GIRIJANANDA CHOWDHURY INSTITUTE OF MANAGEMENT AND TECHNOLOGY,

GUWAHATI

PROJECT REPORT

ON

"SIGN LANGUAGE DETECTION USING MACHINE LEARNING"

Submitted in partial fulfilment of the requirement for the degree of

Bachelor of Technology

in

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING

ASSAM SCIENCE AND TECHNOLOGY UNIVERSITY, GUWAHATI

Submitted by:

Ambar Kashyap 170310007006

Bedanta Protim Deka 170310007007

Kongkon Pratim Saikia 170310007029

Project Guide:

Dr. Minakshi Gogoi

Associate Professor, Dept of CSE

GIMT Guwahati.

DECLARATION

We hereby declare that this project work entitled **"Sign language detection using machine learning"** was carried out by us under the guidance and supervision of **Dr. Minakshi Gogoi,** Associate professor department of Computer science and engineering, Girijananda Chowdhury Institute of Management and Technology, Guwahati. This project is submitted to Department of Computer Science and Engineering during the academic year 2020-21. The work is never produced before any authority except Assam Science and Technology University for evaluation.

Name of Project Members	**Roll N0**	**Signature**
1. Ambar Kashyap	170310007006	
2. Kongkon Pratim Saikia	170310007029	
3. Bedanta Protim Deka	170310007007	

GIRIJANANDA CHOWDHURY INSTITUTE OF

MANAGEMENT AND TECHNOLOGY, GUWAHATI

SESSION 2017-21

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that **Ambar Kashyap, Kongkon Pratim Saikia, Bedanta Protim Deka** students of BTech 4th Year, 8th Semester, have partially completed the project **"Sign language detection using machine learning"** during this academic session 2020-21 under my guidance and supervision.

I approve the project for submission as required for partial fulfilment for the completion of the Bachelors of Technology Degree.

(Signature of the Principal)

Prof. Thuleswar Nath
GIMT, Guwahati

(Signature of Project Guide)

Dr. Th. Shanta Kumar
Department of CSE,
GIMT, Guwahati

GIRIJANANDA CHOWDHURY INSTITUTE OF

MANAGEMENT AND TECHNOLOGY, GUWAHATI

SESSION 2017-21

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that **Ambar Kashyap, Kongkon Pratim Saikia, Bedanta Protim Deka** students of BTech 4th Year, 8th Semester, have partially completed the project **"Sign language detection using machine learning"** during this academic session 2020-21 under my guidance and supervision.

I approve the project for submission as required for partial fulfilment for the completion of the Bachelors of Technology Degree.

(Signature of Project Guide)

Dr. Minakshi Gogoi
Associate Professor
Department of CSE,
GIMT, Guwahati

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to our Principal, **Prof. Thuleswar Nath,** GIMT-Guwahati and to our HOD, **Dr. Th. Shanta Kumar,** Department of Computer Science and Engineering, GIMT-Guwahati for giving us the freedom to work on this project and also for providing occasional guidance when needed. Our sincere gratitude also goes to our project guide **Dr. Minakshi Gogoi,** Associate Professor, Department of Computer Science and Engineering, GIMT-Guwahati and also to all the faculty members of Department of Computer Science and Engineering, GIMT-Guwahati for being with us, encouraging us to complete our project with honesty and sincerity.

Name of Project Members	**Roll No**	**Signature**
1.Ambar Kashyap	170310007006	
2.Kongkon Pratim Saikia	170310007029	
3.Bedanta Protim Deka	170310007007	

ABSTRACT

Hand gesture recognition is very significant for human-computer interaction. In this work, we present a novel real-time method for hand gesture recognition. In our framework, the hand region is extracted from the background with the background subtraction method. Then, the palm and fingers are segmented so as to detect and recognize the fingers. Finally, a rule classifier is applied to predict the labels of hand gestures. The experiments on the data set of 45500 alphabetical data samples and 2661 numerical data samples show that our method performs well and is highly efficient. The model can recognize all 26 alphabets and 10 numbers. It still cannot recognize continuously moving hand signs and words or sentences.

CONTENTS

LIST OF FIGURES

Chapter 1: INTRODUCTION

1.1 Aim: Sign language detection using machine learning.

1.2 Objective: To determine the meaning of the hand sign using live camera imaging and then give the output to the user.

1.3 Scope of the work: Hand gesture recognition serves as a key for overcoming many difficulties and providing convenience for human life. The ability of machines to understand human activities and their meaning can be utilized in a vast array of applications. Human beings interact with each other either using a natural language channel such as words, writing, or by body language (gestures) e.g., hand gestures, head gestures, facial expression, lip motion and so on. As understanding natural language is important, understanding sign language is also very important. Sign language is the basic communication method within hearing disable people. People with hearing disabilities face problems in communicating with other hearing people without a translator. For this reason, the implementation of a system that recognizes sign language would have a significant benefit impact on deaf people's social lives.

Chapter 2: RELATED STUDY

The first approach in relation to sign language recognition was by Bergh in 2011 [1]. Haar wavelets and database searching were employed to build a hand gesture recognition system. Although this system gives good results, it only considers six classes of gestures. Many types of research have been carried out on different sign languages from different countries. In 2011, a real time American Sign Language recognition model was proposed utilizing Gabor filter and random forest [2]. A dataset of colour and depth images for 24 different alphabets was created. An accuracy of 75% was achieved utilizing both colour and complexity images, and 69% using depth images only. Depth images were only used due to changes in the illumination and differences in the skin pigment. In 2013, a multilayered random forest was also used to build a real time ASL model [3]. The system recognizes signs through applying random forest classifiers to the combined angle vector. An accuracy of 90% was achieved by testing one of the training images, and an accuracy of 70% was achieved for a new image. Starner, Weaver & Pentland tracked hand movements by using a (3D) glove and an (HMM) model. This model can gain (3D) information from the hands regardless of the spatial direction. An accuracy of (99.2%) was achieved on the test dataset. HMM utilizes time series datasets to follow hand movement and recognize them according to where the hand has been [4]. All the research that has been discussed above used linear classifiers, which are relatively simple and only require attribute extraction and pre-processing to be successful. Another approach is to use deep learning techniques. This approach was used to build a model that recognizes hand gestures in a continual video stream utilizing DBN models [5]. An accuracy of over 99% was achieved. . However, a research was done on different sign languages from different countries [6], and this was the most relevant work for the current study. An Italian Sign Language recognition system was built using CNNs to classify 20 Italian gestures. A Microsoft Kinect was applied to full body images of people, whereby the Kinect was able to capture depth images. Only the depth images were used for training and an accuracy of 91.7% was achieved. However, it was mentioned that the test dataset could be in the training dataset (and/or) the validation dataset [6]. The structural design of

the system consisted of two convolutional neural networks, one to extract higher body features and one to extract hand features. The data set, looking at People 2014, was used [7]. The depth map images were also used with a data set involving 20 Italian sign motions. The validation was 0.789675, and the final was 0.788804.

Chapter 3: Background of the work

3.1 What is Machine Learning?

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves. The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.[8]

Machine learning algorithms are often categorized as follows.

- **Supervised machine learning** algorithms can apply what has been learned in the past to new data using labelled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.

- In contrast, **unsupervised machine learning** algorithms are used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data. The system

doesn’t figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.

- **Semi-supervised machine learning** algorithms fall somewhere in between supervised and unsupervised learning, since they use both labeled and unlabeled data for training – typically a small amount of labeled data and a large amount of unlabeled data. The systems that use this method can improve learning accuracy. Usually, semi-supervised learning is chosen when the acquired labeled data requires skilled and relevant resources in order to train it / learn from it. Otherwise, acquiring unlabeled data generally doesn’t require additional resources.

- **Reinforcement machine learning** algorithms is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the ideal behavior within a specific context in order to maximize its performance. Simple reward feedback is required for the agent to learn which action is best; this is known as the reinforcement signal.

