MINI PROJECT Report on

Gesture Recognition Interface to Control System Functions

Submitted in partial fulfillment of the requirements
of the degree of

BACHELOR OF ENGINEERING

in

INFORMATION TECHNOLOGY

by

Group No: 16

Roll No.	Name
32	Aayush Makharia
40	Dhiren Pamnani
45	Anirudh Poroorkara

Supervisor:
Dr Madhuri Rao
(Head of Department, Department of Information Technology, TSEC)

Information Technology Department
Thadomal Shahani Engineering College)
University of Mumbai

2019-2020

CERTIFICATE

This is to certify that the MINI PROJECT entitled **"Gesture Recognition Interface to Control System Functions"** is a bonafide work of

Roll No.	Name
32	Aayush Makharia
40	Dhiren Pamnani
45	Anirudh Poroorkara

Submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **"BACHELOR of ENGINEERING"** in **"INFORMATION TECHNOLOGY"**.

(Name and sign)
Project Guide

(Name and sign)
Head of Department

(Name and sign)
Principal

Mini Project Report Approval for T.E sem VI

Mini Project report entitled **Gesture Recognition Interface to Control System Functions** by

Roll No.	Name
32	Aayush Makharia
40	Dhiren Pamnani
45	Anirudh Poroorkara

is approved for the degree of ***"BACHELOR of ENGINEERING" in "INFORMATION TECHNOLOGY"***.

Examiners

1.--

2.--

Date:

Place:

Declaration

We declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

1) ____________________
(Signature)

____
(Name of student and Roll No.)

2) ____________________
(Signature)

____
(Name of student and Roll No.)

3) ____________________
(Signature)

____
(Name of student and Roll No.)

Date:

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

List of Pictures

1. PROJECT INTRODUCTION

Touch and gesture are two natural ways for a user to interact with one's environment. While touch necessarily involves a physical contact (e.g. to write a message on a phone, to grab a physical object, or to swipe touch sensitive textiles), gestures allow remote interactions (e.g. to interact with a smart screen, or with virtual-reality and augmented-reality objects). As such, gesture-based human computer interfaces can ease the use of digital computing in situations where it would previously have been difficult or even impossible because of practical constraints like interacting with everyday life physical objects (e.g. lights, mirrors, doorknobs, notebooks, ...) or like using computers in settings where the person has to focus entirely on a task (e.g. while driving a car, cooking or doing surgery). Gesture can convey semantic meaning, as well as contextual information such as personality, emotion or attitude. For instance, research shows that speech and gesture share the same communication system and that one's gestures are directly linked to one's memory. Among gestures, hand gestures distinguish themselves from two other types of gestures body gestures and head gestures. We chose to work on hand gestures since they can carry more information more easily than the two other types of gestures. One preferred way to infer the intent of a gesture is to use a taxonomy of gestures and to classify the unknown gesture into one of the existing categories based of the gesture data, in a similar way to what is done in computer vision for instance. The classification can either be obtained in real time at each time step or at the end of the gesture, depending on the processing power and the application needs. In this paper we propose a convolutional neural network architecture to classify among 10 gestures. The goal of this project is to train a Machine Learning algorithm capable of classifying images of different hand gestures, such as a fist, palm, showing the thumb, and others. This particular classification problem can be useful for Gesture Navigation, for example. The method I'll be using is Deep Learning with the help of Convolutional Neural Networks based on Tensorflow and Keras.

2. LITERATURE SURVEY

Deep Learning is part of a broader family of machine learning methods. It is based on the use of layers that process the input data, extracting features from them and producing a mathematical model. With the development of ubiquitous computing, current user interaction approaches with keyboard, mouse and pen are not sufficient. Due to the limitation of these devices the useable command set is also limited. Direct use of hands can be used as an input device for providing natural interaction. The traditional approach of keyboard and mouse input is becoming old and less efficient. The use of hand gesture as input to machines will provide a new way of faster interaction with machines. Hand gesture recognition system received great attention in the recent few years because of its manifoldness applications and the ability to interact with machine efficiently through human computer interaction. Through hand gesture we can give commands for various tasks and easily control the machine and its various functions. This project provides with a new way of interacting with the computers and to control various functions by the easy use of hand gestures.

Most of the researchers classified gesture recognition system into mainly three steps after acquiring the input image from camera(s), videos or even data glove instrumented device. These steps are: Extraction Method, features estimation and extraction, and classification or recognition as illustrated in Figure 1.

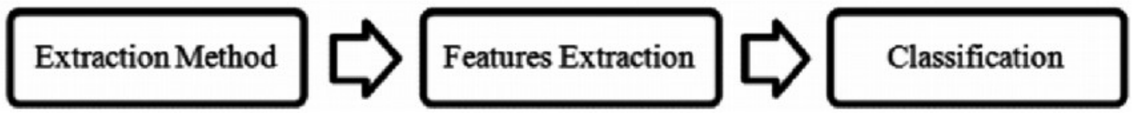


Figure 1: Flow of information

Segmentation is the process used for extraction. It is process is the first process for recognizing hand gestures. It is the process of dividing the input image (in this case hand gesture image) into regions separated by boundaries. The segmentation process depends on the type of gesture, if it is dynamic gesture then the hand gesture needs to be located and tracked, if it is static gesture (posture) the input image have to be segmented only.

