TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING

Kathmandu Engineering College
Department of Computer Engineering

Major Project Final Report

On

SIGN LANGUAGE DETECTION

[Code No: CT707]

By

Sankalpa Pokharel - 74064

Shameen Shrestha - 74067

Shreya Basnet - 74071

Subash Shrestha - 74074

Kathmandu, Nepal

Falgun, 2078

Tribhuvan University

Institute of Engineering

Kathmandu Engineering College

Department of Computer Engineering

SIGN LANGUAGE DETECTION

[Code No: CT707]

PROJECT REPORT SUBMITTED TO
THE DEPARTMENT OF COMPUTER ENGINEERING
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE BACHELOR OF ENGINEERING

By

Sankalpa Pokharel - 74064

Shameen Shrestha - 74067

Shreya Basnet - 74071

Subash Shrestha - 74074

Kathmandu, Nepal
Falgun, 2078
TRIBHUVAN UNIVERSITY
Kathmandu Engineering College
Department of Computer Engineering

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

Kathmandu Engineering College

Department of Computer Engineering

CERTIFICATE

The undersigned certify that they have read and recommended to the Department of Computer Engineering, a major project work entitled "Sign Language Detection" submitted by Sankalpa Pokharel – 25958, Shameen Shrestha – 25961, Shreya Basnet – 25964, Subash Shrestha– 25967 in partial fulfillment of the requirements for the degree of Bachelor of Engineering.

Er. Bishon Lamicchane
(Supervisor)
Department of
Computer Engineering
Kathmandu Engineering College

Assoc Prof. Nischal Acharya
(External Examiner)
Department of Electronics and
Computer Engineering
Pulchowk Campus
Institute of Engineering (IOE)

Er. Sharad Chandra Joshi
(Project Coordinator)
Department of
Computer Engineering
Kathmandu Engineering College

Er. Sudeep Shakya
(Head of Department)
Department of
Computer Engineering.
Kathmandu Engineering College

ACKNOWLEDGEMENT

We express our sincere gratitude to the Head of Department**, Er. Sudeep Shakya** and Deputy Head of Department**, Er. Kunjan Amatya** for helping us in the report. We would like to thank our project supervisor **Er. Bishon Lamichhane** for providing us the required guidance that we need for this project. We are grateful to our project coordinator **Er. Sharad Chandra Joshi** for the necessary templates and docs which has helped us a lot during the completion of our project.

We are grateful to people who helped us directly or indirectly to finalize this report. We must acknowledge our deep sense of gratitude to the people for their constant motivation and suggestions.

ABSTRACT

Sign languages are the primary form of communication between members of the hearing and speech disabled community. However, these languages are not widely known outside of these communities, and hence a communications barrier can exist between hearing and speech disabled and other people. With a rising demand in disability solutions, sign language recognition has become an important research problem in the field of computer vision and artificial intelligence. Current sign language detection systems, however, lack the basic characteristics like accessibility and affordability which is necessary for people with speech disability to interact with everyday environments. This project focuses on providing an affordable solution for understanding sign languages.

Artificial neural networks have proved to be an extremely useful approach to pattern classification tasks. The goal of this project is to build a neural network able to classify which letter of the Sign Language (SL) alphabet is being signed, given a real time image of a signing hand. This project is a first step towards building a possible sign language translator, which can take communications in sign language and translate them into written language. Such a translator would greatly lower the barrier for many hearing and speech disabled individuals to be able to communicate with others in day to day interactions.

Keywords: *Convolutional Neural Network, Sign Language*

TABLE OF CONTENT

TITLE PAGE……..ii

CERTIFICATION……..iii

ACKNOWLEDGEMENT……..iv

ABSTRACT……...v

LIST OF FIGURES……vii

LIST OF ABBREVIATIONS…….……viii

1. INTRODUCTION……1

1.1. BACKGROUND THEORY……..…1

1.2. PROBLEM STATEMENT……...2

1.3. OBJECTIVE……..……2

1.4. SCOPE OF PROJECT……...3

1.5. APPLICATIONS……...3

2. LITERATURE REVIEW……..4

3. METHODOLOGY……6

3.1. PROCESS MODEL……...6

3.2. BLOCK DIAGRAM…….....8

3.3. ALGORITHM……13

3.4. USE CASE DIAGRAM…….18

3.5. TOOLS TO BE USED…….....19

3.6. VERIFICATION AND VALIDATION……....23

4. EPILOGUE……..……26

4.1. RESULT AND CONCLUSION…..…….…….....26

4.2. FUTURE ENHANCEMENT…….…….…….....26

5. REFERENCES…….……27

6. SCREENSHOTS……...28

LIST OF FIGURES

Figure 3.1. Incremental Model……...6

Figure 3.2. Block Diagram…….8

Figure 3.3. Dataset of Sign ‘b’.....9

Figure 3.4. CNN Model…….10

Figure 3.5. Convolutional Layer……14

Figure 3.6. Max-Pooling……15

Figure 3.7. Fully Connected Neural Network……16

Figure 3.8. Receptive Field……17

Figure 3.9. Use Case Diagram……18

Figure 3.10. Train and Validation Accuracy……..23

Figure 3.11. Train and Validation Loss……..24

Figure 3.12. Training and Validation results in 10^{th} Epoch……25

LIST OF ABBREVIATIONS

ANN : Artificial Neural Network

API : Application Programming Interface

ASL : American Sign Language

CNN : Convolutional Neural Network

ML : Machine Learning

MLP : Multi-Layer Perceptron neural network

OpenCV : Open Source Computer Vision Library

ReLU: Rectified Linear Unit

RGB: Red Green Blue

sEMG: Surface ElectromyoGraphy

CHAPTER ONE: INTRODUCTION

1.1 BACKGROUND THEORY:

Sign language is a way of verbal exchange via human beings diminished by speech and listening to loss. Around 360 million human beings globally are afflicted via unable to hearing loss out of which 328 million are adults and 32 million children [1]. Hearing impairment extra than 40 decibels in the better listening ear is referred to as disabling listening to loss. Thus, with a growing range of people with hearing impairment, there is moreover a rise in demand for translators. Minimizing the verbal exchange gap among listening to impaired and regular humans turns into a want to make certain effective conversation among all. Sign language translation is one of the most growing lines of research nowadays and it is the natural manner of communication for the humans with hearing impairments.

Sign language is the mode of communication which uses visual ways like expressions, hand gestures to convey meaning. Sign language helps the hearing and speech disabled community to convey their thoughts with the world. Though there are people who can understand and communicate with sign language, a vast majority still has limited exposure to it. Sign language recognition is a problem that has been addressed in research for years. Sign Language or other variants enable the hearing and speech disabled community to share their thoughts with the world. Services like Google Translate, Bing Translator etc. have played a huge role in reducing barriers to written and spoken communication. Conversion of sign language into words or alphabet by an algorithm or a model can help bridge the gap between people with hearing or speaking impairment and the rest of the world. Research towards making Sign Language Recognition better would do the same for the deaf community and will technology be more accessible and inclusive.

For people with speech and hearing disability, vision perception becomes an even more important ability to interpret the environment. Sign languages become an important part of people with speech and hearing disability and sometimes are the only way people can communicate with society. This makes the problem of understanding sign language even more important and hence the field of computer vision is even more purposeful.

Our project aims to build neural networks that are able to classify signed SL letters using simple images of hands taken in real time with a personal device such as a laptop webcam. A system capable of recognizing finger spellings and hand signs in a real-time stream, requires the following tasks to be performed:

- Object Localization to detect hand gestures from a sampled video stream.
- Recognizing a sequence of finger-spellings gestures as a word
- Recognizing a sequence of frames as words represented by motion signs.
- Constructing a sentence from the recognized words [2].