3.2 CNN

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most applied to analyse visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), 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. Counter-intuitively, most convolutional neural networks are only equivariant, as opposed to invariant, to translation. They have applications in image and video recognition, recommender systems, image classification, image segmentation,

medical image analysis, natural language processing, brain-computer interfaces, and financial time series.

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptron usually means fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "full connectivity" of these networks makes them prone to overfitting data. Typical ways of regularization, or preventing overfitting, include penalizing parameters during training (such as weight decay) or trimming connectivity (skipped connections, dropout, etc.) CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble patterns of increasing complexity using smaller and simpler patterns embossed in their filters. Therefore, on a scale of connectivity and complexity, CNNs are on the lower extreme.[9]

3.3 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms and can be freely distributed.[10]

From development to deployment and maintenance, Python helps developers be productive and confident about the software they’re building. Benefits that make Python the best fit for machine

learning and AI-based projects include simplicity and consistency, access to great libraries and frameworks for AI and machine learning (ML), flexibility, platform independence, and a wide community. These add to the overall popularity of the language.

The following packages have been used under Python in developing this project.

- **NumPy**

NumPy is an open-source numerical Python library. NumPy contains a multi-dimensional array and matrix data structures. It can be utilised to perform several mathematical operations on arrays such as trigonometric, statistical, and algebraic routines. Therefore, the library contains many mathematical, algebraic, and transformation functions.

- **Matplotlib**

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.

Chapter 4: Methods and techniques used

4.1 System design

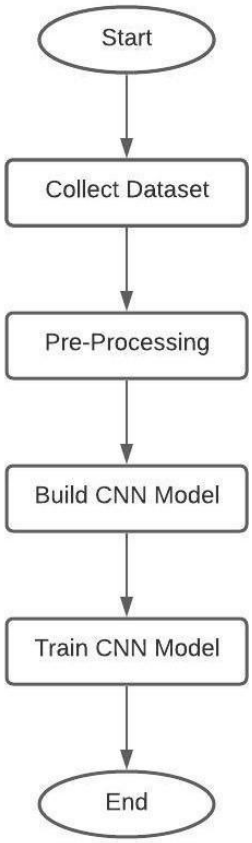

Fig 1: System flow diagram for training

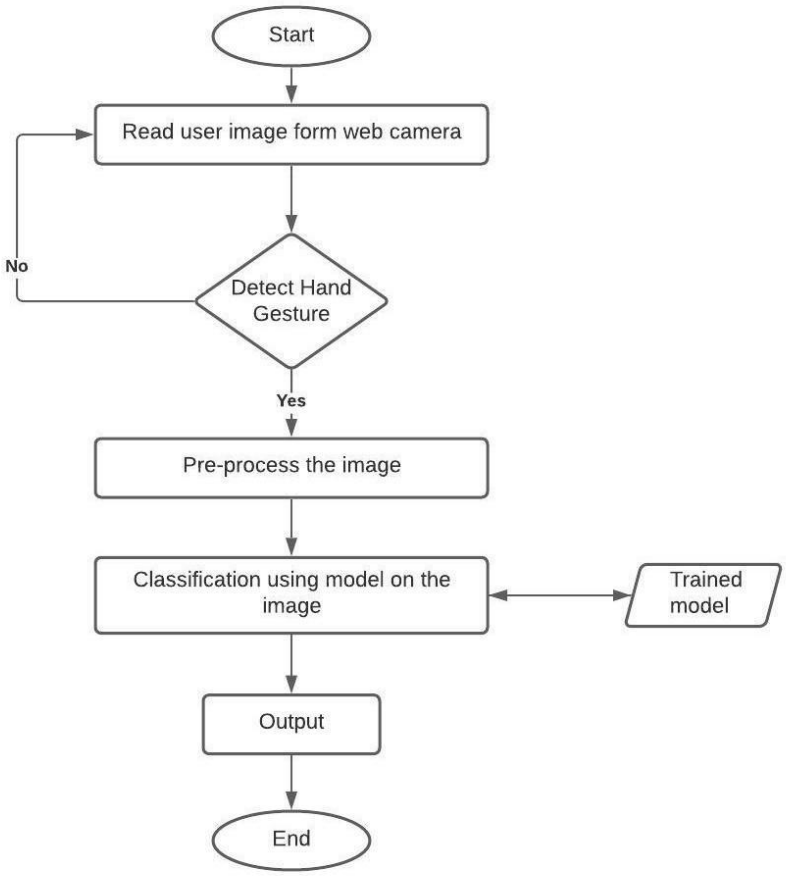

Fig 2: System flow diagram for evaluation

The reason for having two models is that some of the hand signs for numbers and alphabets are identical. The hand sign for ‘V’ is the same as the hand sign for ‘2’ for example. Therefore we decided to train two different models in order to overcome any confusion.

4.2 Specification

SOFTWARE REQUIREMENTS

1. **Operating System:** Windows 7, 8, 10
2. **Language:** Python 3.7
3. **IDE:** jupyter

HARDWARE REQUIREMENTS

1. **RAM:** 8 GB (minimum)
2. **CPU:** Intel i5 or AMD Ryzen 5 1600
3. **GPU:** NVIDIA (preferred) 940MX (minimum)

4.3 Datasets

The datasets have been taken from the kaggle website where for the alphabet dataset there are 45,500 training samples and 6500 testing samples. For the numbers dataset, there are a total of 2661 samples.[12]