Good segmentation process leads to perfect features extraction process and the latter play an important role in a successful recognition process. Features vector of the segmented image can be extracted in different ways according to particular application. Various methods have been applied for representing the features can be extracted. Some methods used the shape of the hand such as hand contour and silhouette while others utilized fingertips position, palm center, etc. created 13 parameters as a feature vector,

After modeling and analysis of the input hand image, gesture classification method is used to recognize the gesture. Recognition process is affected with the proper selection of features parameters and suitable classification algorithm

3. DESIGN CONSIDERATION

Machine Learning is very useful for a variety of real-life problems. It is commonly used for tasks such as classification, recognition, detection and predictions. Moreover, it is very efficient to automate processes that use data. The basic idea is to use data to produce a model capable of returning an output. This output may give a right answer with a new input or produce predictions towards the known data.

The goal of this project is to train a Machine Learning algorithm capable of classifying images of different hand gestures, such as a fist, palm, showing the thumb, and others. This particular classification problem can be useful for Gesture Navigation, for example. The method I'll be using is Deep Learning with the help of Convolutional Neural Networks based on Tensorflow and Keras.

Convolutional Neural Networks

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 x w x d(h = Height, w = Width, d = Dimension). Eg., An image of 6 x 6 x 3 array of matrix of RGB (3 refers to RGB values) and an image of 4 x 4 x 1 array of matrix of grayscale image.

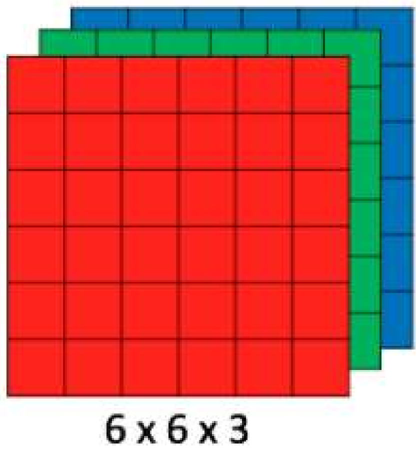


Figure 2: Array of RGB Matrix

Technically, deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernals), 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.

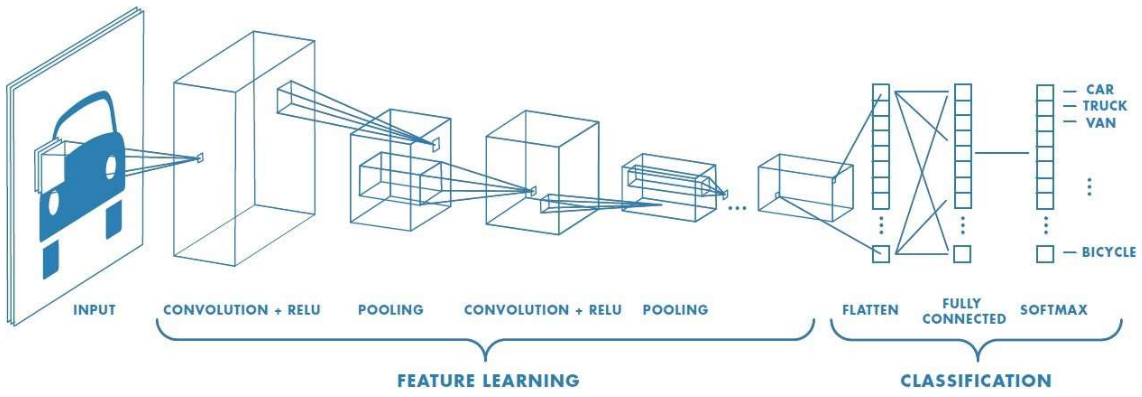
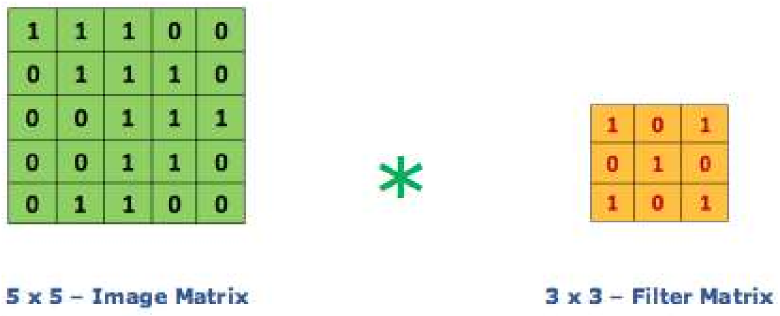


Figure 3: Neural network with many convolutional layers

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

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



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

5 x 5 – Image Matrix

*

1	0	1
0	1	0
1	0	1

3 x 3 – Filter Matrix

Figure 4: 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

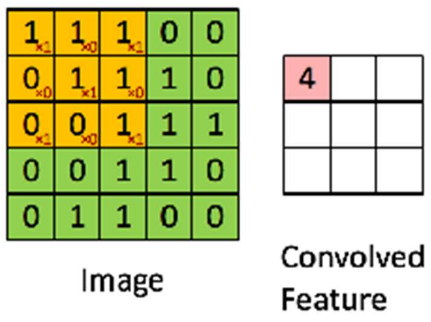


Figure 5: 3 x 3 Output matrix

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).