1.2 PROBLEM STATEMENT

Sign Language enables the hearing and speech disabled community to share their thoughts with the world. They face difficulty interacting with others, because there are limited people who can understand and communicate with sign language, a vast majority still have limited exposure to it. Some previously developed techniques to bridge the gap are mostly sensors based and they didn't give the general solution.

So, with the help of an image processing technique with an convolutional neural network (CNN) we can train the system to recognize the hand gestures captured with a web camera in real time and understand what the hand gestures mean in sign language.

1.3 OBJECTIVES

The purpose of this project is to develop an application that uses gesture recognition to understand sign language. It uses image processing techniques to distinguish the hand from the background and then identify the fingertips to determine the gestures and understand the gestures and show it in text form. The main objectives of our project are:

- To build a neural network that is able to classify which letter of the Sign Language alphabet is being signed.
- To develop a system that can capture hand gestures and navigate the alphabets as in text format.

1.4 SCOPE OF PROJECT

Our project on sign language recognition can provide an affordable solution for understanding Sign Language. The main focus of our project is to bridge the communication gap between people with hearing and speech disabilities and people who do not understand the language used by them .This project is a first step towards building a possible sign language translator.

1.5 APPLICATIONS

- To identify the symbolic expression through images with the aim of bridging the communication gap between normal and hearing and speech impaired person.
- To understand the message and information that speech and hearing disabled people want to convey.
- To provide assistance to people with hearing and speaking disability.

CHAPTER TWO: LITERATURE REVIEW

Sign language recognition is a problem that has been addressed in research for years. However, we are still far from finding a complete solution available in our society.

The first problem of sign language translation is data collection. The current method is that researchers record or shoot hand movements of sign language users through sensors or cameras and carry out the corresponding feature extraction and model establishment of the data flow in the computer [3]. Sign language recognition (SLR) techniques can be categorized into three groups: computer vision-based models, wearable sensor-based systems, and hybrid systems. [4]

Computer vision-based models use at least one camera and image processing techniques to classify gestures. The primary advantage of this approach is that gestures are performed using an unadorned hand, which is comfortable for the user. System costs are limited to the price of the camera, computer, and software used to process data.

Sensor-based sign language recognition approaches involve using strain sensors, surface electromyography (sEMG) sensors [5], tactile or pressure sensors, or inertial sensors such as accelerometers, magnetometers, and gyroscopes. Recent advancements in technology have enabled the development of small and cost-effective sensors, microcontrollers, circuit boards, and batteries. Compared to camera-based systems, sensors are not as easily influenced by environmental conditions.

The last category is hybrid recognition approaches, which typically involve a camera and one or more wearable sensors. In these approaches, sensors provide additional information related to hand configuration. Various studies have examined the feasibility of using color-coded gloves in computer-vision SLR approaches [6]. Although high recognition rates can be attained, hybrid approaches are subject to environmental factors, they can restrict user mobility, and they have high computational power requirements.

Some related works to Sign Language Translator are:

1. A mobile application called hand talk translator. Led by Hugo, the Hand Talk app automatically translates text and audio to American Sign Language (ASL) [Beta] and Brazilian Sign Language (Libras) through artificial intelligence. Hand Talk was elected as the World's Best Social App by the UN, emerging as a pocket translator with the goal of bringing people together.

   The Hand Talk app is a powerful tool, used in the most diverse contexts:
   • In the classroom, by teachers, students, and interpreters as a complementary communication resource.
   • By sign language students that want to improve their vocabulary with Hugo's Help. [7]
2. SignAll is another application that has successfully translated between signed and spoken languages. It is an innovative tech company that has developed the first automated sign language translation solution by leveraging computer vision and natural language processing (NLP). [8]
3. SLAIT is another such application that recognizes ASL using Media Pipe and custom neural networks. [9]

CHAPTER THREE: METHODOLOGY

3.1 PROCESS MODEL

"Sign Language Recognition" developed using the incremental model. Incremental model is a process of software development where requirements are broken down into multiple standalone modules of the software development cycle. Incremental development is done in steps from analysis, design, implementation, testing/verification, maintenance. Each iteration passes through theo0 requirements, design, coding and testing phases. And each subsequent release of the system adds function to the previous release until all designed functionality has been implemented. We chose this model of development as the program will contain more than one complex features and they are added one at a time.

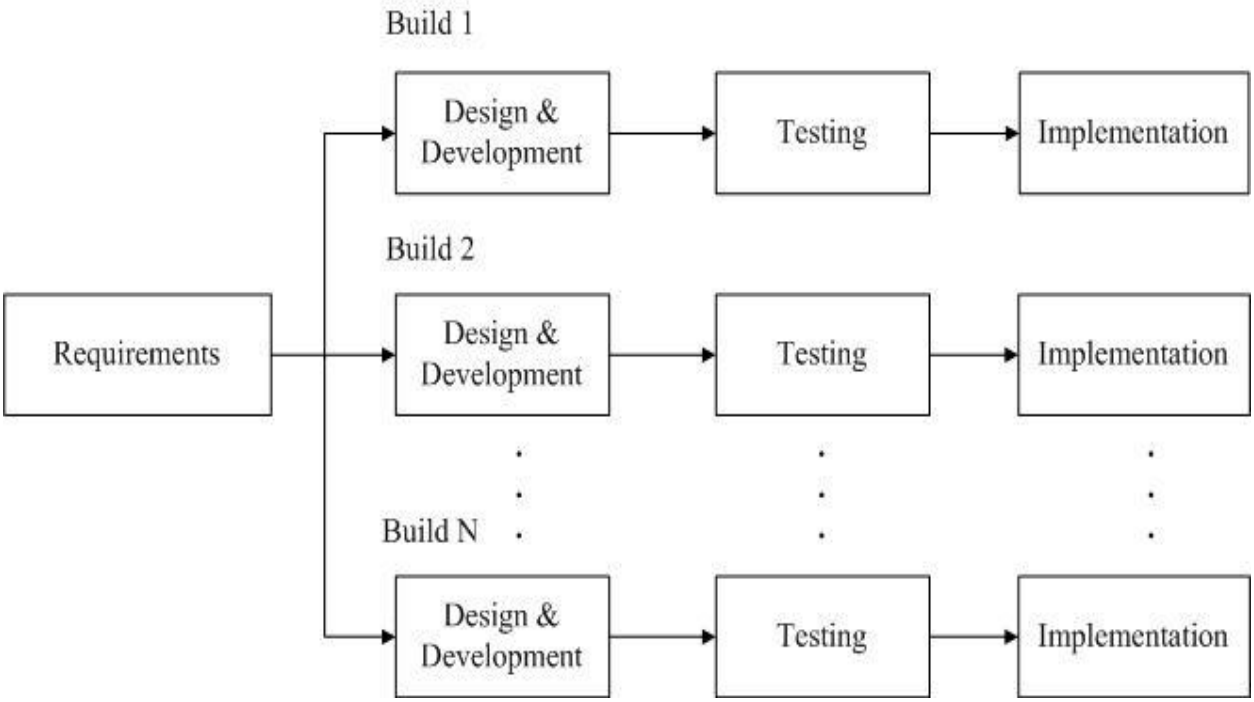


Figure 3.1. Incremental model

The various phases of incremental model are as follows:

Requirement analysis:

In the first phase of the incremental model, the product analysis expertise identifies the requirements. To develop the software under the incremental model, this phase performs a crucial role.

Design and development:

In this phase of the Incremental model of SDLC, the design of the system functionality and the development method are finished. When the software develops new practicality, the incremental model uses the design and development phase.

Testing:

In the incremental model, the testing phase checks the performance of each existing function as well as additional functionality. In the testing phase, the various methods are used to test the behavior of each task.

