and doesn't really learn the characterizations of the movements and gestures.

In order to prevent overfitting, we tried using different methods and layers in different modes:

1. Batch normalization - explain
2. Max pooling
3. Layer weight regularizes
4. Increasing Dropout rate
5. Decrease learning rate
6. Determine training early stopping according to the validation loss improvement.

3.2.1.2 The new network architecture we used as a base architecture



01 Data after pre proccesing stage

02 LSTM Layer using convolution operation with 64 filter size

03 LSTM Layer using convolution operation with 32 filter size

04 Taking maximum pixel Value after filtering with a 2x2 filter

05 Normalize the data according to a common scale

06 Reorginizing the data to 1D vector

07 Fully connected layers

08 Output layer using sigmoid activation function

The layers and parameters of the network we tested:

- batch sizes:
  tested sizes: 4, 8, 16, 32
- learning rate:
  tested rates: 0.01, 0.001, 0.0001
- Number of dense layers + size of each layer (number of neurons).
- Dropout rate:
  tested rates: 0.4, 0.5, 0.6
- Number of MaxPooling layers+ the pool size of each layer:
  - 1 layer after all the convLSTM2D layers VS 1 layer after each convLSTM2D layer.
  - We tested pool sizes: (1,2,2), (1,3,3), (1,3,3).
- Numbers of convLSTM2D layers + number of filters
  - We tested 1,2 and 3 convLSTM2D layers
  - Number of filters checked in different order: 32, 64, 128.
- Number of batch normalizations layers
  - 1 layer after all the convLSTM2D layers VS 1 layer after each convLSTM2D layer.

Each parameter and layer tested by setting the other parameters and changing the desired parameter.

For Each network, we used the same data:

Training data that contained:

- Vertical fist movement - 586 videos 3 seconds each.
- Head turn movement – 607 videos 3 seconds each.

Test data contained:

- Vertical fist movement – 48 videos 3 seconds each of completely different people.
- Head turn movement – 108 videos 3 seconds each.

In the training process, we use splitting the training data to 0.8-training set and 0.2- validation set which gave us the best result due to our relatively small data set.

Results

❖ The results for different **Batch sizes** are:

Batch sizes	Train - Fist vertical	Train - Head Turn	Test- Head Turn	Test- Fist Vertical
4	93%	92%	72%	71%
8	95%	90%	77%	77%
16	95%	91%	86%	77%
32	97%	89%	83%	68%

We can see that batch size 16 gave us the most balanced results.

❖ We tested different **learning rates** for the network- 0.01, 0.001 and 0.0001.
We got the best result with slower learning rate (0.0001) the training time took more time, but we got more accurate results in the prediction on the test set.

❖ Results for different number of **DENSE layers** we tested:

- We noticed that for more than 3 Dense layers the complexity of the network it's to high relative to our Data size, same about using too many neurons in each network (more than 256 for the max size layer).
- We tested using 2 Dense layers – first one with 256 neurons and the second with the number of our classes (2 because we try to characterize 2 gestures) as required and compared it to 3 dense layers architecture – first one with 256 neurons second one with different sizes last one of course with 2 neurons for the decision layer.

Number of Dense layers (including the decision layer)	Train-Fist vertical	Train- Head Turn	Train- Head Turn	Train-Fist vertical
2	87%	86%	72%	71%
3	95%	91%	86%	77%
>4	High complexity	High complexity	~60%	~60%

Using 3 dense layers sizes- 256,32 and 2 respectively to their order gave us the best results.

❖ **Dropout rate** – tested different dropouts from 0.2 to 0.7- 0.6 gave us the best results, less than that the network over fit fast due to our relatively not divorce data. 0.6 dropout rate caused the network to train on more epochs without over fit which led to better results on the test set.

❖ **convLSTM2D** different number of layers and sizes results:

- we tested different convLSTM2D layers configurations. 1,2 and 3 layers with different sizes the results we got:

Number of convLSTM2D layers	Train- Fist Vertical	Train- Head Turn	Test- Head Turn	Test- Fist Vertical
1	91%	90%	73%	68%
2	95%	91%	86%	77%
3	97%	85%	76%	70%

We saw from the results we got that using 2 convLSTM2D layers its optimal for our network more than 2 raised the complexity level and 1 layer wasn't enough for satisfying features extracting from the video frames.

- In each convLSTM2D layer we used we added **weight regularizes-** l2 with parameter of 0.01 **that** prevent the values of the weight from exploding and getting to big values- it helped as with preventing overfitting.
- Regularizes allow us to apply penalties on layer parameters or layer activity during optimization. These penalties are summed into the loss function that the network optimizes.

- Regularization penalties are applied on a per-layer basis.
- L2 regulaizer characterized in the formula:

L1 regularization on least squares:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \boxed{\lambda \sum_{i=1}^{k} |w_i|}$$

L2 regularization on least squares:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \boxed{\lambda \sum_{i=1}^{k} w_i^2}$$

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

❖ Using max pooling layers and results:

- We tested our network architecture with 2 max pooling configurations:
  1. Using max pooling after each convLSTM2D layer.
  2. Using 1 max pooling after all the convLSTM2d layers.