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.

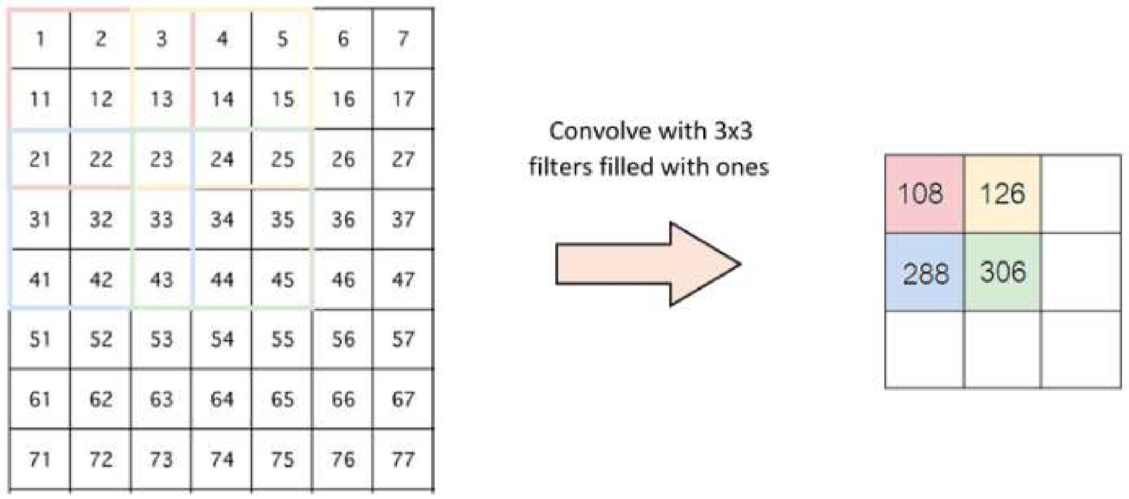


Figure 6: Stride of 2 pixels

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.

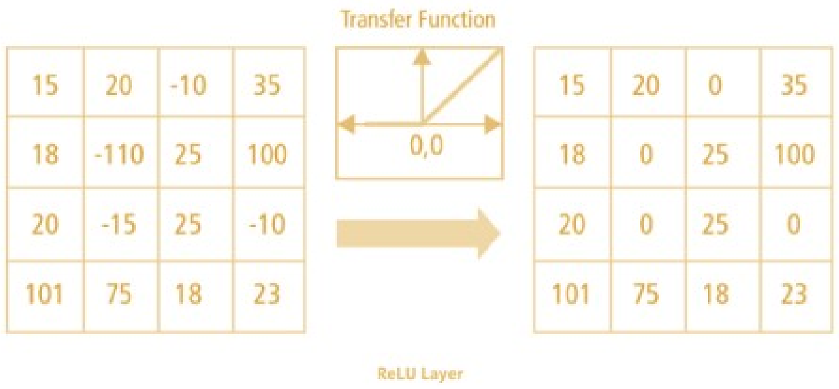


Figure 7: ReLU operation

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 down sampling 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.

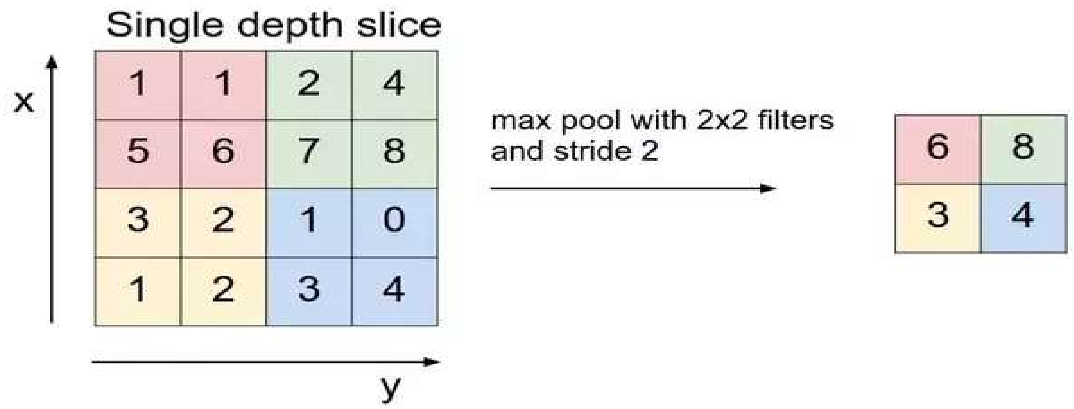


Figure 8: Max Pooling

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.