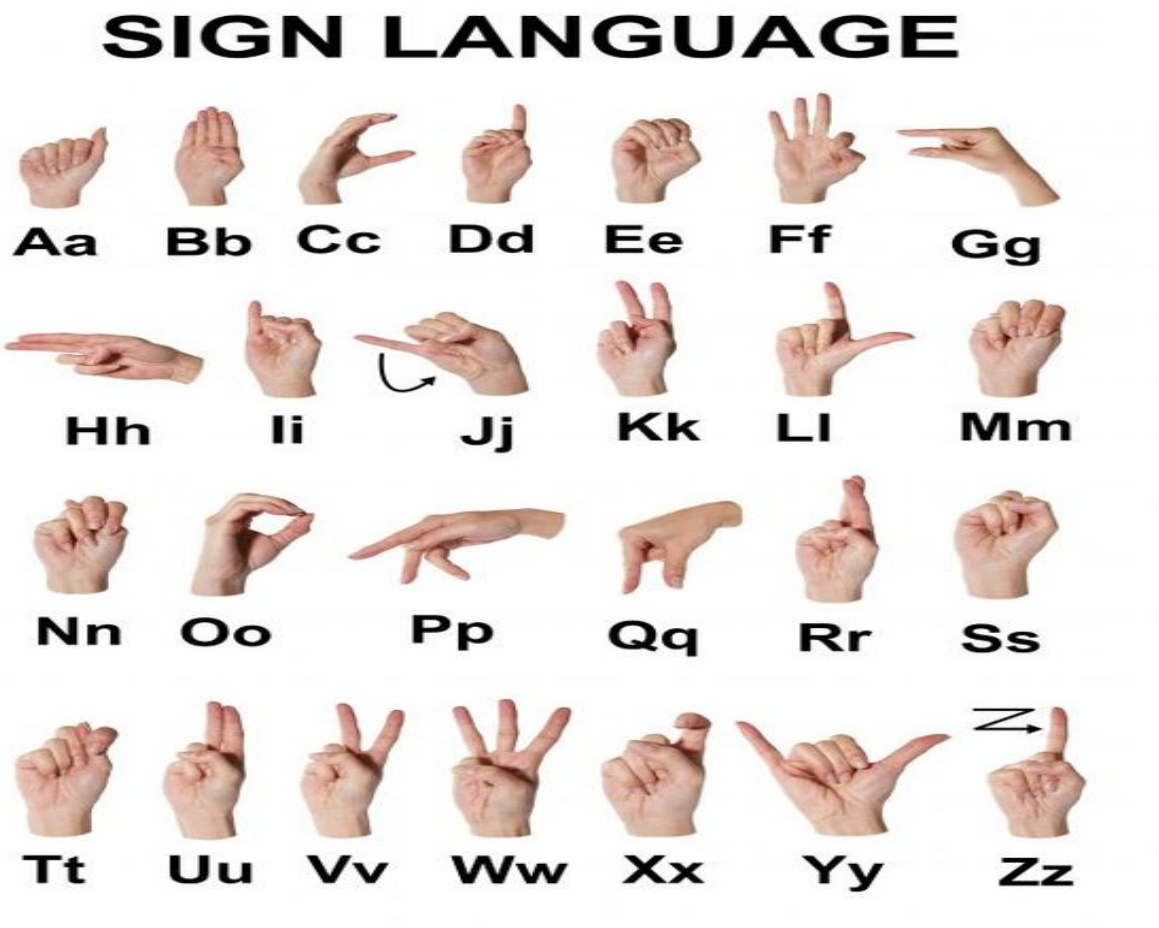

Fig 3: Dataset for alphabetical signs

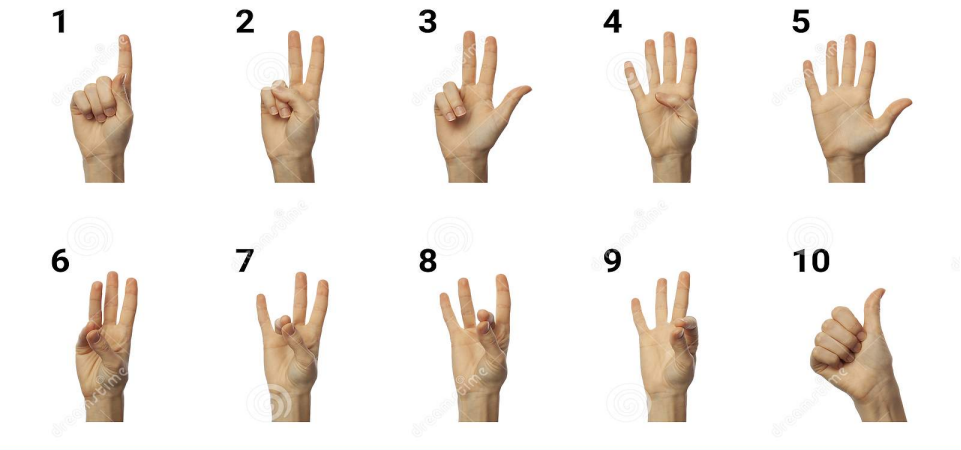

Fig 4: Dataset for numerical signs

4.4 Pre-processing

The alphabet dataset was already pre-processed and hence did not require any pre-processing. All the samples were already threshold images. The pre-processing steps applied to the numerical dataset are as follows.

1. Changing the RGB colours to greyscale.
2. Resizing the image into 100x100 pixels.
3. Removing the zero arrays and shuffling the data.
4. Visualizing the images.

After pre-processing, the image was visualized as below.

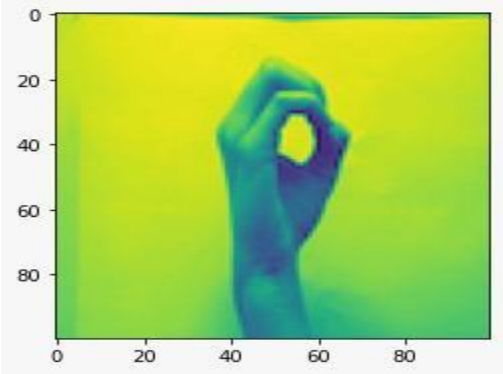

Fig 5: Pre-processed image

4.5 CNN ARCHITECTURE

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.[11]

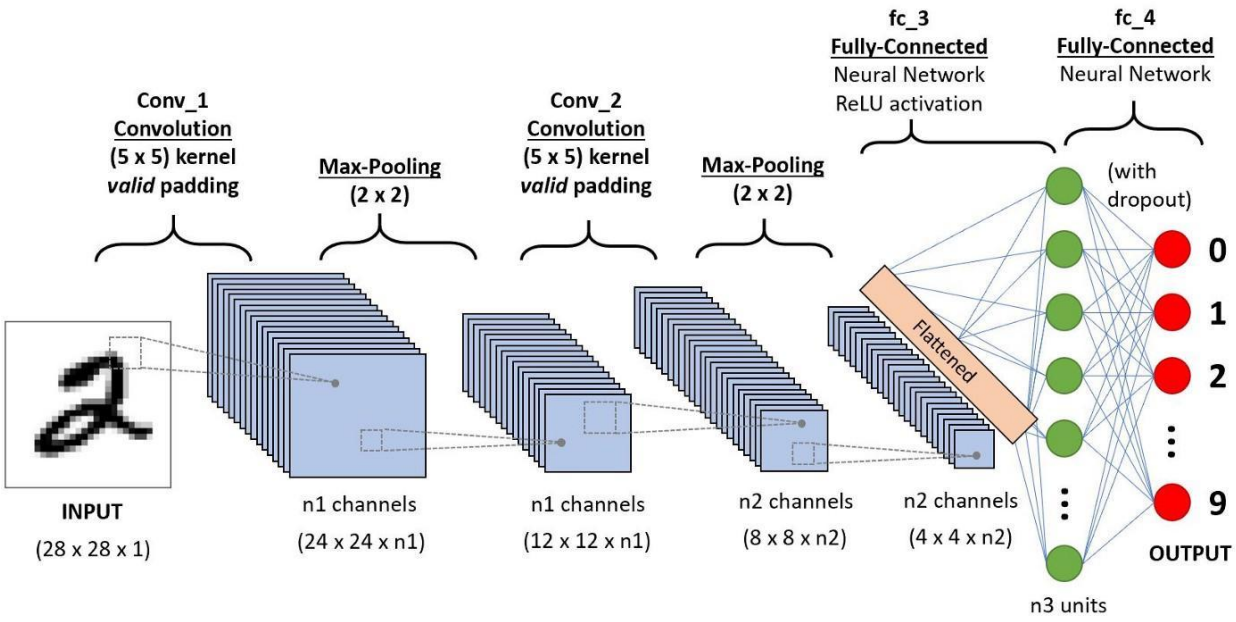

Fig 6: A CNN sequence to classify handwritten digits

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, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

4.5.1 Why ConvNets over Feed-Forward Neural Nets?

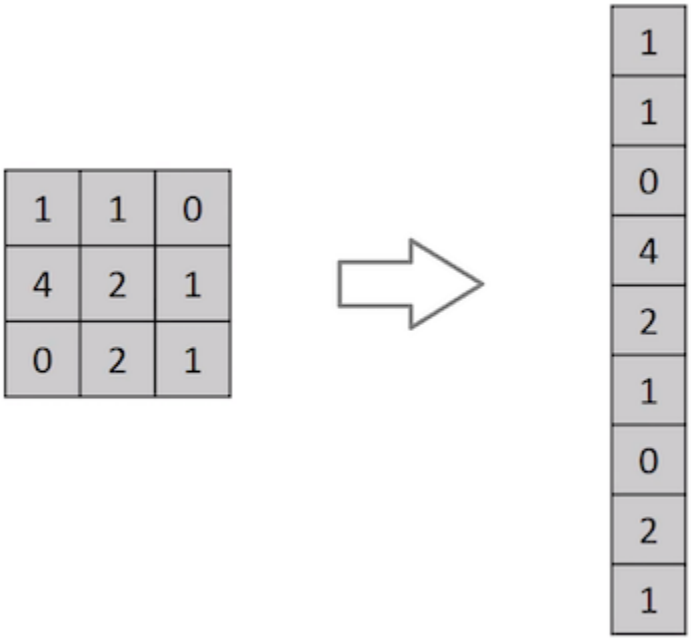

Fig 7: Flattening of a 3x3 image matrix into a 9x1 vector

An image is nothing but a matrix of pixel values, right? So why not just flatten the image (e.g. 3x3 image matrix into a 9x1 vector) and feed it to a Multi-Level Perceptron for classification purposes? In cases of extremely basic binary images, the method might show an average precision score while performing prediction of classes but would have little to no accuracy when it comes to complex images having pixel dependencies throughout.