Implementation:

Implementation phase enables the coding phase of the development system. It involves the final coding that design in the designing and development phase and tests the functionality in the testing phase. After completion of this phase, the number of the product working is enhanced and upgraded up to the final system product.

3.2. BLOCK DIAGRAM

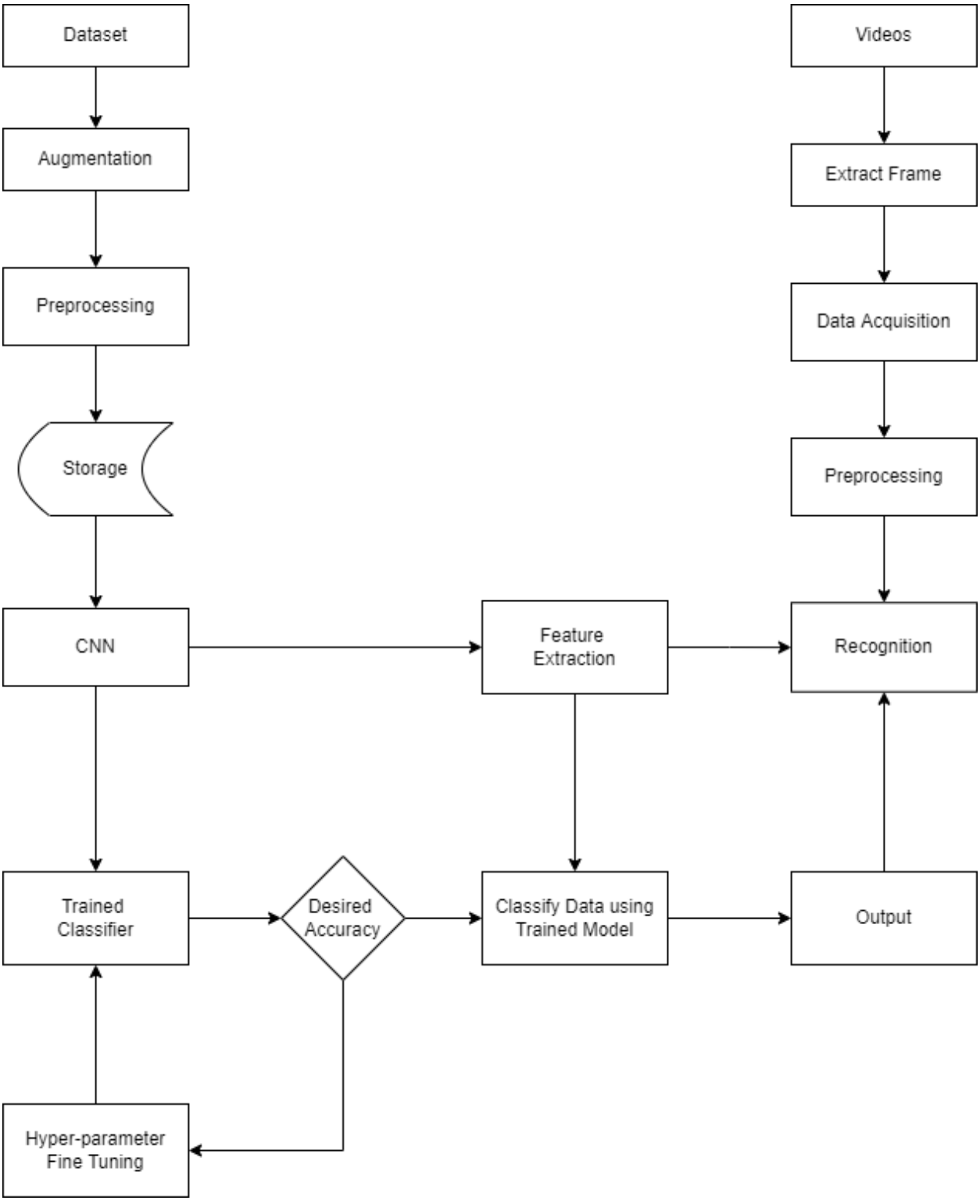


Figure 3.2. Block Diagram

Dataset collection:

The first step of the proposed system is to collect data. Data collection is indelibly an essential part in this research as our result highly depends on it.

We have created our own dataset having 3000 images of alphabets a to z each along with space, de. Gestures of different letters including space, delete were taken with the help of a webcam. To replicate different conditions like lighting and background, each alphabetical gesture has been performed 70 times in alternate lighting conditions.

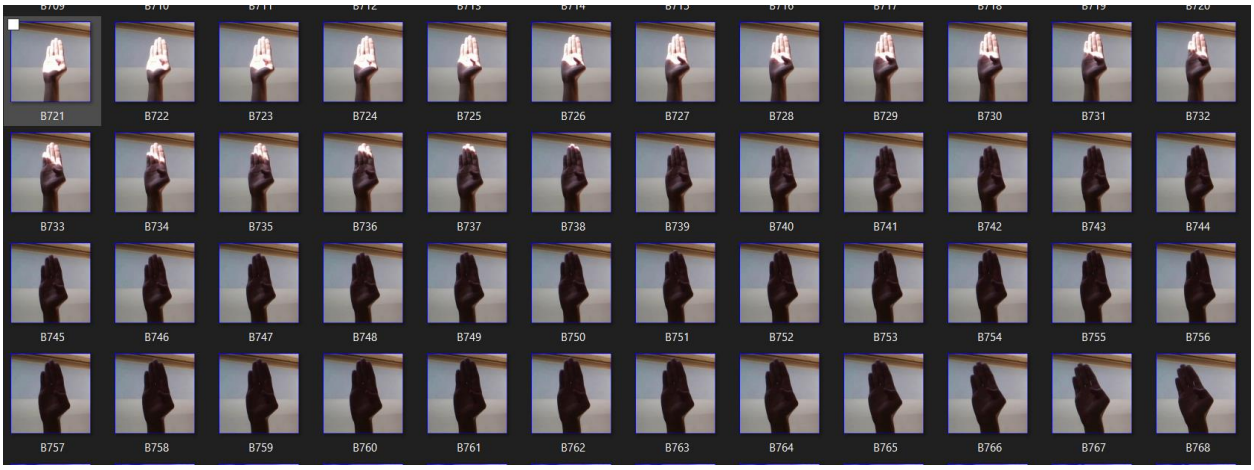


Fig 3.3. Dataset of Sign ‘b’

Data Augmentation:

Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset. Since training deep learning neural network models on more data can result in more skillful models, and the augmentation techniques can create variations of the images that can improve the ability of the fit models to generalize what they have learned to new images.

Preprocessing and storing the dataset:

The images obtained were resized for uniformity and consistency in data models so that image format remains same throughout the dataset during training the model. The dataset folder of gestures was further split into 2 more folders, one for training and the other for testing. Out of all the images captured, 2550 images were used for training and the rest 450 were used for testing. The images obtained were saved in the png format and there was no loss in quality.

Apply CNN:

The stored data was then fit into the Convolutional Neural Network (CNN). With the help of the dataset that had been prepared, the system was trained to recognize the pattern of the data that is acquired. In our proposed system, we apply a 2D CNN model with a tensor flow library. The convolution layers scan the images with a filter of size 3 by 3 and 5 by 5. The dot product between the frame pixel and the weights of the filter were calculated. This particular step extracted important features from the input image to pass on further. The pooling layers were then applied after each convolution layer. One pooling layer decrements the activation map of the previous layer. It merged all the features that were learnt in the previous layers' activation maps. This helped to reduce overfitting of the training data and generalized the features represented by the network.

```
CustomCNN(
  (conv1): Conv2d(3, 16, kernel_size=(5, 5), stride=(1, 1))
  (conv2): Conv2d(16, 32, kernel_size=(5, 5), stride=(1, 1))
  (conv3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1))
  (conv4): Conv2d(64, 128, kernel_size=(5, 5), stride=(1, 1))
  (fc1): Linear(in_features=128, out_features=256, bias=True)
  (fc2): Linear(in_features=256, out_features=29, bias=True)
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
)
```

Fig 3.4. CNN Model

Trained Classifier:

The data from the convolutional layer was extracted and classified based on the trained model. The convolutional layers have a ReLU activation. After layers of convolution and pooling the neural network were then flattened. Then a fully connected layer was applied. The loss function used is 'Cross Entropy' and the optimizer used for this training model was 'Adam'.