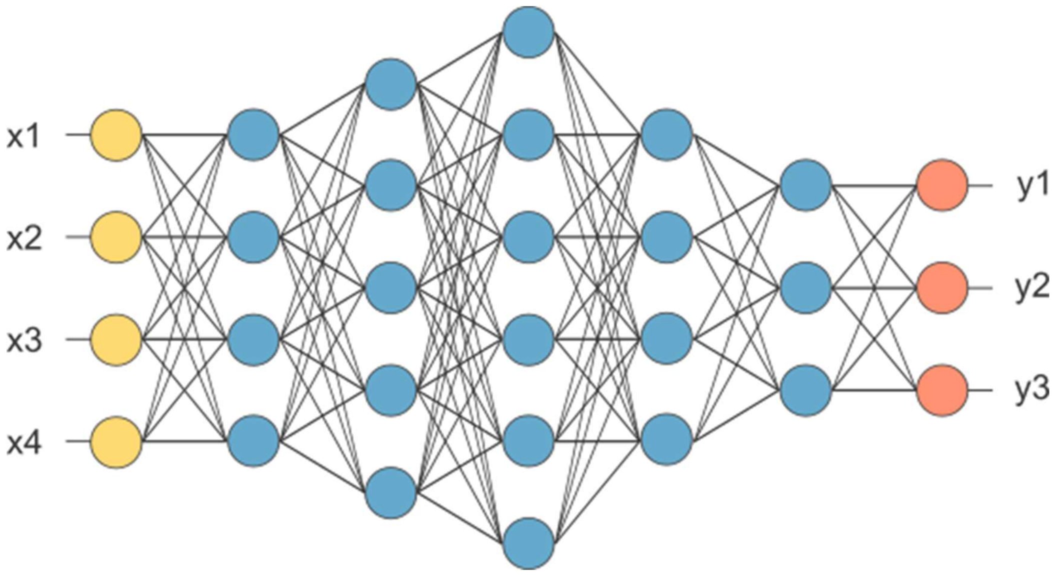


Figure 9: Fully Connected layer

In the above diagram, feature map matrix will be converted as vector (x1, x2, x3, …). 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.,

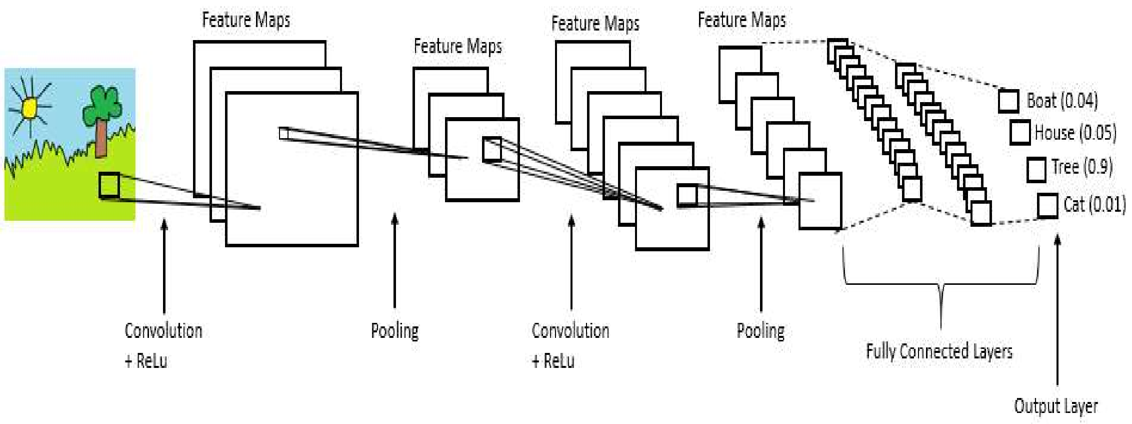


Figure 10: Complete CNN architecture

3. SOFTWARE REQUIREMENT

This project deals with a training model based on CNN and an Open CV user interface to interact and pass live images to the trained model. The CNN model is created using Keras and TensorFlow. The major Software requirements are:

IDE

An integrated development environment (IDE) is a software suite that consolidates basic tools required to write and test software. Developers use numerous tools throughout software code creation, building and testing. Development tools often include text editors, code libraries, compilers and test platforms. Without an IDE, a developer must select, deploy, integrate and manage all of these tools separately. An IDE brings many of those development-related tools together as a single framework, application or service. The integrated toolset is designed to simplify software development and can identify and minimize coding mistakes and typos.
The two major IDE used are:

- Syder
- Visual studio

CUDA

CUDA is a parallel computing platform and programming model developed by Nvidia for general computing on its own GPUs (graphics processing units). CUDA enables developers to speed up compute-intensive applications by harnessing the power of GPUs for the parallelizable part of the computation. CUDA is essential for accessing the GPU processing while training the model.

Keras

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It is used to structure the layers of the model and its different functions and classes help to create the CNN model.