A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

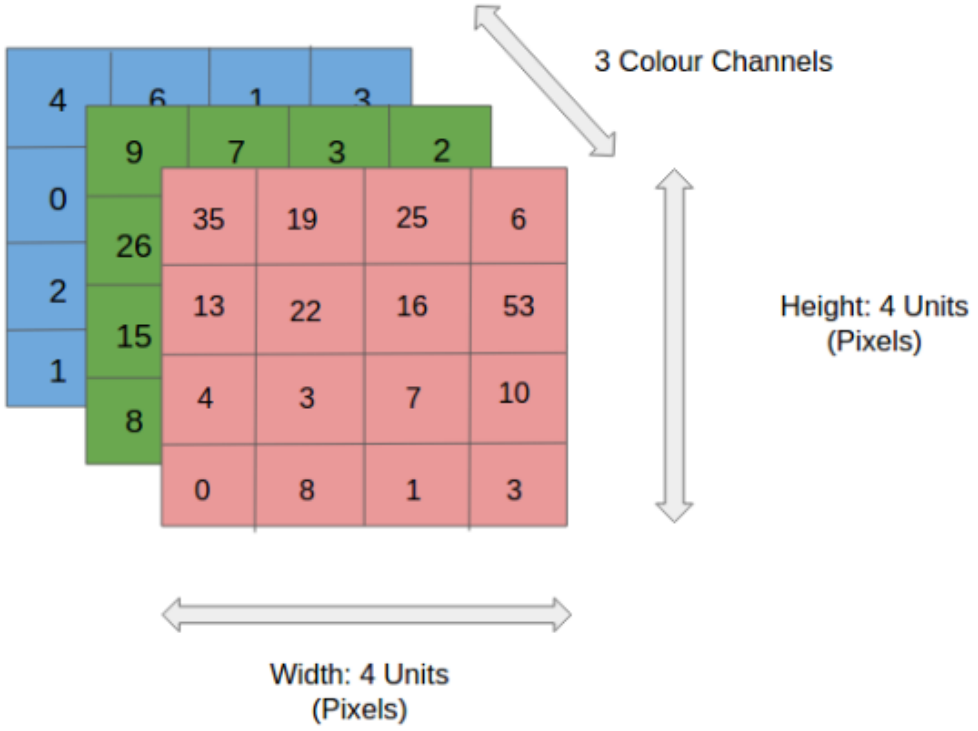

Fig 8: 4x4x3 RGB Image

In the figure, we have an RGB image which has been separated by its three colour planes — Red, Green, and Blue. There are a number of such colour spaces in which images exist — Grayscale, RGB, HSV, CMYK, etc.

You can imagine how computationally intensive things would get once the images reach dimensions, say 8K (7680×4320). The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good

prediction. This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets.

4.5.2 Convolution Layer — The Kernel

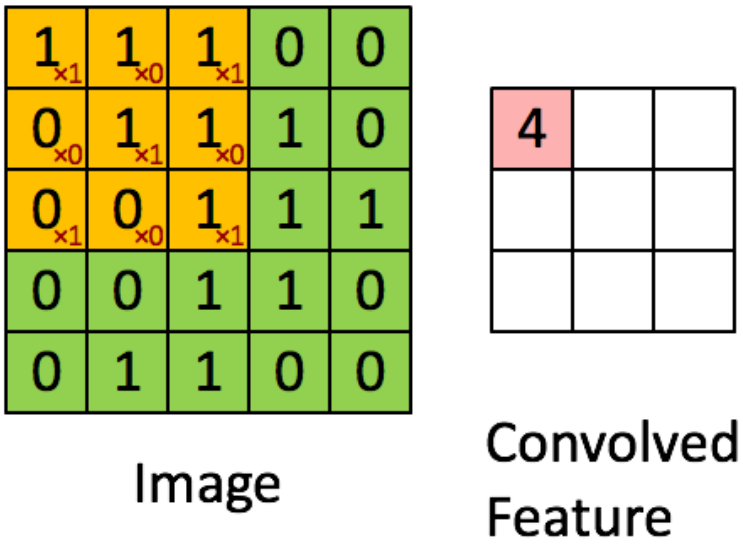

Fig 9: Convoluting a 5x5x1 image with a 3x3x1 kernel to get a 3x3x1 convolved feature

Image Dimensions = 5 (Height) x 5 (Breadth) x 1 (Number of channels, eg. RGB)

In the above demonstration, the green section resembles our **5x5x1 input image, I**. The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the **Kernel/Filter, K**, represented in the color yellow. We have selected **K as a 3x3x1 matrix.**

The Kernel shifts 9 times because of **Stride Length = 1 (Non-Strided)**, every time performing a **matrix multiplication operation between K and the portion P of the image** over which the kernel is hovering.

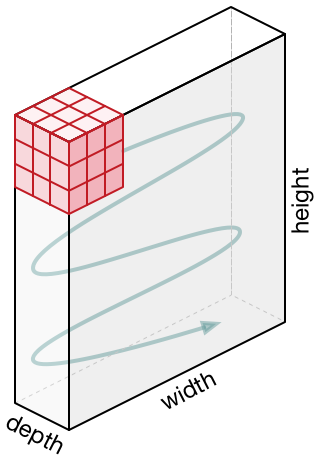

Fig 10: Movement of the Kernel

The filter moves to the right with a certain Stride Value till it parses the complete width. Moving on, it hops down to the beginning (left) of the image with the same Stride Value and repeats the process until the entire image is traversed.

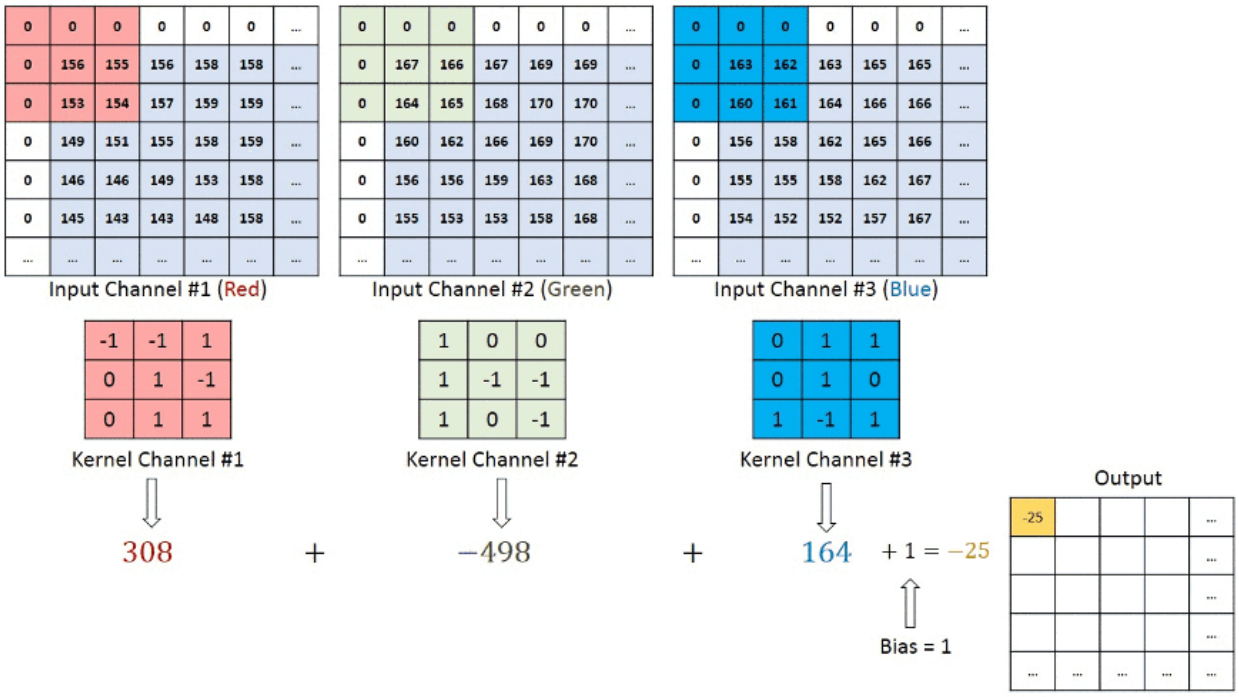

Fig 11: Convolution operation on a MxNx3 image matrix with a 3x3x3 Kernel

In the case of images with multiple channels (e.g. RGB), the Kernel has the same depth as that of the input image. Matrix Multiplication is performed between Kn and In stack ([K1, I1]; [K2, I2]; [K3, I3]) and all the results are summed with the bias to give us a squashed one-depth channel Convoluted Feature Output.