Checking accuracy and hyperparameter fine-tuning:

The accuracy of the data prediction based on our training set and testing sets had been checked. If the desired accuracy was not obtained from the system, it could be retrained with the process called hyperparameter fine-tuning. Hyperparameters are crucial as they control the overall behavior of a machine learning model. The ultimate goal is to find an optimal combination of hyperparameters that minimizes a predefined loss function to give better results. Here, tuning of parameters could be further done to improve the accuracy.

Classifying data using the trained model:

The trained model was ready to make predictions from the provided input data. Once a test data was provided to the trained model from our webcam, it would make predictions and the accuracy of the system was determined based on those predictions. This particular model predicted up to 29 classes which belonged to 26 different alphabets of the Sign Language along with special gesture for space and delete.

Data Acquisition:

Data acquisition is the process of sampling signals that measure real world physical conditions and converting the resulting samples into digital numeric values that can be manipulated by a computer. Here in our system, we acquire the data to be predicted using the webcam.

Extract frames:

Extraction of frames refers to the process of extracting the images from the video. The frames were extracted one by one in a short duration of time such that the user had ample time to make hand gestures.

Preprocessing the acquired data:

We perform segmentation where the image is then transformed to grayscale, this will enhance the robustness of our system to changes in lighting or illumination. Non-black pixels in the transformed image are binarized while the others remain unchanged, therefore black. The hand gesture is segmented firstly by taking out all the joined components in the image and secondly by letting only the part which is immensely connected. The frame is then resized to a certain pixel size. At the end of the segmentation process, binary images of a certain size were obtained where the area in white represented the hand gestures and the remaining area was black.

Feature Extraction:

One of the most crucial parts in image processing is to select and extract important features from an image. Feature extraction helped reducing the data after having extracted the important features automatically. It also contributes in maintaining the accuracy of the classifier and simplifies its complexity. In this part, the network will perform a series of convolutions and pooling operations during which the features are detected. Here, the network would recognize fingers, hand, and edges from the input hand gesture.

Recognition:

After the system is trained and data is classified according to the model, it is ready for making predictions. The necessary features from the input frame of the system are extracted then it is compared with the output of the model and thus we obtain the prediction for the corresponding hand gesture obtained from the webcam.

3.3 ALGORITHM

3.3.1. CONVOLUTIONAL NEURAL NETWORK

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. 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

The CNN architecture includes several building blocks, such as convolution layers, pooling layers, and fully connected layers. A typical architecture consists of repetitions of a stack of several convolution layers and a pooling layer, followed by one or more fully connected layers. The step where input data are transformed into output through these layers is called forward propagation. Although convolution and pooling operations described in this section are for 2D-CNN, similar operations can also be performed for three-dimensional (3D)-CNN.

Convolutional

When programming a CNN, each convolutional layer within a neural network should have the following attributes: Input is a tensor with shape (number of images) x (image width) x (image height) x (image depth). Let's assume that the input will be a color image, which is made up of a matrix of pixels in 3D. This means that the input will have three dimensions—a height, width, and depth—which correspond to RGB in an image. We also have a feature detector, also known as a kernel or a filter, which will move across the receptive fields of the image, checking if the feature is present. This process is known as a convolution. Convolutional kernels whose width and height are hyper-parameters, and whose depth must be equal to that of the image. Convolutional layers

convolve the input and pass its result to the next layer. This is similar to the response of a neuron in the visual cortex to a specific stimulus.

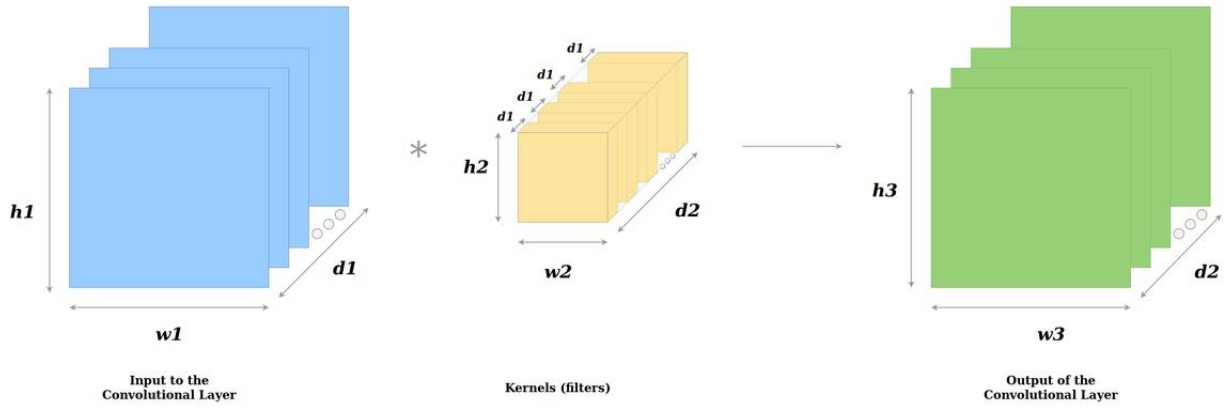


Figure 3.5. Convolutional Layer

Pooling

Convolutional networks may include local or global pooling layers to streamline the underlying computation. Pooling layers reduce the dimensions of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, typically 2 x 2. Global pooling acts on all the neurons of the convolutional layer. In addition, pooling may compute a max or an average.

1. Max pooling: As the filter moves across the input, it selects the pixel with the maximum value to send to the output array. As an aside, this approach tends to be used more often compared to average pooling.

2. Average pooling: As the filter moves across the input, it calculates the average value within the receptive field to send to the output array.

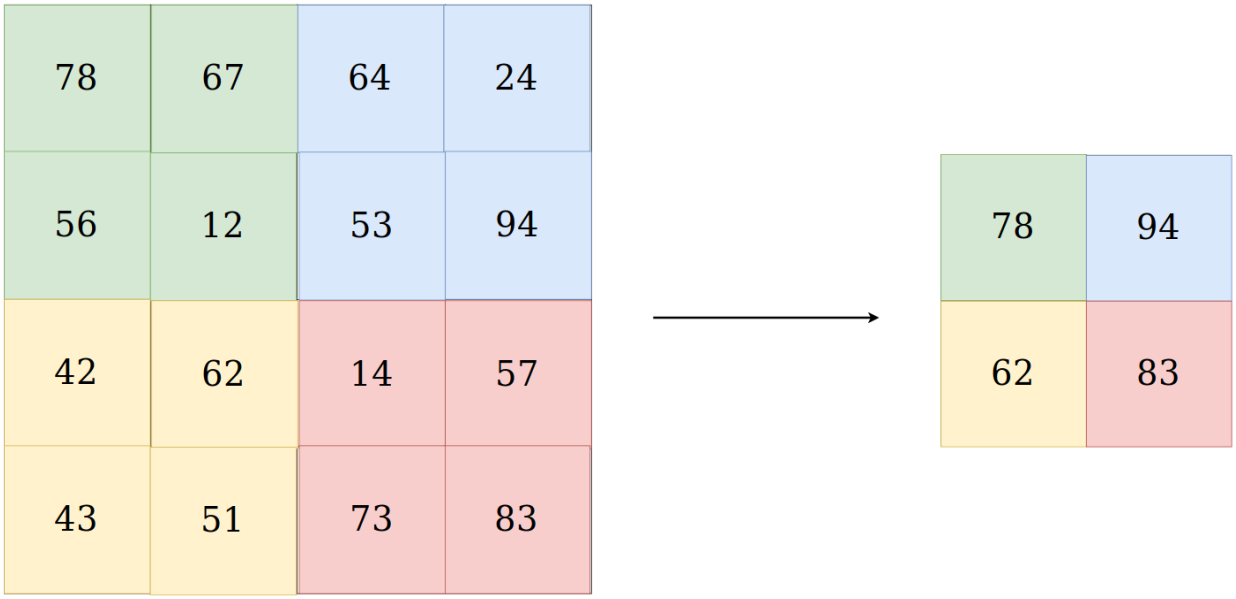


Figure 3.6. Max-Pooling

Fully connected

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images. The input to the fully connected layer is the output from the final Pooling or Convolutional Layer, which is flattened and then fed into the fully connected layer.