  Also, we tested different pool sizes: (1,2,2), (1,3,3), (1,4,4)

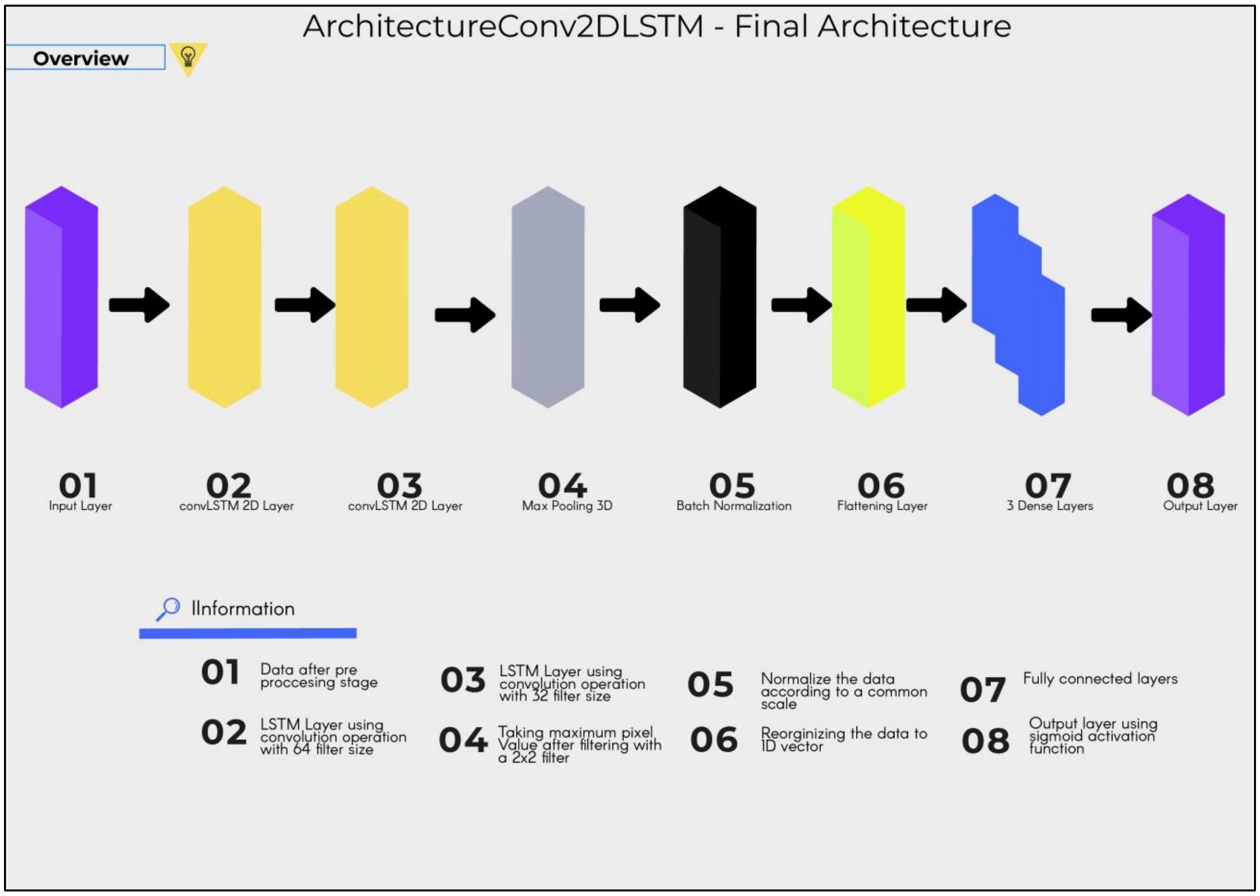
Maxpooling configuration and size	Train- fist vertical	Train- Head turn	Test- Head turn	Test-fist vertical
(1,2,2) pooling after each convLSTM2D layer	98%	93%	84%	73%
(1,4,4) pooling after the convolutions	96%	93%	81%	80%
(1,3,3) pooling after the convolutions	96%	92%	80%	75%
(1,2,2) pooling after the convolutions	95%	91%	%86	77%

- using 1 max pooling layer after all the convolution layers gave us the best and the most balanced results.

❖ Batch Normalization:

- We tested 2 Batch normalization configuration following research we did about the use of batch normalization we follow that these layers work best pairing with the MaxPooling layer. After we decided on using 1 MaxPooling after the convolution, we added the Batch Normalization layer right after it that gave us better results than placing it after each convolution layer. Putting the Batch Normalization layer after the convolution layers apply the normalization on the output of the convolutions before the output inserted to the dense layers.

3.2.1.3 Our Final Network architecture –


ArchitectureConv2DLSTM - Final Architecture
Overview
01 Input Layer
02 convLSTM 2D Layer
03 convLSTM 2D Layer
04 Max Pooling 3D
05 Batch Normalization
06 Flattening Layer
07 3 Dense Layers
08 Output Layer
IInformation
01 Data after pre proccesing stage
02 LSTM Layer using convolution operation with 64 filter size
03 LSTM Layer using convolution operation with 32 filter size
04 Taking maximum pixel Value after filtering with a 2x2 filter
05 Normalize the data according to a common scale
06 Reorginizing the data to 1D vector
07 Fully connected layers
08 Output layer using sigmoid activation function

The results we got are:

Train- Fist Vertical	Train- Head Turn	Test- Head Turn	Test- Fist Vertical
95%	91%	%86	77%

3.3 <u>**Data and Image Processing**</u>

On the image processing stage, the are several variables we will discuss in this sub-chapter. On each of them we had to optimize. As we are fairly new to any of these fields our main method was design a solution and optimize it in a trial-and-error manner.

3.3.1 Data representation

The basic data structure in this chapter is an image. As we can see in chapter 2.6 an image can be represented as a matrix. Our convention handling the images in our code was via NumPy library which allowed us to review our data and manipulate it to our goals.

The second data type which we were handling was videos, which is a sequence of images. Our goal is to predict whether a gesture was made in a fixed sized sequence if frames. By adding the video object, we are adding another domain to problem – the time domain. We need to add that parameter to our testing in each step.