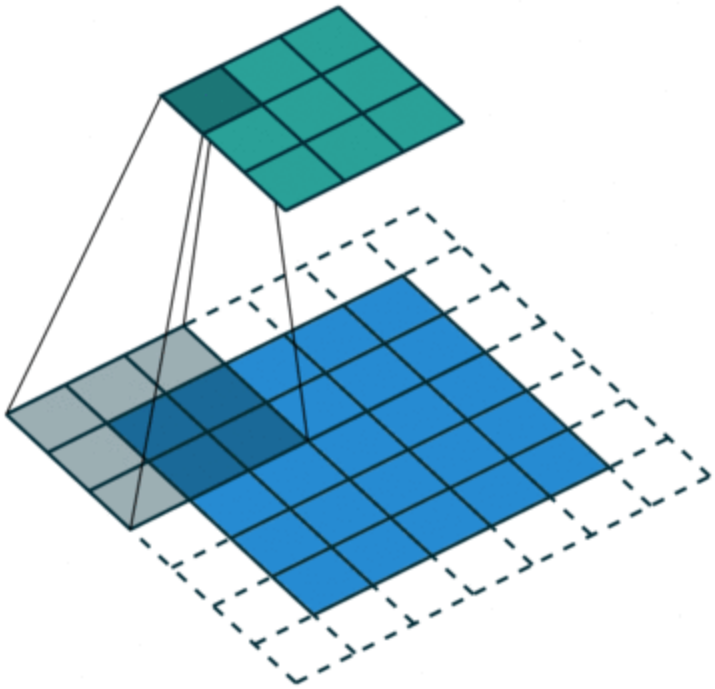

Fig 12: Convolution Operation with Stride Length = 2

The objective of the Convolution Operation is to **extract the high-level features** such as edges, from the input image. ConvNets need not be limited to only one Convolutional Layer. Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network, which has the wholesome understanding of images in the dataset, similar to how we would.

There are two types of results to the operation — one in which the convolved feature is reduced in dimensionality as compared to the input, and the other in which the dimensionality is either

increased or remains the same. This is done by applying **Valid Padding** in case of the former, or **Same Padding** in the case of the latter.

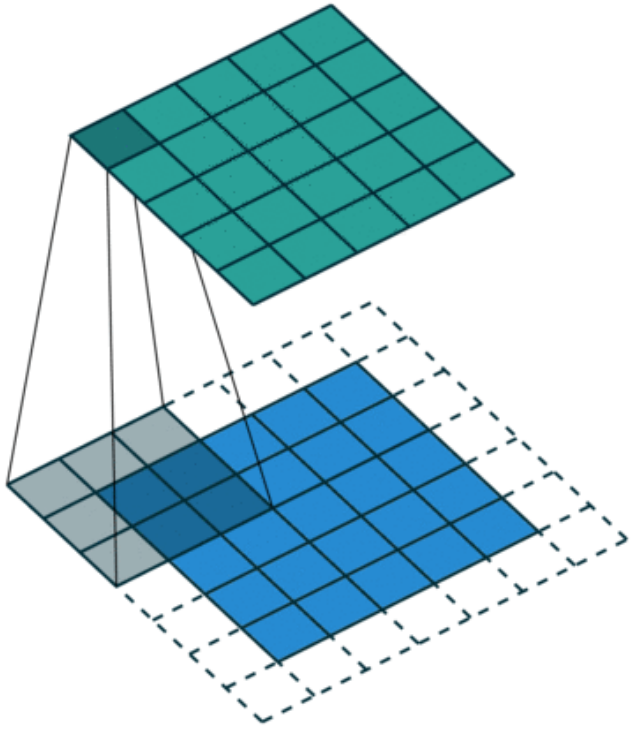

Fig 13: SAME padding: 5x5x1 image is padded with 0s to create a 6x6x1 image

When we augment the 5x5x1 image into a 6x6x1 image and then apply the 3x3x1 kernel over it, we find that the convolved matrix turns out to be of dimensions 5x5x1. Hence the name — **Same Padding**.

On the other hand, if we perform the same operation without padding, we are presented with a matrix which has dimensions of the Kernel (3x3x1) itself — **Valid Padding**.

4.5.3 Pooling Layer:

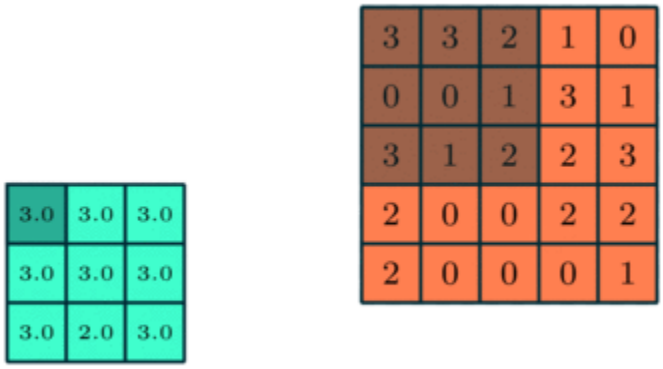

Fig 14: 3x3 pooling over 5x5 convolved feature

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to **decrease the computational power required to process the data** through dimensionality reduction. Furthermore, it is useful for **extracting dominant features** which are rotational and positional invariant, thus maintaining the process of effectively training the model.

There are two types of Pooling: Max Pooling and Average Pooling. **Max Pooling** returns the **maximum value** from the portion of the image covered by the Kernel. On the other hand, **Average Pooling** returns the **average of all the values** from the portion of the image covered by the Kernel.

Max Pooling also performs as a **Noise Suppressant**. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction. On the other hand, Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that **Max Pooling performs a lot better than Average Pooling**.