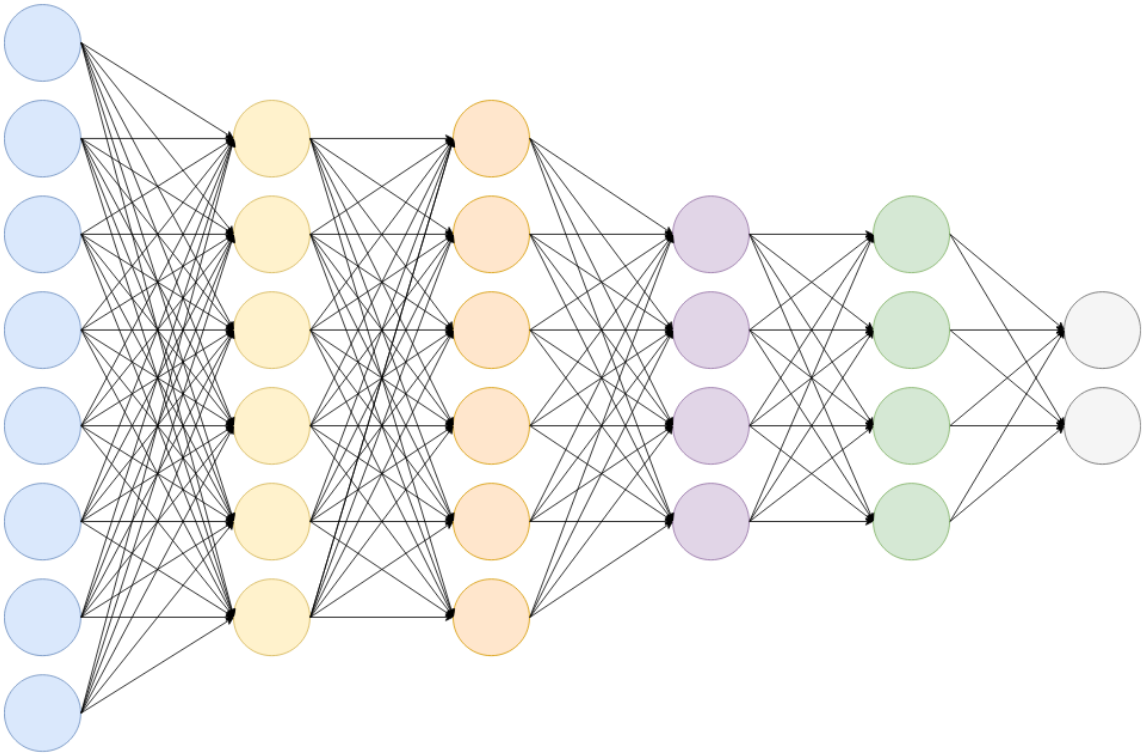


Figure 3.7. Fully Connected neural network

Receptive field

In neural networks, each neuron receives input from some number of locations in the previous layer. In a fully connected layer, each neuron receives input from every element of the previous layer. In a convolutional layer, neurons receive input from only a restricted subarea of the previous layer. Typically, the subarea is of a square shape (e.g., size 5 by 5). The input area of a neuron is called its receptive field. So, in a fully connected layer, the receptive field is the entire previous layer. In a convolutional layer, the receptive area is smaller than the entire previous layer.

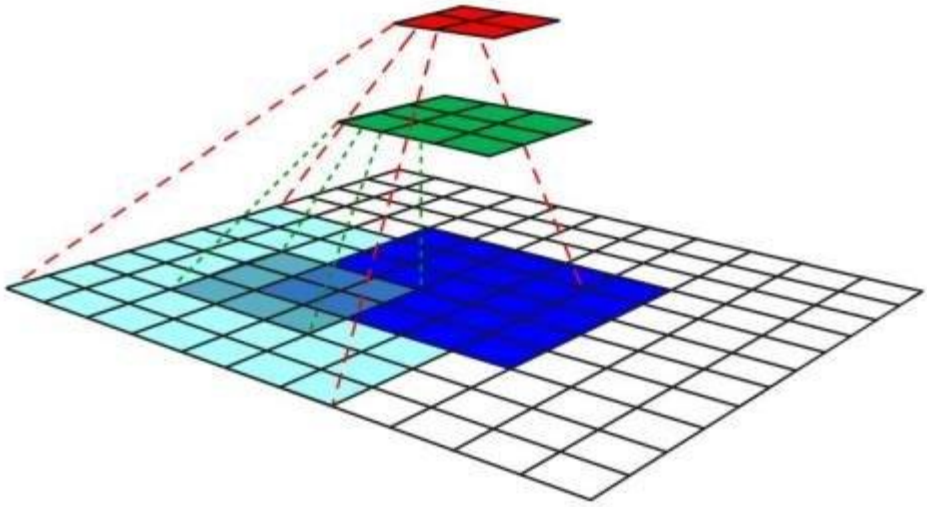


Figure 3.8. Receptive field

Weights

Each neuron in a neural network computes an output value by applying a specific function to the input values coming from the receptive field in the previous layer. The function that is applied to the input values is determined by a vector of weights and a bias (typically real numbers). Learning, in a neural network, progresses by making iterative adjustments to these biases and weights. The vector of weights and the bias are called filters and represent particular features of the input (e.g., a particular shape). A distinguishing feature of CNNs is that many neurons can share the same filter. This reduces memory footprint because a single bias and a single vector of weights are used across all receptive fields sharing that filter, as opposed to each receptive field having its own bias and vector weighting [11].