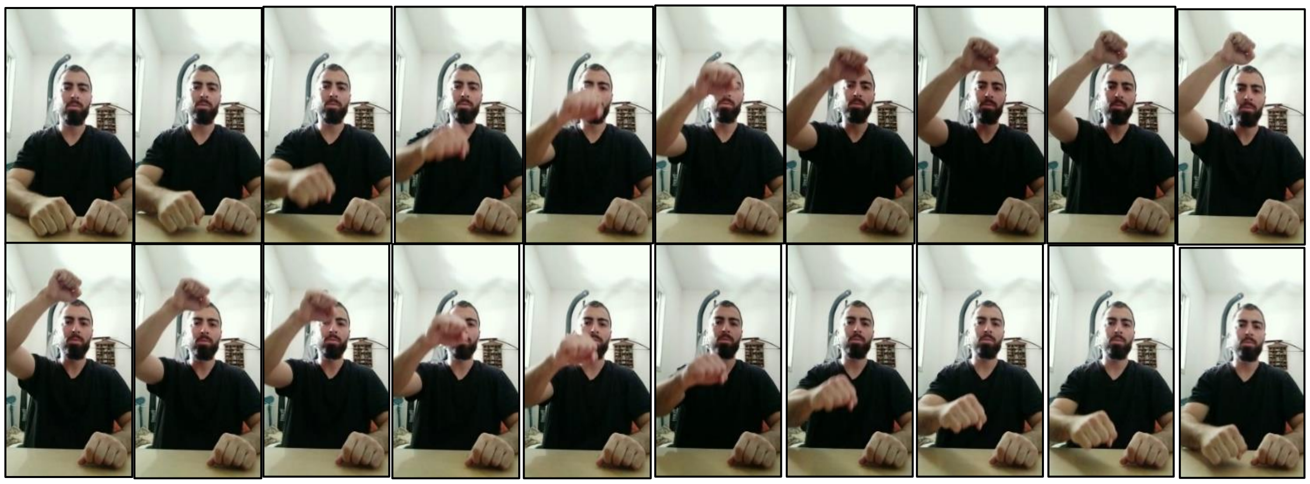
3.3.2 Data Gathering:

The data was created by us via OpenCV library tools in a python environment. The raw data was provided by our friends and family. Note, the testing data contains movement from different people used to form the training data.
As we have researched, the amount data has direct correlation with the result of the network. To add more we wrote a script that flips the video starting point to the end. By doing so we have doubled our data and added a different kind of motion as our gestures are periodic.

The following sequence of images displays the gestures we have defined:

1. Head movement:
    - Training data: 607 videos, 3 seconds each.
    - Test data: 108 videos 3 seconds each

2. Vertical fist movement:
   - Training data: 586 videos, 3 seconds each.
   - Test data: 48 videos, 3 seconds each.



3.3.3 Data optimization

We faced several optimization problems.
We needed to optimize the representation of our data structure to serve our goal– to “help” our network to learn the “right” features. There are multiple factors that we needed to consider and balance between results and model complexity with our computing power at hand.

- Step 1 – Size:
  First, we needed to decide on the size of the image. While bigger size will represent the features in the image better it will also cause a longer training time. For example, training the same number of images on our basic CNN network while only changing the dimensions of the pixel size between 640x640 to 64x64 differs more than 15 hours. To balance between the system complexity and sufficient feature representation we decided to conduct a test to compare the network between different image sizes to see the impact of the size reduction:
  1. 640x640
  2. 256x256
  3. 64x64

640x640 compared to 64x64 image resolution

resizing each frame image to 64x64 pixels size- reducing the number of pixels in each frame will help as reduce the model complexity and will help in improving the models result, we tried different size and got best results with 64x64 also we saw that it's a common size for using in CNN/RNN models.

- Step 2 – Color:
  We wanted to discard any irrelevant parameter in the image. Irrelevant in our case means it will not help the network to detect a movement was made. So, we decided to work with greyscale images. To verify the impact will not be significant on the result we did a comparison on a basic CNN to detect if a human fist is in the image. We sampled each version x images and fed them into a training process inside the same network with the same parameters When the training was done, we fed the same test sample inside both models learned in the training process and saw the following results:

  - Black and white binary images gave us better results than colored images due to the simplicity of the data when it represented in binary.

- Step 3 – Sample Rate:
  Our main goal is to train out network to detect pre known movement – so the sample rate we decide may have a great impact on the success rate of the detection. A rate too low may result in a bigger training sample set to contain the desired movement. To high may not represent the movement accordingly and will not differ from the other movements. To find the "sweet spot" we worked in a trial-and-error method. Isolating the entire system parameters and only change the rate on which we sample the images from a given set of videos. To conduct this test, we used our first simple CNN network.
  We sample 4 different rates:
  1. Take every 10th frame
  2. Take every 2nd frame
  3. Take every 5th frame
  4. Take every 3rd frame

  - The best results we had was using every 2nd frame.

- Step 4 – Substruction:

The 4th step we took to optimize out data was driven by the idea to eliminate even more unwanted features in our data. To do so we used a simple physical concept:

Subtracting the location of an object on time t_2 from the location of the same object on time t_1 while t_2 > t_1 would indicate if a movement occurred.

$result = later_{frame} - previous_{frame}.$

On our problem domain we deduce from this basic concept the following: If we sample x images from a video of a given movement and subtract each adjacent frames the result will be an image which display the difference between the two – meaning only the pixels that has different values between the two frames will have a value greater than 0.

Only where there was a movement in the frames, we will have values in the result after the subtraction of every two adjacent frames because where there was no movement the pixels didn't changed from frame to frame so subtracting in this place will give us 0. To do so we used NumPy library to represent each frame as a matrix and the substruction works according to matrix subtraction conventions:

- Step 5 – Threshold:

As we can see on the example above, the result was good but not optimal – a noise was added on the process. We could not optimize the other factors such as sample rate and image size so minimize the noise consistently to justify the subtraction operation.

To discard the redundant noise, we relayed on the fact that the added noise is represented as a group of pixels which their value is lesser then the pixels value that represent the gesture.

Researching OpenCV library resulted in finding a method that serves our goal:

cv2.THRESH_BINARY:

$$\text{Dst}(x, y) = \begin{cases} Max\_Value_{*} , & if src(x,y) > thresh \\ 0 , & otherwise \end{cases}$$

- Max_Value in our case, as our image is grey scaled, is 255.

We have conducted a test to review the threshold value, thresh, which will be optimal to our needs. To do so we applied different values on our image data set, S, and review the result by the following rule:

$$Image = \begin{cases} 1 , & if\ the\ gesture\ representaion\ is\ shown \\ 0 , & othwerwise \end{cases}$$

Note, the results have a setback. Not in all images a sequence is shown even with 0 thresh value. A gesture has different stages and there are stages which we sample 2 images with no movement was made. We have reviewed 3 different threshold values:

1. 30
2. 40
3. 50

- The test results were better using the threshold value 40.