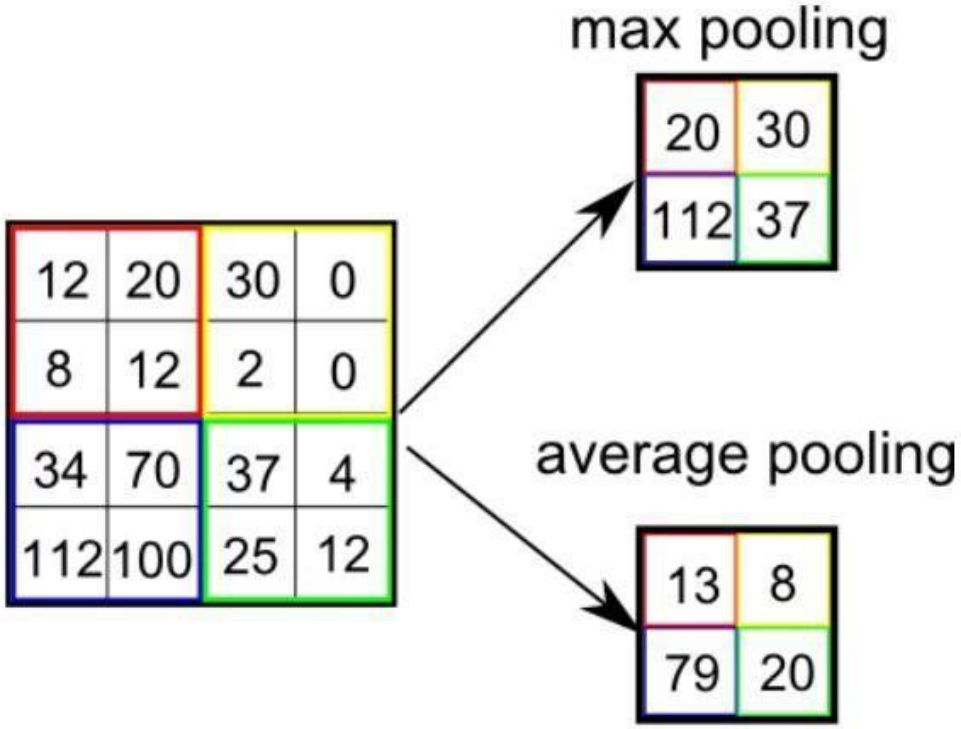

Fig 15: Types of Pooling

The Convolutional Layer and the Pooling Layer, together form the i-th layer of a Convolutional Neural Network. Depending on the complexities in the images, the number of such layers may be increased for capturing low-level details even further, but at the cost of more computational power.After going through the above process, we have successfully enabled the model to understand the features. Moving on, we are going to flatten the final output and feed it to a regular Neural Network for classification purposes.

4.5.4 Classification — Fully Connected Layer (FC Layer):

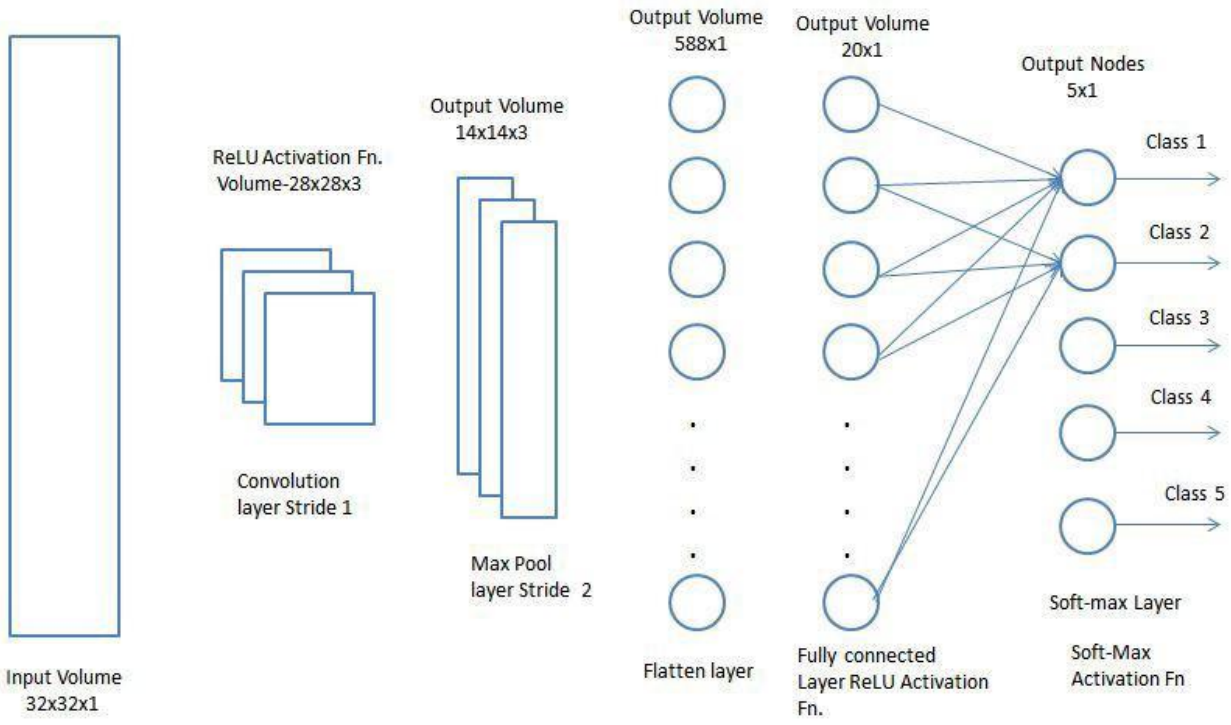

Fig 16: Fully connected layer

Adding a Fully-Connected layer is a (usually) cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully-Connected layer is learning a possibly non-linear function in that space.

Now that we have converted our input image into a suitable form for our Multi-Level Perceptron, we shall flatten the image into a column vector. The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the **Softmax Classification** technique.

Chapter 5: Results

5.1 CNN model summary

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 64, 64, 64)        1792
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 32)        18464
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 5, 5, 32)          9248
_________________________________________________________________
flatten (Flatten)            (None, 800)               0
_________________________________________________________________
dense (Dense)                (None, 256)               205056
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 26)                6682
=================================================================
Total params: 241,242
Trainable params: 241,242
Non-trainable params: 0
_________________________________________________________________
```

Fig 17: CNN model summary of alphabets

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 100, 100, 32)      320
_________________________________________________________________
module_wrapper (ModuleWrappe (None, 50, 50, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 50, 50, 64)        18496
_________________________________________________________________
module_wrapper_1 (ModuleWrap (None, 25, 25, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 40000)             0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               20480512
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 128)               65664
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_4 (Dense)              (None, 10)                1290
=================================================================
Total params: 20,566,282
Trainable params: 20,566,282
Non-trainable params: 0
_________________________________________________________________
```