Tensorflow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. TensorFlow is an open source library for numerical computation and large-scale machine learning. TensorFlow bundles together a slew of machine learning and deep learning (aka neural networking) models and algorithms and makes them useful by way of a common metaphor. It uses Python to provide a convenient front-end API for building applications with the framework

TensorFlow can train and run deep neural networks for handwritten digit classification, image recognition, word embeddings, recurrent neural networks, sequence-to-sequence models for machine translation, natural language processing, and PDE (partial differential equation) based simulations. Best of all, TensorFlow supports production prediction at scale, with the same models used for training. It is the main building block of the CNN model for gesture recognition and classification.

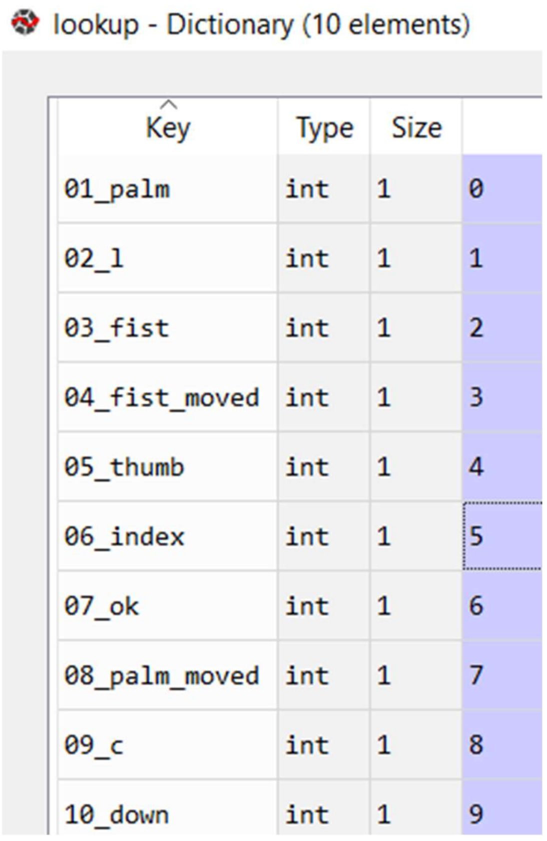
Open CV

OpenCV is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez. The library is cross-platform and free for use.

OpenCV provides with various functions to not only read live data but also perform actions on them like change the RGB image to Grayscale image. OpenCV is the best tool for creating a simple user interface that can interact with the model. The OpenCV interface captures live video from the webcam and each frame is passed as an image to the CNN model which predicts and classifies each image and returns the value to be displayed on the OpenCV interface.

5. IMPLEMENTATION

This project uses the Hand Gesture Recognition Database (citation below) available on Kaggle. It contains 20000 images with different hands and hand gestures. There is a total of 10 hand gestures of 10 different people presented in the data set. There are 5 female subjects and 5 male subjects.
The images were captured using the Leap Motion hand tracking device.



lookup - Dictionary (10 elements)

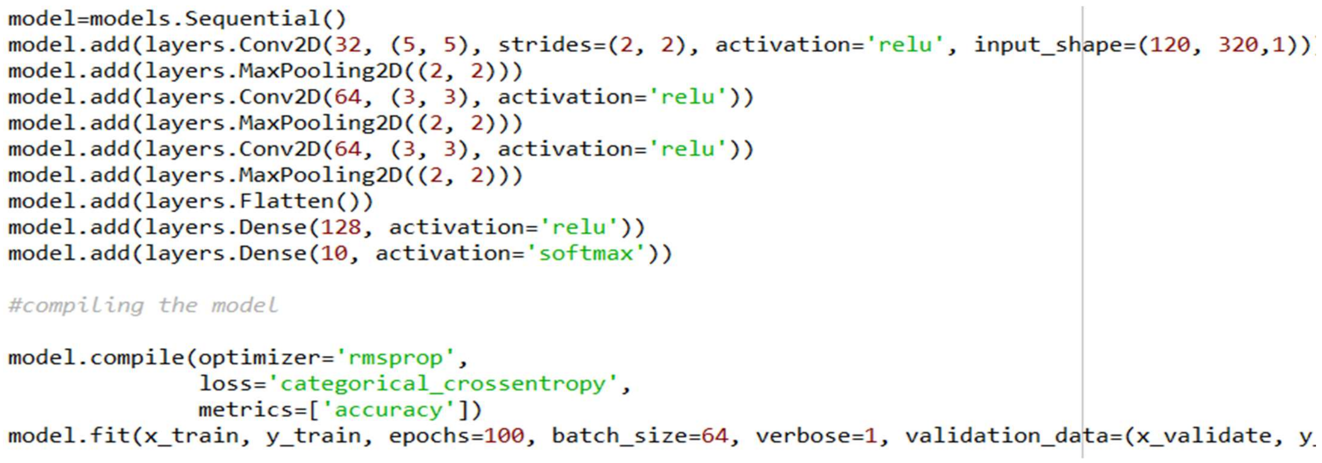
Key	Type	Size	
01_palm	int	1	0
02_l	int	1	1
03_fist	int	1	2
04_fist_moved	int	1	3
05_thumb	int	1	4
06_index	int	1	5
07_ok	int	1	6
08_palm_moved	int	1	7
09_c	int	1	8
10_down	int	1	9

Figure 11: Classification used for every hand gesture.