3.4. Use Case Diagram

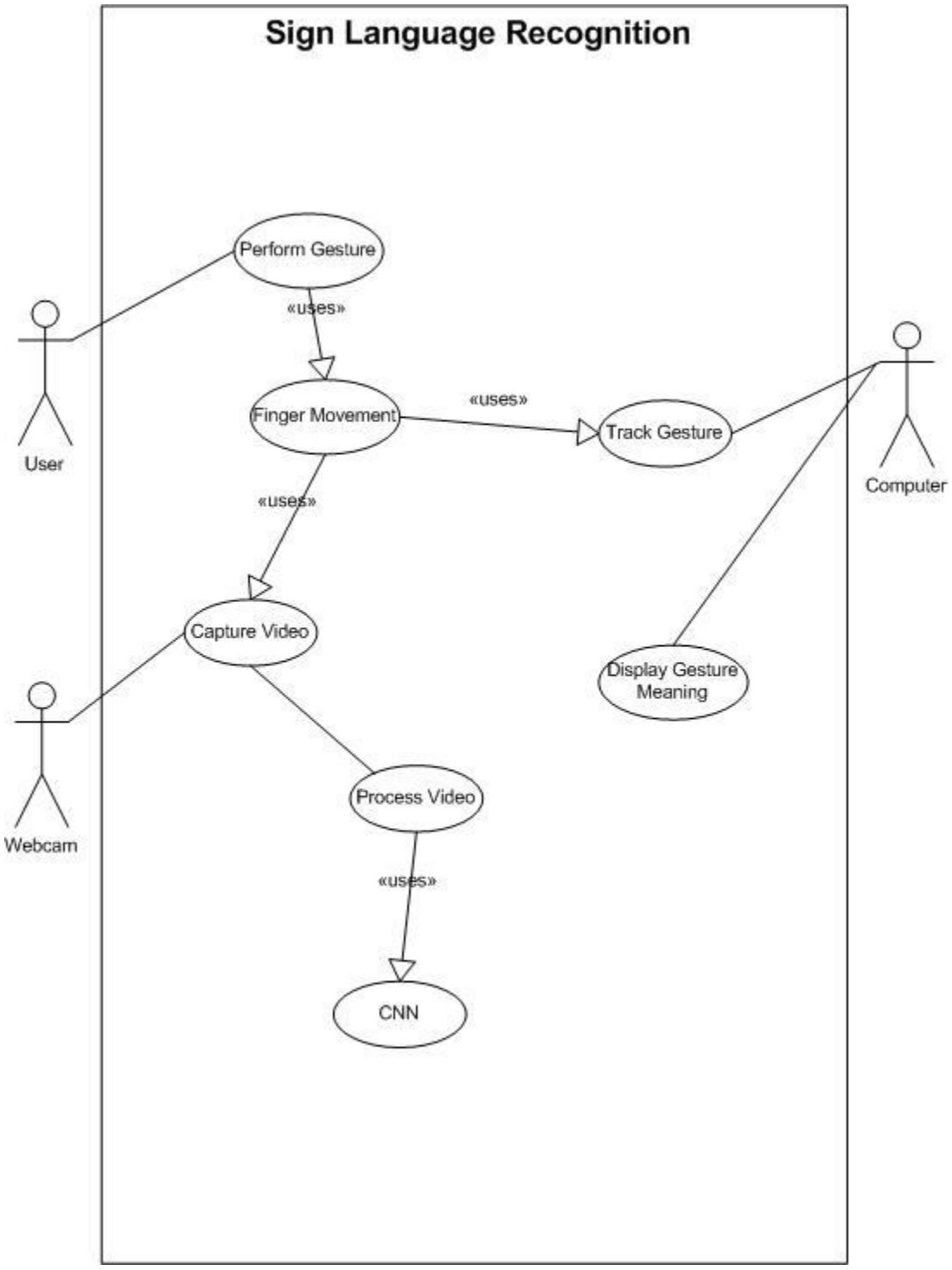


Fig 3.9. Use Case Diagram

3.5. Tools Used

1. Python

Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

2. Pytorch

PyTorch is an open source machine learning framework based on the Torch library, used for applications such as computer vision and natural language processing,primarily developed by Facebook's AI Research lab (FAIR). It is free and open-source software released under the Modified BSD license. Although the Python interface is more polished and the primary focus of development, PyTorch also has a C++ interface.

PyTorch uses a technique called reverse-mode auto-differentiation, which allows developers to modify network behavior arbitrarily with zero lag or overhead, speeding up research iterations.

a. torch.nn :

PyTorch uses modules to represent neural networks. Modules are:

- Building blocks of stateful computation. PyTorch provides a robust library of modules and makes it simple to define new custom modules, allowing for easy construction of elaborate, multi-layer neural networks.
- Tightly integrated with PyTorch’s autograd system. Modules make it simple to specify learnable parameters for PyTorch’s Optimizers to update.
- Easy to work with and transform. Modules are straightforward to save and restore, transfer between CPU / GPU / TPU devices, prune, quantize, and more.

b. torch.nn.functional :

Sub-module of PyTorch that is used to specify the functions of the neural network. It defines functions like Max Pooling, activation as well as convolution functions.

c. **torch.optim**:

It is a package implementing various optimization algorithms. Most commonly used methods are already supported, and the interface is general enough, so that more sophisticated ones can be also easily integrated in the future. To use torch.optim you have to construct an optimizer object that will hold the current state and will update the parameters based on the computed gradients.

d. **torch.vision:**

The torchvision package consists of popular datasets, model architectures, and common image transformations for computer vision. Torchvision is a library for Computer Vision that goes hand in hand with PyTorch. It has utilities for efficient Image and Video transformations, some commonly used pre-trained models, and some datasets .

3. Albumentation

Albumentation is a fast image augmentation library and easy to use with other libraries as a wrapper. The package is written on NumPy, OpenCV, and imgaug. What makes this library different is the number of data augmentation techniques that are available. While most of the augmentation libraries include techniques like cropping, flipping, rotating and scaling, albumentation provides a range of very extensive image augmentation techniques like contrast, blur and channel shuffle.

4. Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatterplots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

5. Imutils

A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and both Python 2.7 and Python 3.

6. Numpy

NumPy is a Python library that provides a simple yet powerful data structure: the n-dimensional array. This is the foundation on which almost all the power of Python's data science toolkit is built, and learning NumPy is the first step on any Python data scientist's journey. NumPy is the fundamental package for scientific computing with Python. It contains among other things:

• a powerful N-dimensional array object

• sophisticated (broadcasting) functions

• tools for integrating C/C++ and Fortran code

• useful linear algebra, Fourier transform, and random number capabilities.

Besides its obvious scientific uses, NumPy can also be used as an efficient multidimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases

7. OpenCV

OpenCV (Open Source Computer Vision Library) 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 (which was later acquired by Intel). The library is cross-platform and free for use under the open-source Apache 2 License. Starting in 2011, OpenCV features GPU acceleration for real-time operations.

8. ArgParse

The argparse module makes it easy to write user-friendly command-line interfaces. The program defines what arguments it requires, and argparse will figure out how to parse those out of sys.argv. The argparse module also automatically generates help and usage messages and issues errors when users give the program invalid arguments.

9. Joblib

Joblib is a set of tools to provide lightweight pipelining in Python. In particular: transparent disk-caching of functions and lazy re-evaluation (memoize pattern) and easy simple parallel computing. Joblib is optimized to be fast and robust on large data in particular and has specific optimizations for numpy arrays.

10. Tqdm

Tqdm is a library in python which is used in creating Progress Meters or Progress Bars. tqdm got its name from the Arabic name taqaddum which means ‘progress’. Implementing tqdm can be done effortlessly in our loops, functions or even Pandas. Progress bars are pretty useful in Python because:

- One can see if the kernel is still working
- Progress Bars are visually appealing to the eyes
- It gives Code Executive Time and Estimated Time for the code to complete which would help while working on huge datasets.

3.6. Verification and Validation

The hand tracking module was able to track the coordinates of the hand and then display the respective coordinates.