- illustration of the binary conversion for the subtracted frames of the "vertical hand movement":

- <u>Step 6 – Isolate the gesture representation:</u>

  By reviewing the results of a large data set we noticed several recurring features. On the result image we had different components separated by areas with pixel value of 0. Using the threshold method did not discard the noise completely.

  Following our assumption, whom we tested manually on about 600 images, that the main movement component displayed on the result image was always bigger than the added noise. By this assumption we thought about implementing a known algorithm –Connected Components in an undirected graph.

  In graph theory, a component of an undirected graph is an induced subgraph in which any two vertices are connected to each other by paths, and which is connected to no additional vertices in the rest of the graph. Components are also sometimes called connected components.

  Connected component labeling (also known as connected component analysis, blob extraction, or region labeling) is an algorithmic application of graph theory used to determine the connectivity of “blob”-like regions in a binary image.

The algorithm works as followed:

```
Algorithm 1: Finding Connected Components using DFS
Data: Given an undirected graph G(V, E)
Result: Number of Connected Components
Component_Count = 0;
for each vertex $k \in V$ do
    Visited[k] = False;
end
for each vertex $k \in V$ do
    if Visited[k] == False then
        DFS(V,k);
        Component_Count = Component_Count + 1;
    end
end
Print Component_Count;
Procedure DFS(V,k)
Visited[k] = True;
for each vertex $p \in V.Adj[k]$ do
    if Visited[p] == False then
        DFS(V,p);
    end
end
```

On our domain we can project the algorithm is projected in the following manner:

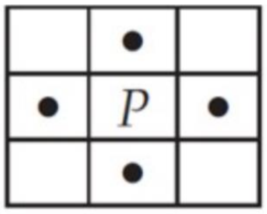
1. Define the pixels as vertices.

$$|V| = 4096$$

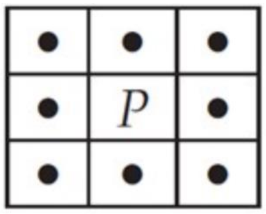
2. An edge is defined as followed:

$$E = \{(u, v) \mid u, v \in V \ AND\ u\ is\ adjecent\ to\ v\ AND\ u.value > 0\ AND\ v.value > 0\}$$

Adjacent pixels are defined by the following placement:



Pixel P has 4 neighbors

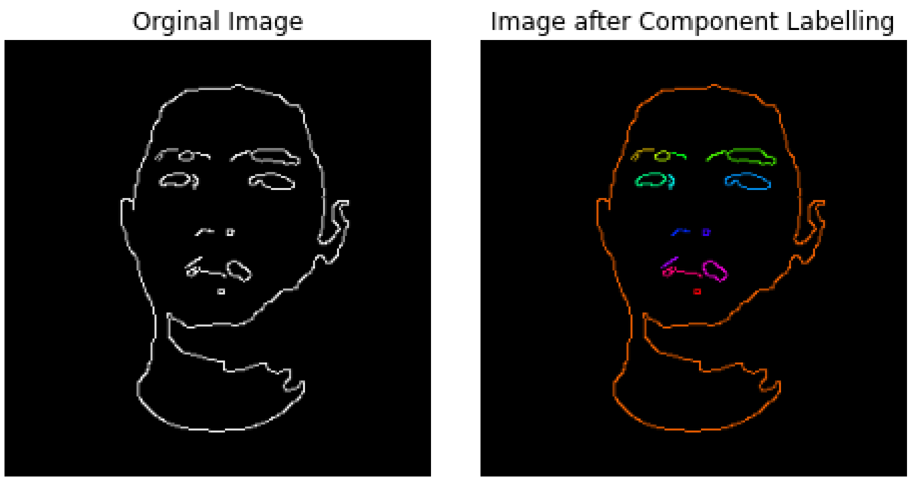


Pixel P has 8 neighbors

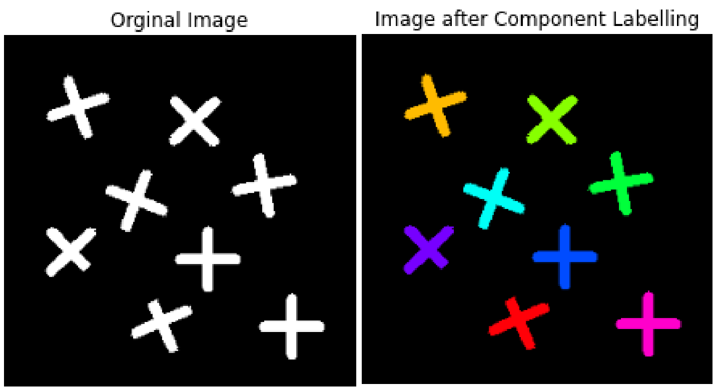
Upon setting the conventions above we searched for an optimal way to implement the algorithm and found one on OpenCV library – cv2.connectedComponentsWithStats. The method input is an 8-bit single-channel image and the user choice between 4 or 8 neighbor flag (see illustration above). It also provides several stats for each component such as: area, centroid, x value, y value height and more.

For our needs we used the 8-neighbor flag and under our assumption we've iterated on the components outputting the one with maximum area on a blank image.