With that, we have to prepare the images to train the algorithm. We have to load all the images into an array that we will call **X** and all the labels into another array called **y**.

Scipy's train_test_split allows us to split our data into a training set and a test set. The training set will be used to build our model. Then, the test data will be used to check if our predictions are correct. A random_state seed is used so the randomness of our results can be reproduced. The function will shuffle the images it's using to minimize training loss.

Creating The model

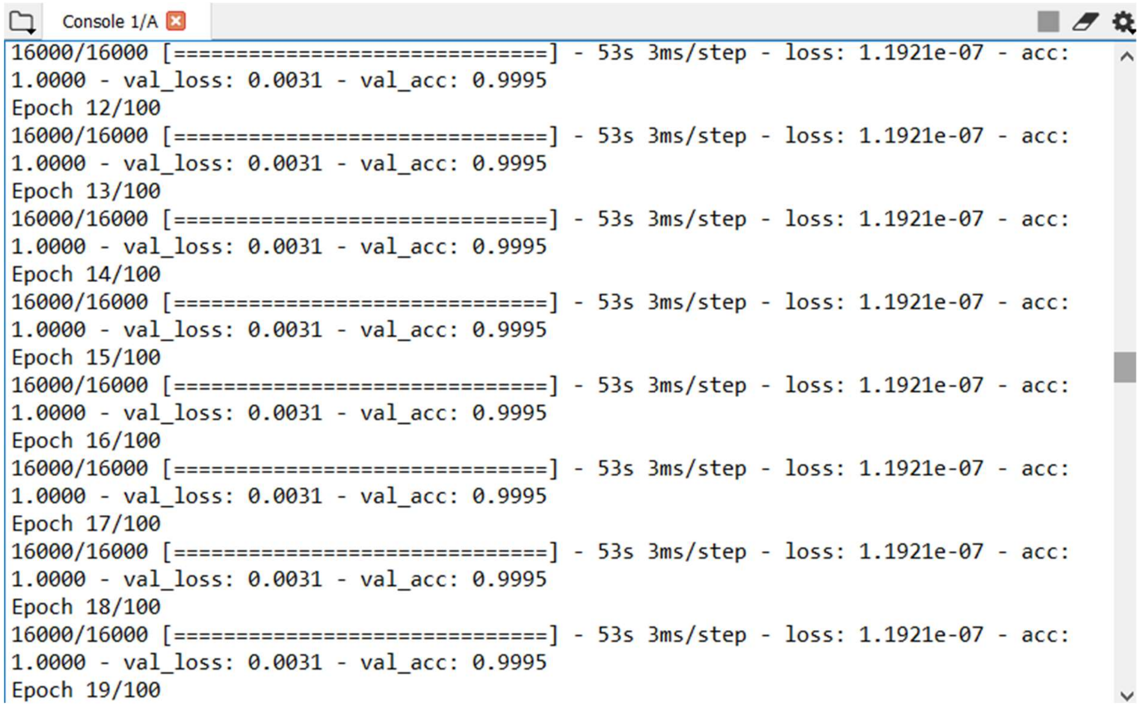


```
model=models.Sequential()
model.add(layers.Conv2D(32, (5, 5), strides=(2, 2), activation='relu', input_shape=(120, 320,1))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

#compiling the model

model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=100, batch_size=64, verbose=1, validation_data=(x_validate, y
```

Figure 12: Model Structure

To simplify the idea of the model being constructed here, we're going to use the concept of Linear Regression. By using linear regression, we can create a simple model and represent it using the equation $y = ax + b$. a and b (slope and intercept, respectively) are the parameters that we're trying to find. By finding the best parameters, for any given value of x, we can predict y. This is the same idea here, but much more complex, with the use of Convolutional Neural Networks.

We run them model through 100 epochs for a greater accuracy



```
16000/16000 [==============================] - 53s 3ms/step - loss: 1.1921e-07 - acc:
1.0000 - val_loss: 0.0031 - val_acc: 0.9995
Epoch 12/100
16000/16000 [==============================] - 53s 3ms/step - loss: 1.1921e-07 - acc:
1.0000 - val_loss: 0.0031 - val_acc: 0.9995
Epoch 13/100
16000/16000 [==============================] - 53s 3ms/step - loss: 1.1921e-07 - acc:
1.0000 - val_loss: 0.0031 - val_acc: 0.9995
Epoch 14/100
16000/16000 [==============================] - 53s 3ms/step - loss: 1.1921e-07 - acc:
1.0000 - val_loss: 0.0031 - val_acc: 0.9995
Epoch 15/100
16000/16000 [==============================] - 53s 3ms/step - loss: 1.1921e-07 - acc:
1.0000 - val_loss: 0.0031 - val_acc: 0.9995
Epoch 16/100
16000/16000 [==============================] - 53s 3ms/step - loss: 1.1921e-07 - acc:
1.0000 - val_loss: 0.0031 - val_acc: 0.9995
Epoch 17/100
16000/16000 [==============================] - 53s 3ms/step - loss: 1.1921e-07 - acc:
1.0000 - val_loss: 0.0031 - val_acc: 0.9995
Epoch 18/100
16000/16000 [==============================] - 53s 3ms/step - loss: 1.1921e-07 - acc:
1.0000 - val_loss: 0.0031 - val_acc: 0.9995
Epoch 19/100
```

Figure 13: Number of Epochs

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, CNNs have the ability to learn these filters/characteristics.