Fig 18: CNN model summary of numbers

5.2 Testing output

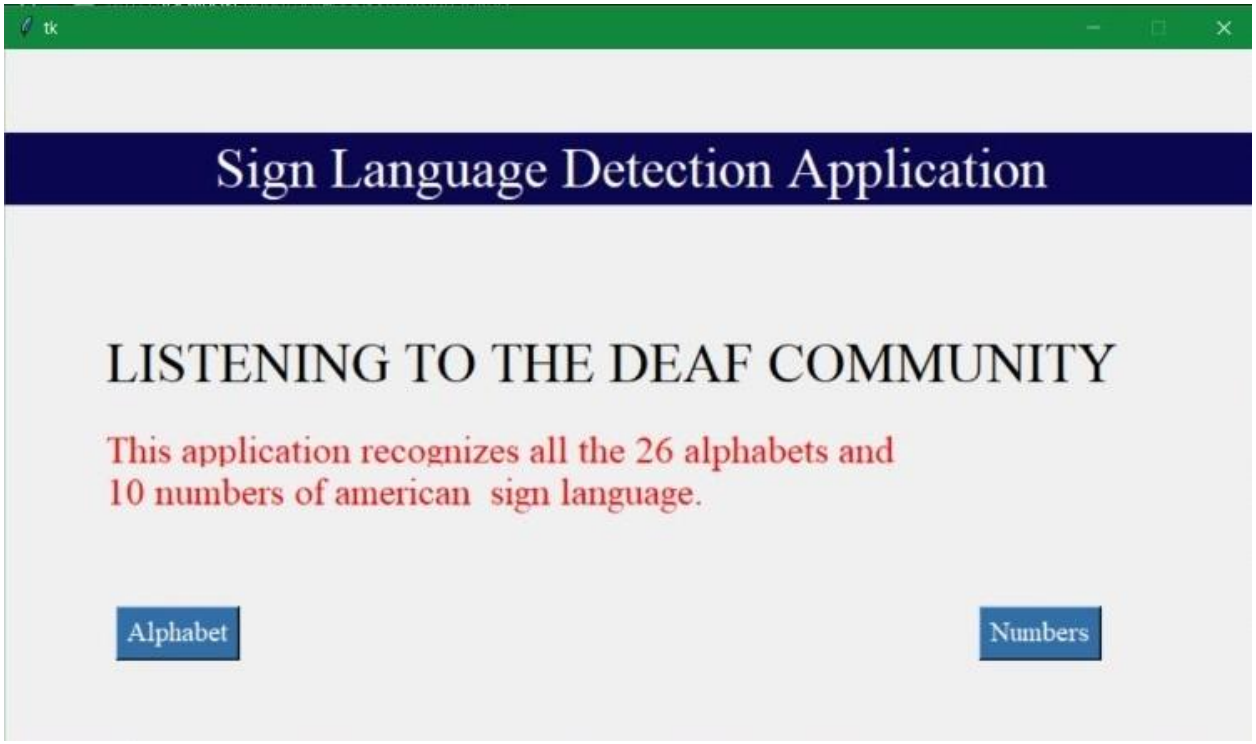

Fig 19: Graphical user interface

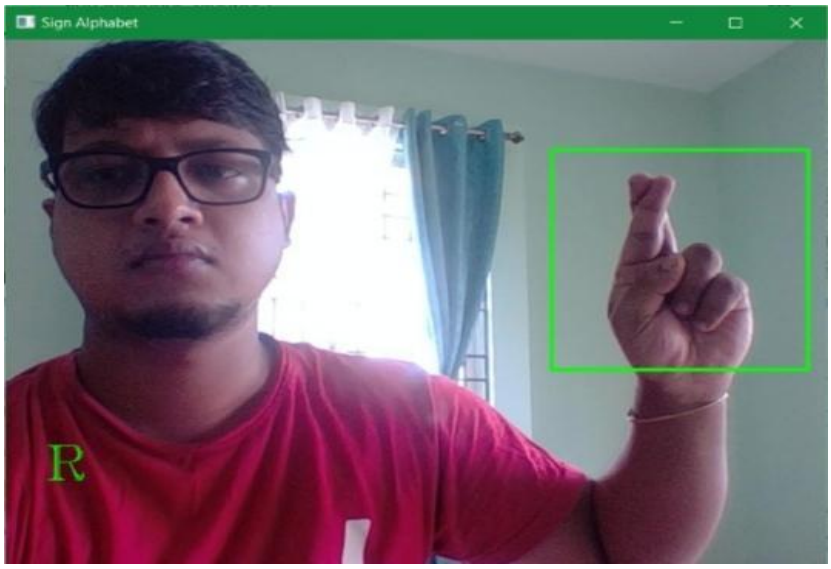

Fig 20: Testing-The letter ‘R’

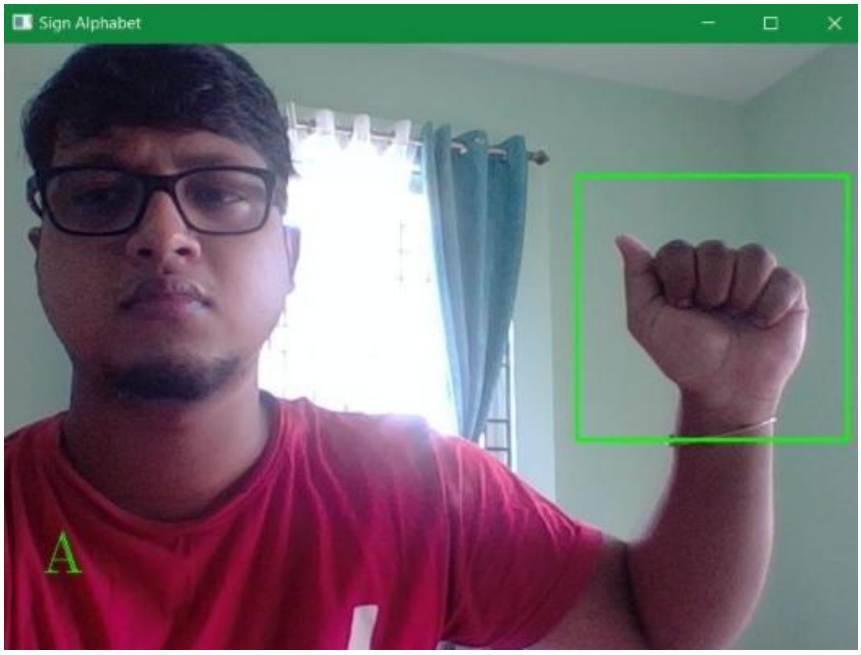

Fig 21: Testing-The letter ‘A’

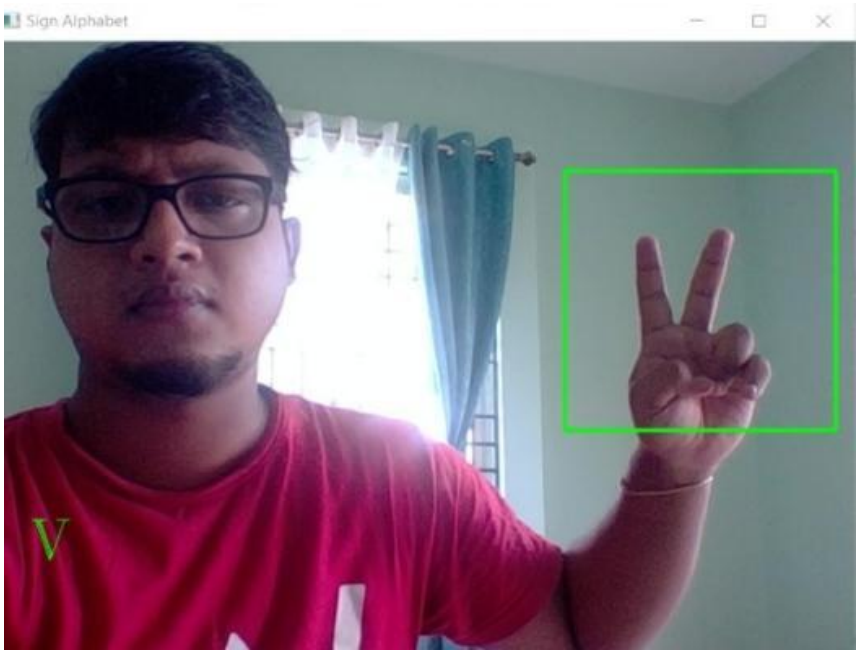

Fig 22: Testing-The letter ‘V’

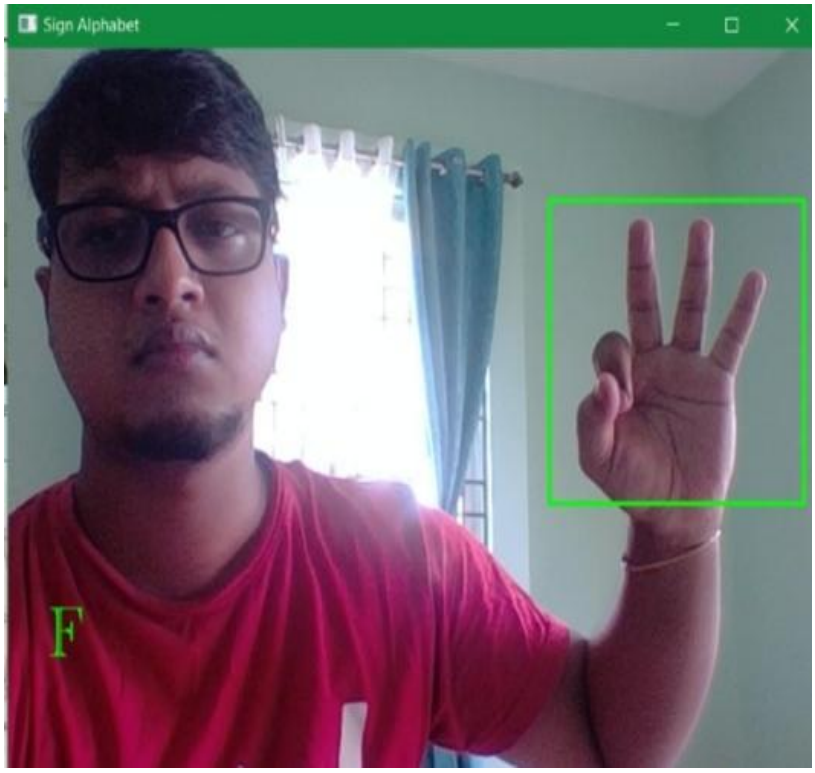

Fig 23: Testing-The letter ‘F’