Illustration of the connected components abilities and uses:

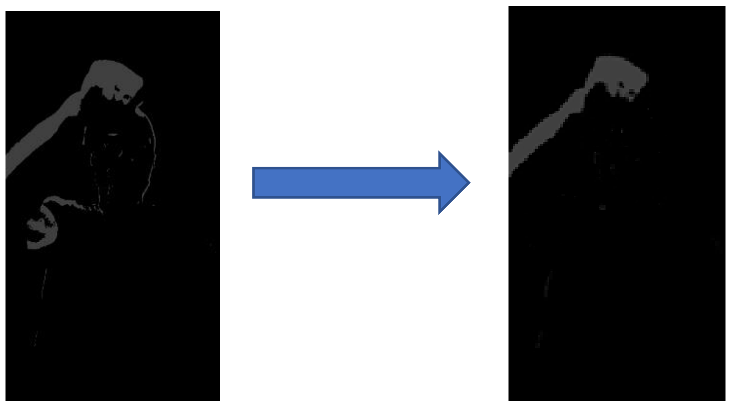


- Applying cv2 connectedcomponentswithstats() function to each frame:
  We used the idea of finding the connected components in our binary images for noise filtering.After we apply the connectedcomponentswithstats() function and got all the connected components in the image we took only the biggest (by area) connected component and delete all the others, in this way we make sure we left with only the real movement representation in each image.



Noise that created by small movements like background movements, etc. will be deleted, and when we propagate the images as train data for our neural network, these little noises will not create confusion in the learning process.

Example of image filtering keeping only the biggest connected component:

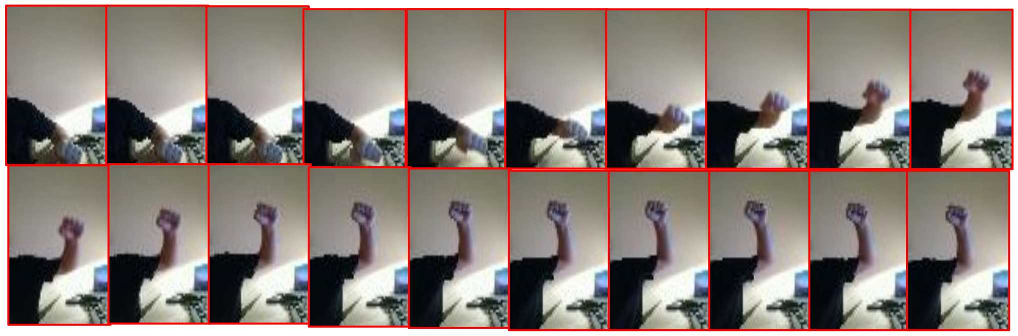


- Final stage in our images data-preparing algorithm: taking the sequence of subtracted and filtered images and insert it to array of frames for each video, each frames array will be inserted to our neural network as input.
  Each frame in each array is labeled- what movement it belongs to, in that way the network will try to learn the features check if it predicted the right movement and back-propagate for weights updating and results improving.

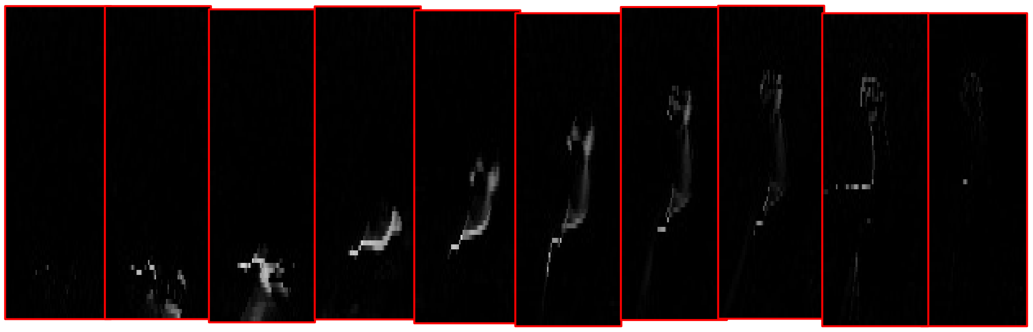
3.3.4 Summery:

By following the 6 steps above we have prepared our data in a way that will ease our network process and will prevent learning of unwanted features. The step optimization made helped to reduce the loss of relevant feature in the data preparation operation. Here is a single sequence on the different stagers of our algorithm:

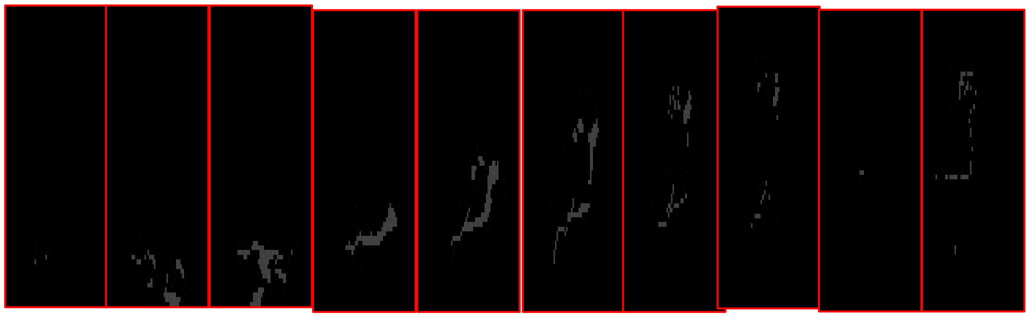
Original sequence



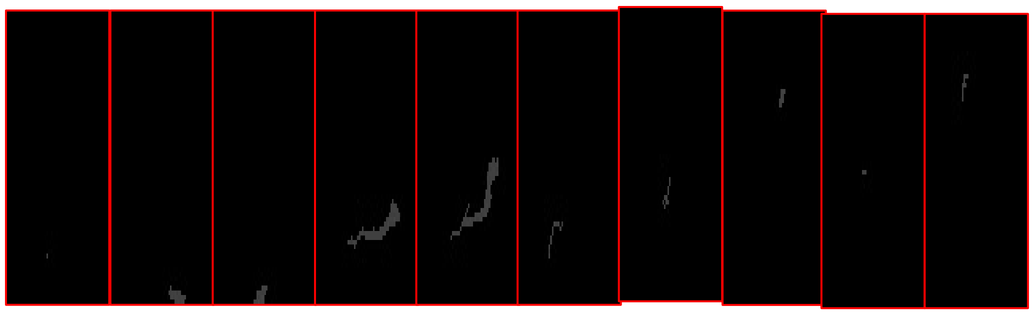
Sequence after substruction



Sequence after applying threshold



Original sequence



3.4 **Application**

The application we have developed aim to help operate and monitor the user use of our system. The main benefits are that anyone with a phone, which is very common product these days, can use our system. As we mentioned on chapter 2, people who suffer severe disabilities face numerous hardships – financials as well. By accessing a system on such an everyday product will help to reach more people in need.
In this chapter we will review 3 main elements which we had in front of us during the development.