CNNs apply a series of filters to the raw pixel data of an image to extract and learn higher-level features, which the model can then use for classification. CNNs contains three components:

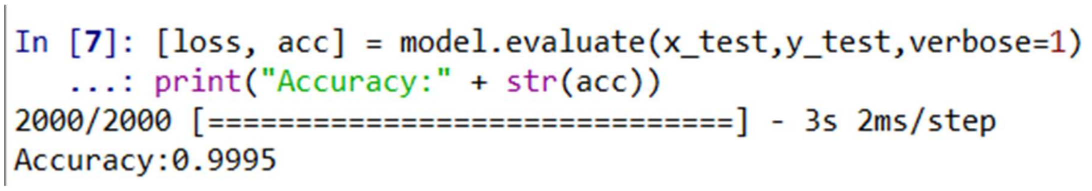
- Convolutional layers, which apply a specified number of convolution filters to the image. For each subregion, the layer performs a set of mathematical operations to produce a single value in the output feature map. Convolutional layers then typically apply a ReLU activation function to the output to introduce nonlinearities into the model.

- Pooling layers, which down sample the image data extracted by the convolutional layers to reduce the dimensionality of the feature map in order to decrease processing time. A commonly used pooling algorithm is max pooling, which extracts subregions of the feature map (e.g., 2x2-pixel tiles), keeps their maximum value, and discards all other values.

- Dense (fully connected) layers, which perform classification on the features extracted by the convolutional layers and down sampled by the pooling layers. In a dense layer, every node in the layer is connected to every node in the preceding layer.

Testing The model

Now that we have the model compiled and trained, we need to check if it's good. First, we run 'model.evaluate' to test the accuracy. Then, we make predictions and plot the images as long with the predicted labels and true labels to check everything. With that, we can see how our algorithm is working.
Later, we produce a confusion matrix, which is a specific table layout that allows visualization of the performance of an algorithm.



```
In [7]: [loss, acc] = model.evaluate(x_test,y_test,verbose=1)
   ...: print("Accuracy:" + str(acc))
2000/2000 [==============================] - 3s 2ms/step
Accuracy:0.9995
```

Figure 14: Test accuracy

Test accuracy of 99.95% was achieved.

OpenCV Interface

An OpenCV interface is created using the python-opencv library. The interface uses the webcam of the computer or laptop to read live video data, it then reads it in the form of frames per second. Each frame is treated as an image each frame is converted into an NumPy array and passed through the model to classify the gesture inside it. We resize the image to 120 x 320 for the model was trained on images of size 120 x 320. The model then returns the result with the highest probability gesture that the model thinks its in in that frame. The highest probable gesture is displayed on the screen.

6. RESULT AND CONCLUSION

We create an OpenCV interface to read and capture the real time videos and send each frame as an image to the model to recognize and classify the gesture and display the result on the screen. We achieve an accuracy of 99.95% and in idle conditions the model is able to recognize and classify each of the 10 gestures accurately. Here are sample of 2 results:

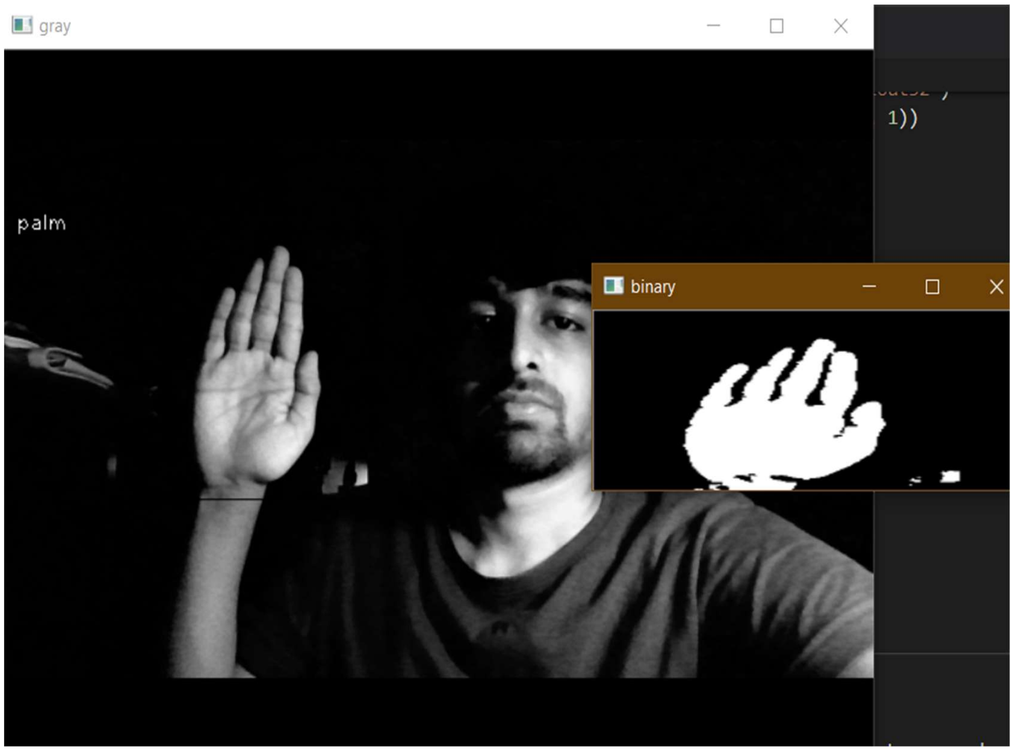


Figure 15: Result in OpenCV for Palm gesture

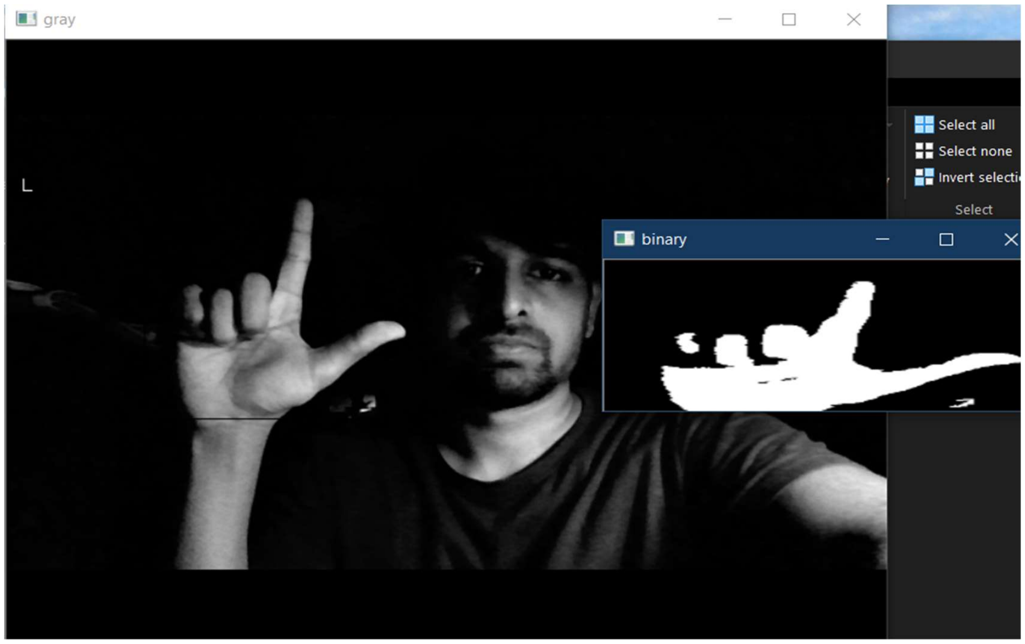


Figure 16: Result in OpenCV for L gesture

Thus, we can conclude that a Convolutional Neural Network model trained on a sample data set of 20,000 gesture images can recognize and classify gestures up to an accuracy of 99.95% in idle conditions.

Some future work could be to connect the OpenCV user interface with the PyAutoGUI library of python which can be used to control the basic functionalities of a computer like opening a web browser, or opening a particular website or scrolling up and down the browser or controlling the VLC media player. Each gesture can be assigned a particular task and when the model detects that gesture passed by the live feed of the OpenCV User Interface it would perform the assigned task. Thus, we could use hand gesture to control the computers day to day functions without having the need to touch the keyboard or mouse.

LIST OF REFERENCES

[1] Deep Learning for Hand Gesture Recognition on Skeletal Data Guillaume Devineau, Wang Xi, Fabien Moutarde, Jie Yang

[2] HAND GESTURE RECOGNITION: A LITERATURE REVIEW1Rafiqul Zaman Khanand 2Noor Adnan Ibraheem1,2 Department of Computer Science, A.M.U. Aligarh, India

[3] Hand Gesture Recognition with Batch and Reinforcement Learning, Arpit Goyal, Andrew Ciambrone, HossameldinShahin

[4] Machine Learning Techniques for Gesture Recognition by Carlos Antonio Caceres

[5] Deep Learning in Object Recognition, Detection, and Segmentation by Xiaogang Wang

[6] https://machinelearningmastery.com/stateful-stateless-lstm-time-series-forecasting-python/

[7] https://arxiv.org/abs/1602.00134

[8] https://web.stanford.edu/~jurafsky/slp3/9.pdf

[9] https://pjreddie.com/darknet/yolo/

[10] https://www.researchgate.net/publication/291051102_Hand_Gesture_Recognition_A_Literature_Review

[11] https://towardsdatascience.com/build-hand-gesture-recognition-from-scratch-using-neural-network-machine-learning-easy-and-fun-d7652dd105af

[12] http://www.ijergs.org/files/documents/Dynamic-Hand-Gesture-4.pdf