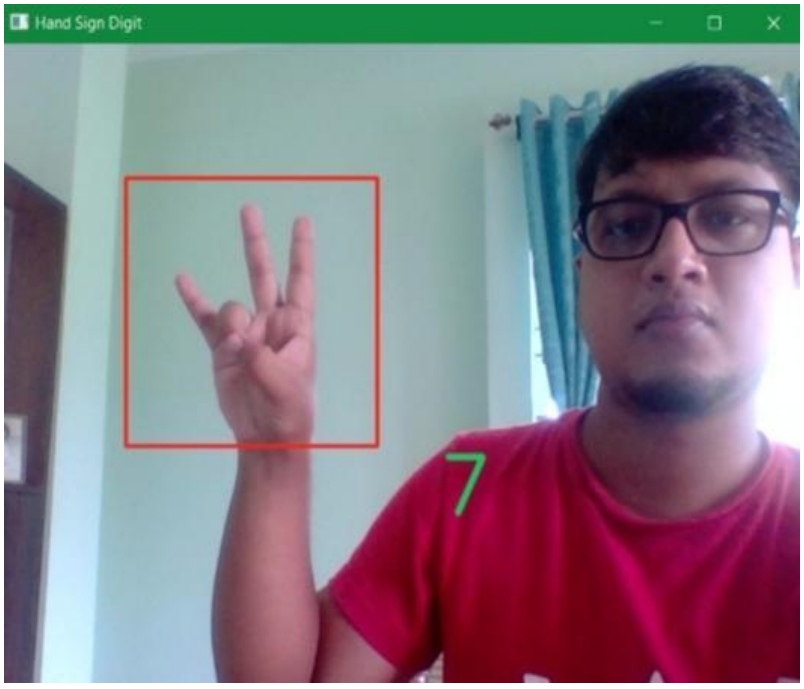

Fig 24: Testing-The number ‘7’

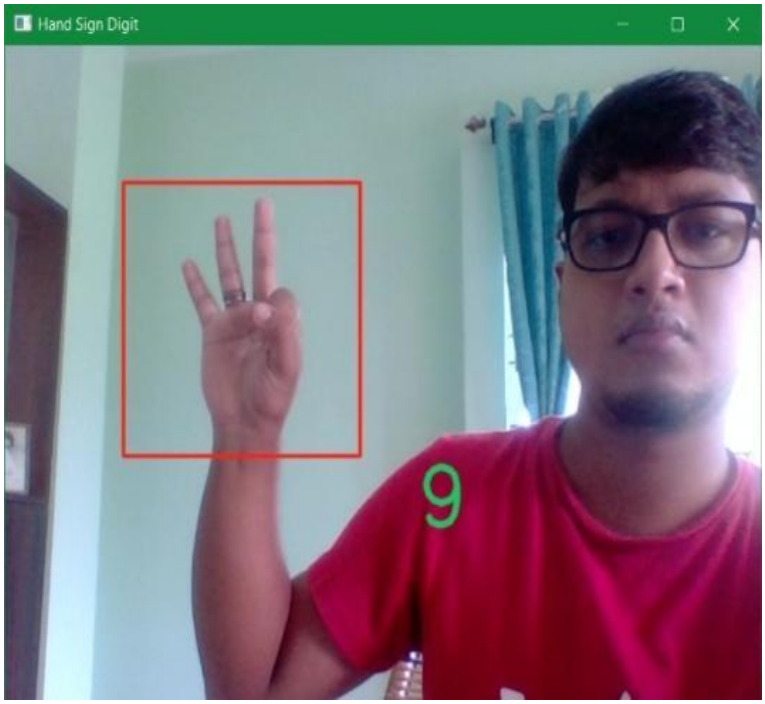

Fig 25: Testing-The number ‘9’

Fig 26: Testing-The number ‘3’

Chapter 6: Applications

1. It can be integrated with smart robots so that, along with verbal input, they can also take inputs using hand signs for people who can't speak.
2. It can be used in home automation software so that people who can't speak can also give commands with hand signs.
3. Another application of this hand sign detection can be used in the gaming industry. It can provide a fully contactless gaming experience in terms that no keyboard, mouse, joystick, gamepad etc. has to be used.

Chapter 7: Conclusion

Hand gesture recognition addresses a fault in interaction systems. Controlling things by hand is more natural, easier, more flexible, and cheaper, and there is no need to fix problems caused by hardware devices, since none is required. It was clear to need to put much effort into developing reliable and robust algorithms with the help of a camera sensor that has a certain characteristic to encounter common issues and achieve a reliable result. Each technique, however, has its advantages and disadvantages and may perform well in some challenges while being inferior in others.

7.1 Advantages:

1. A free of cost platform for the verbally disabled.
2. Simple user interface and very easy to use for anyone.

7.2 Limitations:

1. The model can only recognize static hand signs for the moment. Words or sentences that need hand movements still cannot be recognised by the model.
2. The accuracy of skin segmentation algorithms is limited because of objects in the background that are similar in colour to human skin.

Chapter 8: Future Scope

The model can be further developed to adapt to fast moving hand signs and eventually understanding video feeds containing continuously changing hand signs relating to a sentence. It can be used in various applications such as gaming, home automation, robotic control, creating a virtual environment etc.

REFERENCES:

[1]. Van den Bergh, M., & Van Gool, L. (2011, January). Combining RGB and ToF cameras for real-time 3D hand gesture interaction. In 2011 IEEE workshop on applications of computer vision (WACV) (pp. 66-72). IEEE.

[2]. Pugeault, N., & Bowden, R. (2011, November). Spelling it out: Real-time ASL fingerspelling recognition. In 2011 IEEE International conference on computer vision workshops (ICCV workshops) (pp. 1114-1119). IEEE.

[3]. Kuznetsova, A., Leal-Taixé, L., & Rosenhahn, B. (2013). Real-time sign language recognition using a consumer depth camera. In Proceedings of the IEEE International Conference on Computer Vision Workshops (pp. 83-90).

[4]. Starner, T., Weaver, J., & Pentland, A. (1998). Realtime american sign language recognition using desk and wearable computer based video. IEEE Transactions on pattern analysis and machine intelligence, 20(12), 1371-1375.

[5]. Suk, H. I., Sin, B. K., & Lee, S. W. (2010). Hand gesture recognition based on dynamic Bayesian network framework. Pattern recognition, 43(9), 3059- 3072.

[6]. Pigou, L., Dieleman, S., Kindermans, P. J., & Schrauwen, B. (2014, September). Sign langu age recognition using convolutional neural networks. In European Conference on Computer Vision (pp. 572-578). Springer, Cham.

[7]. Escalera, S., Baró, X., Gonzalez, J., Bautista, M. A., Madadi, M., Reyes, M., ... & Guyon, I. (2014, September). Chalearn looking at people challenge 2014: Dataset and results. In European Conference on Computer Vision (pp. 459-473). Springer, Cham.

[8] https://www.expert.ai/blog/machine-learning-definition/

[9] https://en.wikipedia.org/wiki/Convolutional_neural_network

[10] https://www.python.org/doc/essays/blurb/

[11]https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[12] https://www.kaggle.com/grassknoted/asl-alphabet