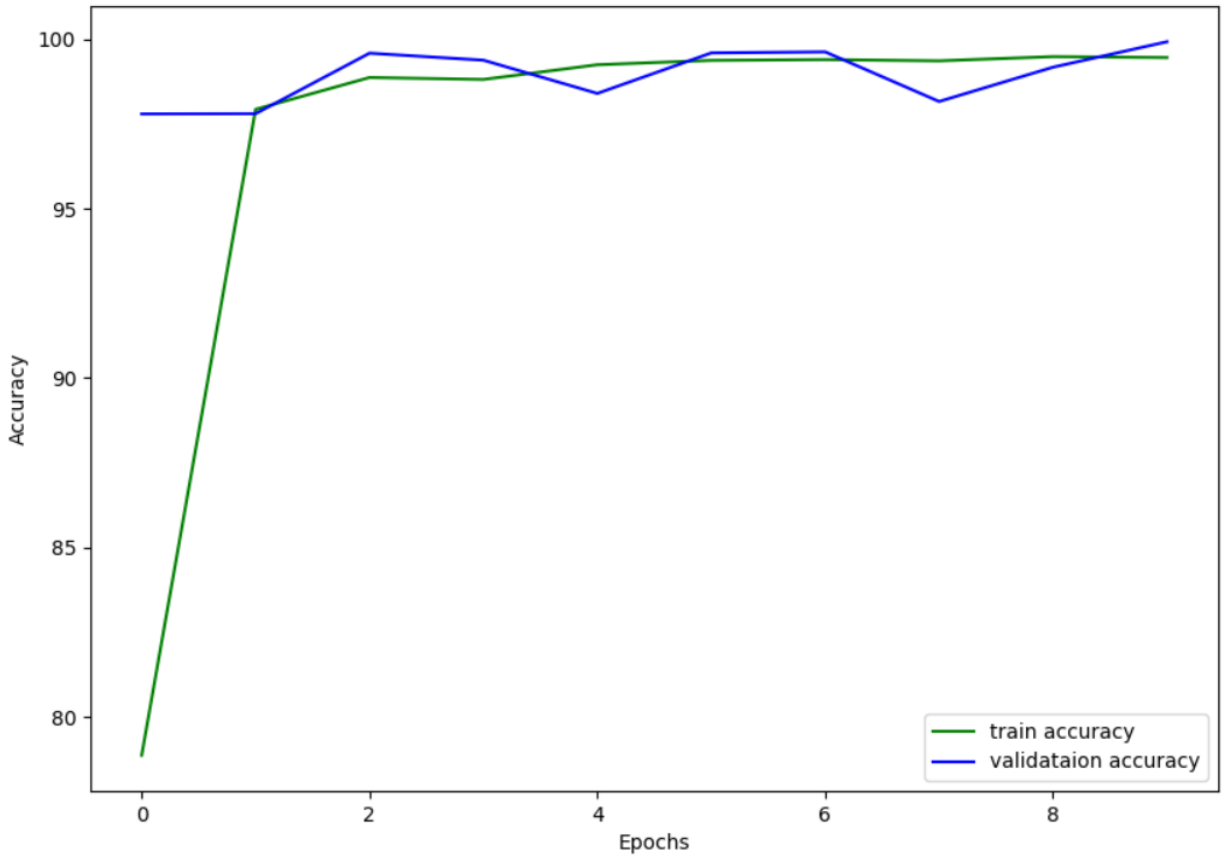


Fig 3.10. Train and Validation Accuracy

Training accuracy is usually the accuracy obtained when the model is applied on the training data whereas the test (or testing) accuracy often refers to the validation accuracy, that is, the accuracy calculated on the data set not used for training, but used (during the training process) for validating (or "testing"). This model has an accuracy of ~99% on the training set and ~99% on the validation set. This means that the model is able to perform with ~99% accuracy on new data.

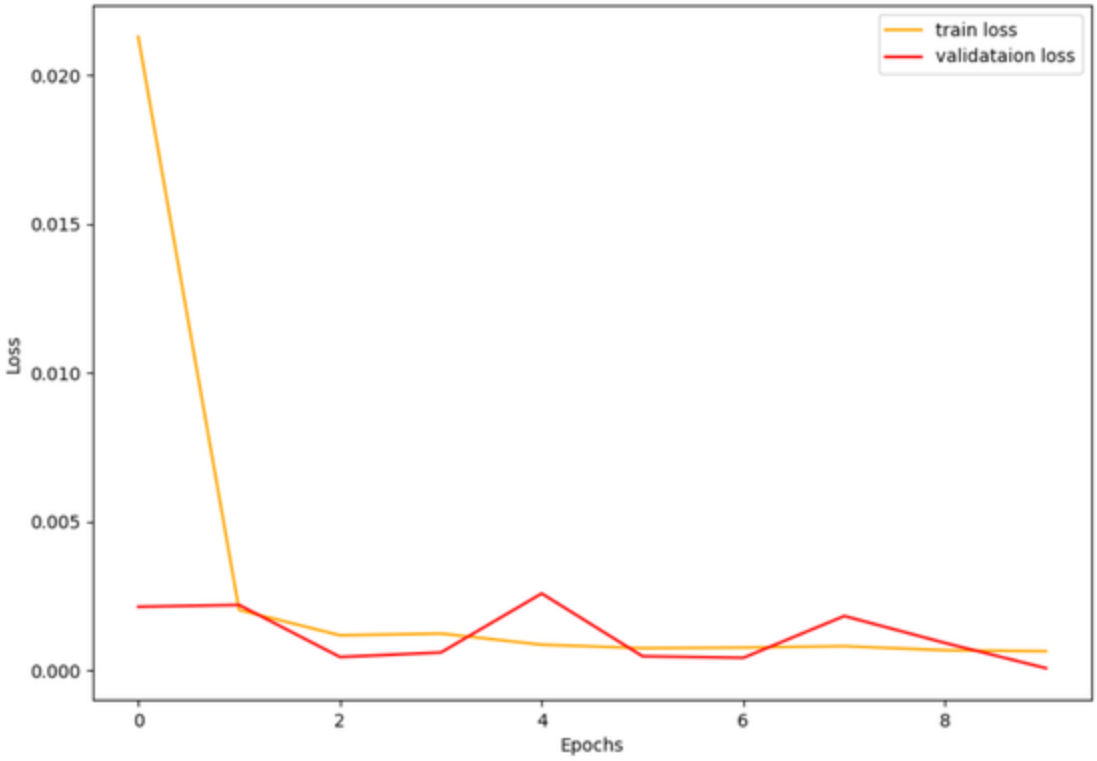


Fig 3.11. Train and Validation Loss

The training loss indicates how well the model is fitting the training data, while the validation loss indicates how well the model fits new data.

At the moment our model has a loss of 0.0006 on the training set and 0.0001 on the validation set. The model performs well on both training and validation data because augmentation was done on the training data.

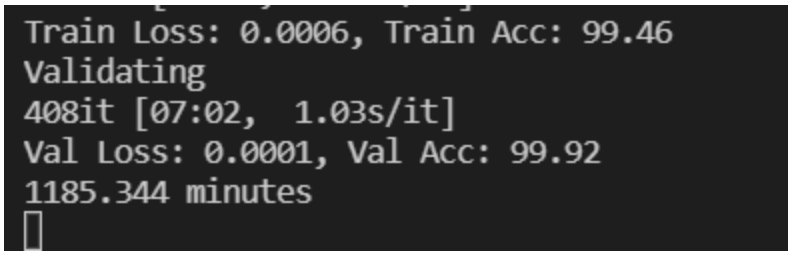


Fig 3.12. Training and validation results in 10th epoch

We can see that after the ten epochs, the training accuracy and loss values are following closely. This is a good thing for this neural network model. Both, the training and validation accuracy values remained above 99% at the end of training.

CHAPTER FOUR: EPILOGUE

4.1 RESULT AND CONCLUSION

We were able to build a Sign Language Detection model which was capable of transforming the hand gestures shown in the camera into the alphabets of sign language. The built model was able to predict the sign language accurately with an exemplary accuracy of ~99 percent. However, certain lighting and background condition may affect the predictions.

The primary objective to bridge the communication gap with hearing and speech impaired people which was achieved in a cost-effective way without the use of any expensive hardwares.

4.2 FUTURE ENHANCEMENT:

We aim to implement the following enhancements in future:

- Make the application recognize numbers as well as commonly used words such as ‘thank you’.
- Hardware implementation using external cameras for capturing images.

.

REFERENCES

[1] Sign Language to Text and Speech Translation in Real Time Using a Convolutional Neural Network. Ankit Ojha, Ayush Pandey, Shubham Maurya, Abhishek Thakur,Dr. Dayananda P. s.l. : International Journal of Engineering Research & Technology (IJERT), 2020.