3.4.1 Functionality

We implemented the following functionality in our application:

- **Detection**: By pressing the Detect button in the detection page the system will be activated. On activation the application will sample images from the camera and continuously feeding them into the network after processing them as explained. Upon detecting a gesture, it will execute the pre-configured command. The system will continue its work until a gesture is detected.
- **Supported Gestures**: Vertical Hand movement, Head turn movement.
- **Configuration**: We enabled the user a possibility to insert a number for emergency in the Setup page. By doing so, upon detecting the movement the application will issue a call to the inserted number. If no number is inserted the default is to call to emergency services.
- **Setup:** As our audience cannot operate the phone by themselves, a Camera Setup page is enabled. Before activation it is recommended that the phone will be placed in the recommended manner (will be detail on the Setup sub-chapter).
- **Information:** In the Help page, the user can review the supported gestures of the system and its related action.

3.4.2 **Setup:**

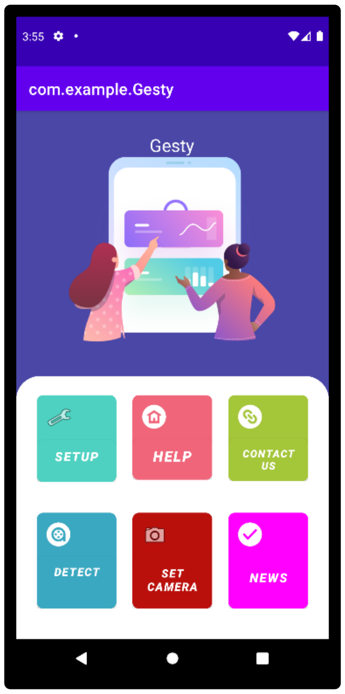
There is a big importance to the setup process. As the user cannot adjust the phone \ computer it is recommended to follow the recommendation in this chapter.

- First, enter the Setup page and insert an emergency number and press the submit button.
- Second, with the aid of the Set Camera page adjust the phone \ webcam to face the user in a manner that its body is seen from the waist up. It is recommended that the camera will view the entire movement range of the user to avoid inaccurate detection.
- After the setup process, Press the Detect button in the detection page to activate the system.
- Upon detecting “Bathroom” gesture an alarm will set of indicating the event. To turn the alarm off press the Stop button.
- After each detection the system will stop its operation and will require re-activation as explained on step I

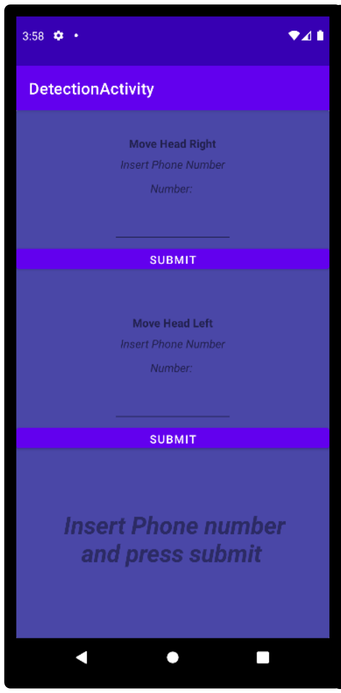
3.4.3 <u>Interface</u>

In this following sub-chapter, we will display our application interface.

- <u>Main page</u>



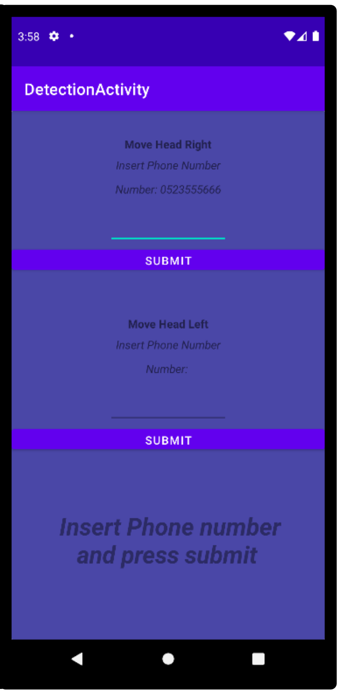
- <u>Setup Page: Before and after insertion</u>



Move Head Right
Insert Phone Number
Number:

Move Head Left
Insert Phone Number
Number:

Insert Phone number and press submit

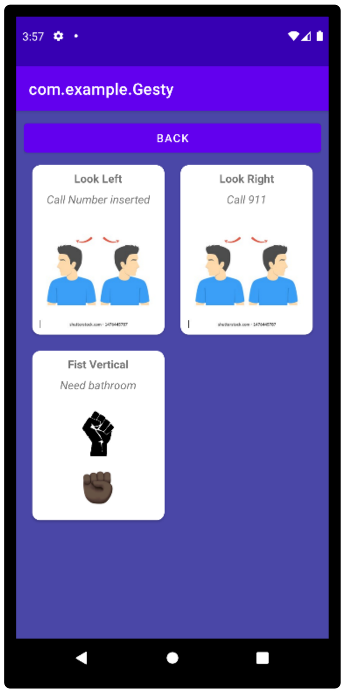


Move Head Right
Insert Phone Number
Number: 0523555666

Move Head Left
Insert Phone Number
Number:

Insert Phone number and press submit

- <u>Help Page</u>

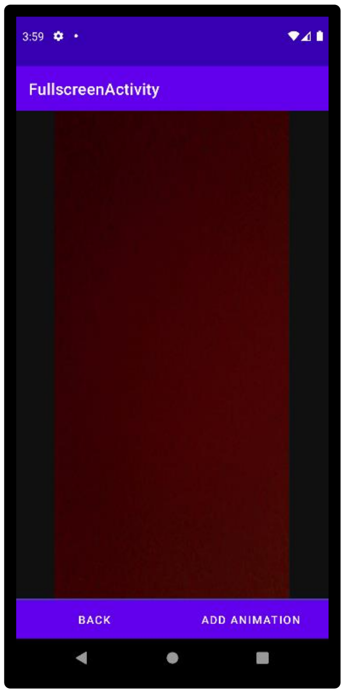


Look Left
Call Number inserted

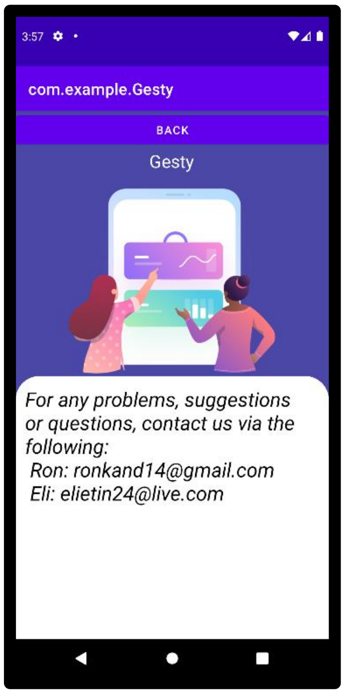
Look Right
Call 911

Fist Vertical
Need bathroom

- <u>Set Camera Page</u>

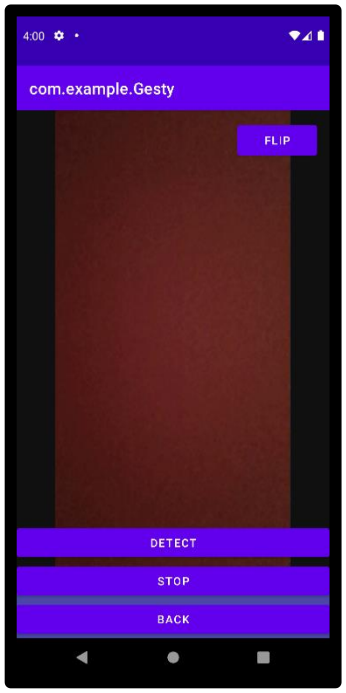


- Contact us Page



For any problems, suggestions or questions, contact us via the following:
Ron: ronkand14@gmail.com
Eli: elietin24@live.com

- Detection page



Note:

The news page is not yet implemented and will be discussed in future work chapter 5.

Chapter 4 – Results

In the following chapter we will display our conducted tests result that helped us to design our final network. The results order will be aligned with chapter 3 and ending with our final network design result.

4.1. Size test (w.r.t 3.1)

The optimal size we chose was an image of size 64x64.
The choice was made after observing no decrees in our network accuracy on test sample. By reducing the image size, we reduce our system complexity and allowing fast training process. Moreover, it reduced the network run time. By using the predict method called from the application the images are pre-processed with our script and then process by our network model – smaller images result in faster run-time in our application. One more un-tested observation is that in smaller the images, the noise has less representation in them. The gestures we train preserve their representation as they are simple movements on a 2 dimensional plain. We can conclude that inserting a more complex gesture that composed of radial movement or 3 dimensional movements we will need to change our model parameters for good results.

4.2. Color test (w.r.t 3.2)

As our system is gesture detection and we subtract the network, we expected the color will have no effect on the network prediction success rate. And as expected, the test we conducted assured us this is the case. There was no different between the accuracy results of our network on the same test sample. After progressing in our implementation, the Connected component algorithm used cancel the color representation of the image, so it aligned with the result.

4.3. Sample Rate test (w.r.t 3.3)

We have concluded that the best result was taking every 2nd frame. This sample rate gave us the best representation to the movement after the subtracting made on the next step. By manually iterating on the result images, we saw the gesture has better representation. For example, we can see the following images that represent the same stage of the fist gesture and that differ only by the sample rate.

4.4. Threshold test (w.r.t 3.4)

The best threshold value for our goal is 40. Meaning, every pixel below the value of 40 will be decreased to zero and any equal or above will be increased to 255. We have witnessed that value of 50 discard part important to gesture and we are losing to much data. Picking 30 did not serves its purpose discarding the noise.

4.5. Subtraction test (w.r.t 3.5)

After the steps made, we needed to test if our pre-processing stage is contributing to our goal – perhaps leaving the images as is achieve similar results?
The test we have conducted showed that using the subtraction pre-processing gives a big improvement in the accuracy value on the test sample. While training the images without feeding them into the pre-processing stage resulted in a 55% accuracy the subtraction algorithm boosts the accuracy to 73%.

4.6. Network accuracy (w.r.t 3.6)

◈ **Image processing parameters**

Size	Color	Sample Rate	Threshold Value
64x64	Grey Scale	Each 2nd Frame	40

◈ **Network**

Train- Fist Vertical	Train- Head Turn	Test- Head Turn	Test- Fist Vertical
95%	91%	%86	77%

Chapter 5 - Future Work

In this chapter we will review our future work suggestion. The improvements to the system vary in several domains and the chapter will be divided in this manner.

5.1. Network

5.1.1 Improve design

As one can witness reading out work process, we have started with almost no knowledge regarding the design and behavior of the python implementation. Moreover, our theoretical knowledge was not sufficient to design beforehand a valid network. The main leaning process was trial and error. The network architecture needs to be improved and even expanded to several network, each supports in different type of movement.

5.1.2 Change input size

Design a network we smaller input without losing accuracy. Review 5.3.1.a for the reason. Note, as the input size will change it affects the network accuracy (tested). The design needs to be changed to support a smaller input and the entire parameters needs to be re-tested in this domain. See (5.3.1.a) for more information.

5.2. Data

5.2.1 Data gathering

As we mentioned, the accuracy of the network has a direct correlation to the size of our training data. To improve our networking even more we need to gather more videos of the gestures.

5.2.2 Add gestures

We only trained our system to support 2 gestures. We wish to expand our gestures pool to allow our system to assist more people I need. We advise to plan the next gestures according to the needs of specific people groups after conducting more research.

5.2.3 Self-learning Algorithm

Create algorithm that will automatically create and find data for specific movement and train the network on it.

5.3. Application

Our developed application has basic features and unoptimized functionality. For the near future we advise to work on the following:

5.3.1 Faster feeding time

As we need to detect movement of a subject who suffer from physical disability, we wish to shorten the image sequence fed into the network. By doing so the detection functionality run time will be much faster and allow a more comfort user experience.

5.3.2 Notification page

By looking from the user's family angle, it is important to being able to review the daily request the user has made. By creating a notification page that will save the log of the system in a readable manner can serve the purpose.

5.3.3 Minimized view

To allow the user to operate the phone as new feature will be added it is crucial to extend the development to support the App operation while minimized on the phone.

5.3.4 IOS support

The application needs to be supported on IOS system. As we developed in an android environment it cannot be deployed on Apple devices. There are various converters from apk file to IOS but will need to test if the functionality remains after the conversions.

5.4. Research

As we mentioned our work was aimed by experience and by reading basic research on the field. Although we contacted several organizations to direct out system to help disabled people it is crucial to continue the development hand in hand with an organization who has access to the people in need and data regarding the state today.

Chapter 6 - Final conclusions

- We saw that there is full correlation between how big and diverse the data is to how the network work well. By doubling our dataset, we saw an improvement of ~10%.
- Interesting note: because most of our training videos was videos of men, we noticed that the network not functioning well when predicting movements of women. We are assuming this behavior is due to the hair movement, causing noise.
- Our image-processing algorithm increase's our final network accuracy by 30%.
- We had to find that right balance between good results and small complexity due to the lack of computing resources. We assume that with better processing power we would achieve better results as our images could be in higher resolution.
- Drawing conclusions from the simple CNN fist recognition network we created helped us understand what parameters and what values to use in our RNN network. Therefore, when we build our RNN network we perform different tests on the network layers and parameters but the basic parameters like color, image size, threshold etc. we already knew.

Chapter 7 - Acknowledgments

We would like to express our gratitude to our supervisors who led, helped, and guided us during the making of the project: **Professor Tzvika Lotker** and **Doctor Adi Makmal.**
For allowing us to pursue an idea that started as a thread of thought and guided us to evolve it to a tangible product. During the last year we have acquired much needed skills for our future endeavors.

We wish to extend my special thanks to the IDF Disabled Veterans Organization Shefela District for the willingness to assist and share the needs in the community to guide us in our development. Special gratitude to the chairman, Dekel Sharoni, who didn't hesitate to meet with us and share his resource to promote the project.

References

Scientific paper and researchs:

- [1 A Comprehensive Guide to Machine Learning Soroush Nasiriany, Garrett Thomas, William Wang, Alex Yang, Jennifer Listgarten, Anant Sahai Department of Electrical Engineering and Computer Sciences University of California, Berkeley November 18, 2019
- [2] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, International Journal of Conflict and Violence (IJCV) 115 (3) (2015) 211–252.
- [3] R. Vaillant, C. Monrocq, Y. Le Cun, Original approach for the localisation of objects in images, IEE Proceedings-Vision, Image and Signal Processing 141 (4) (1994) 245–250.
- [4] R. Girshick, F. Iandola, T. Darrell, J. Malik, Deformable part models are convolutional neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 437–446.
- [5] J. Fan, W. Xu, Y. Wu, Y. Gong, Human tracking using convolutional neural networks, IEEE Trans. Neural Networks (TNN) 21 (10) (2010) 1610–1623.
- [6] A. Toshev, C. Szegedy, Deeppose: Human pose estimation via deep neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 1653–1660.
- [7] S. Eskenazi, P. Gomez-Kr¨amer, J.-M. Ogier, A comprehensive survey of mostly textual document segmentation algorithms since 2008, Pattern Recognition 64 (2017) 1–14.
- [8] G. Guo, A. Lai, A survey on still image based human action recognition, Pattern Recognition 47 (10) (2014) 3343–3361.
- [9] L. L. Presti, M. La Cascia, 3d skeleton-based human action classification: A survey, Pattern Recognition 53 (2016) 130–147.
- [10] J. Zhang, W. Li, P. O. Ogunbona, P. Wang, C. Tang, Rgb-d-based action recognition datasets: A survey, Pattern Recognition 60 (2016) 86–105.
- [11] Recent Advances in Convolutional Neural Networks Jiuxiang Gua,∗ , Zhenhua Wangb,∗ , Jason Kuenb , Lianyang Mab , Amir Shahroudyb , Bing Shuaib , Ting Liub , Xingxing Wangb , Li Wangb , Gang Wangb , Jianfei Caic , Tsuhan Chenc.

- [12] אנשים עם מוגבלות בישראל: אוכלוסייה, תעסוקה ועוני – נתונים מנהליים - אופיר פינטו, נטליה גיטלסון, אורן הלר, מירי אנדבלד, רבקה פריאור, דניאל גוטליב – 2018
- [13] אנשים עם מוגבלות בישראל 2019 נתונים סטטיסטים נבחרים. ד"ר גבי אדמון-ריק (נציבות שוויון זכויות לאנשים עם מוגבלות), אבנר גורדון (נציבות שוויון זכויות לאנשים עם מוגבלות). פרסום של משרד המשפטים ונציבות שוויון זכויות לאנשים עם מוגבלות.

Sites and coding libraries:

- https://keras.io
- https://github.com
- https://opencv.org
- https://www.analyticsvidhya.com/blog/2019/03/opencv-functions-computer-vision-python/
- https://www.analyticsvidhya.com/blog/2018/09/deep-learning-video-classification-python/
- https://www.tensorflow.org/
- https://towardsdatascience.com
- https://zulko.github.io/moviepy/ref/videofx.html
- https://wepik.com
- https://chaquo.com/chaquopy
- https://developer.android.com/studio