[2] A review of hand gesture and sign language recognition techniques. Ming Jin Cheok, Zaid Omar & Mohamed Hisham Jaward. s.l. : International Journal of Machine Learning and Cybernetics volume , 2019.

[3] Research of a Sign Language Translation System Based on Deep Learning. He, Siming. s.l. : IEEE, 2019.

[4] Wearable Sensor-Based Sign Language. Karly Kudrinko, Emile Flavin, Xiaodan Zhu, and Qingguo Li.

[5] Evaluation of surface EMG features for the recognition of American Sign Language gestures. V. E. Kosmidou, L. J. Hadjileontiadis, and S. M. Panas. s.l. : Annu. Int. Conf. IEEE Eng. Med. Biol.

[6]"Isolated sign language recognition using hidden Markov models. Assan, K. Grobel and M. s.l. : " Syst. Man, Cybern.

[7] https://handtalk.me/en/blogpost/Index/?Id=4. [Online]

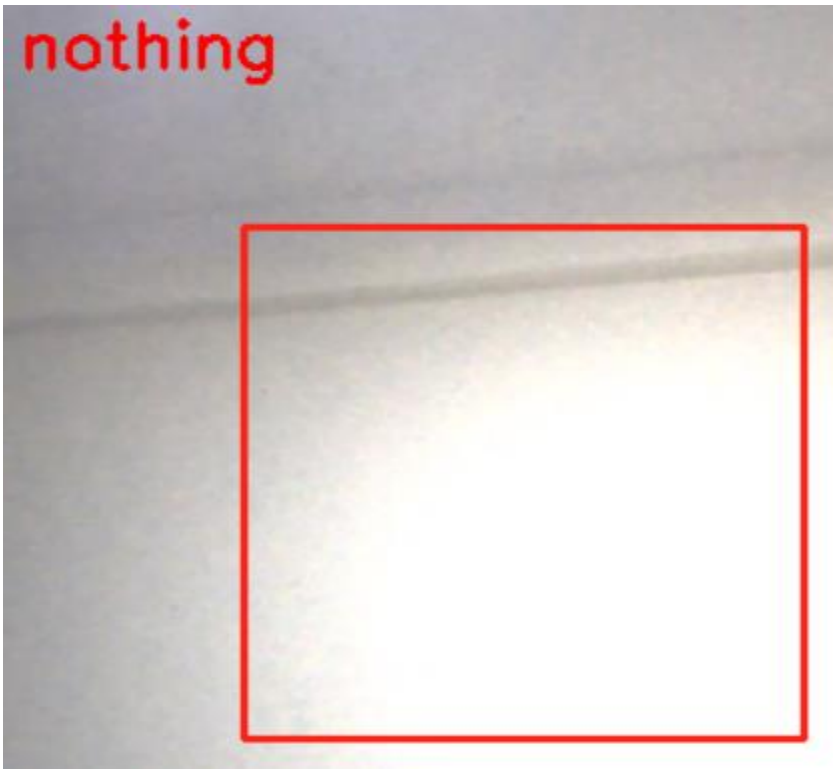
[8] https://www.signall.us/about-us. https://www.signall.us. [Online]

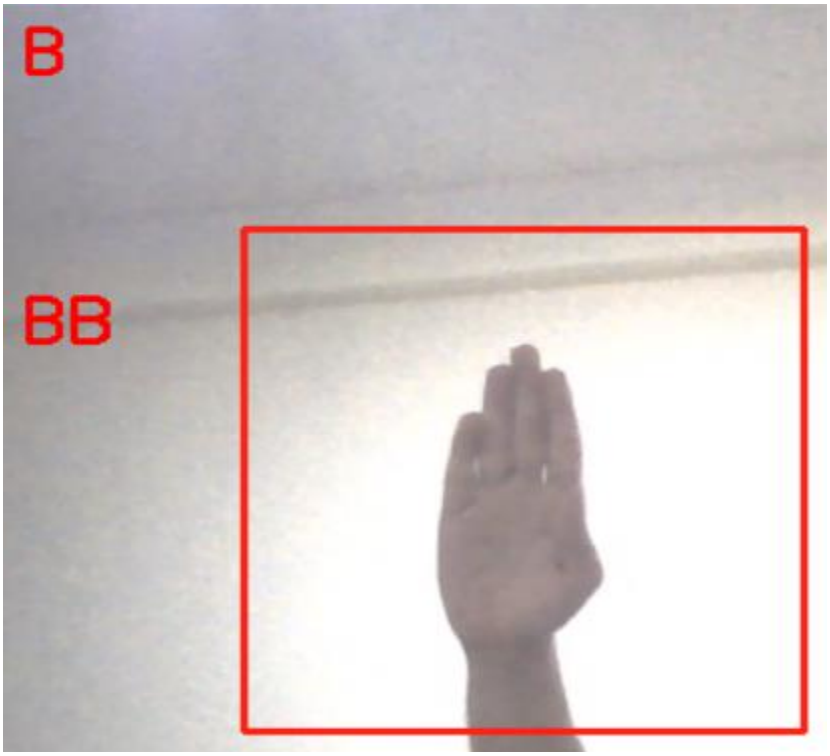
[9] https://slait.ai/. [Online]

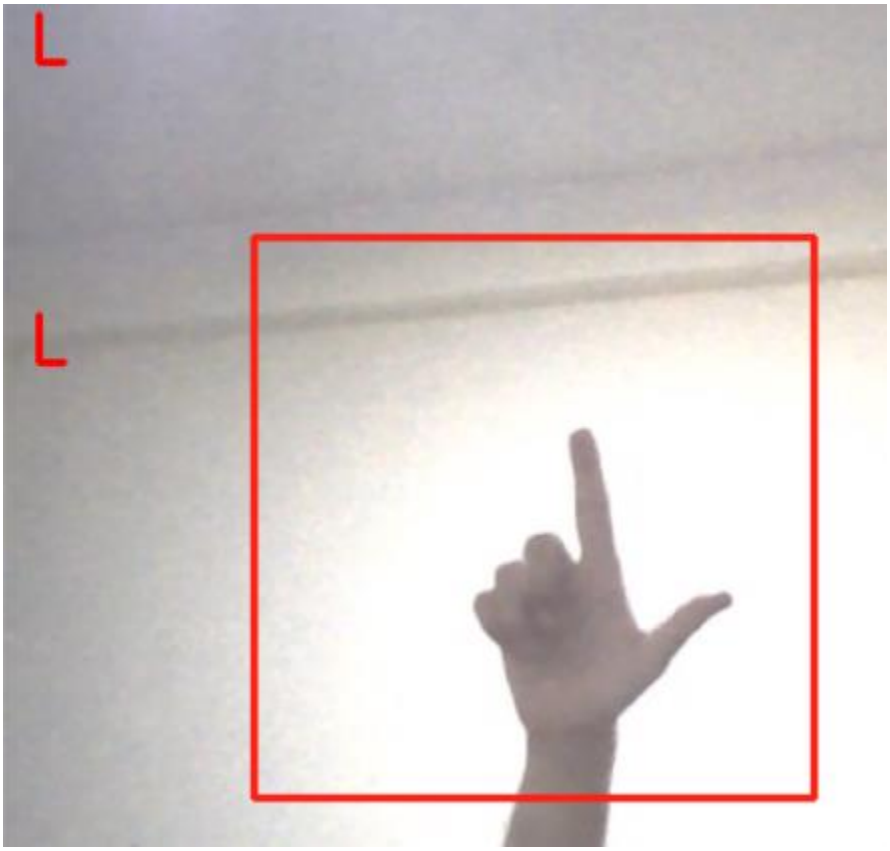
[10] https://en.wikipedia.org/wiki/Convolutional_neural_network. [Online]

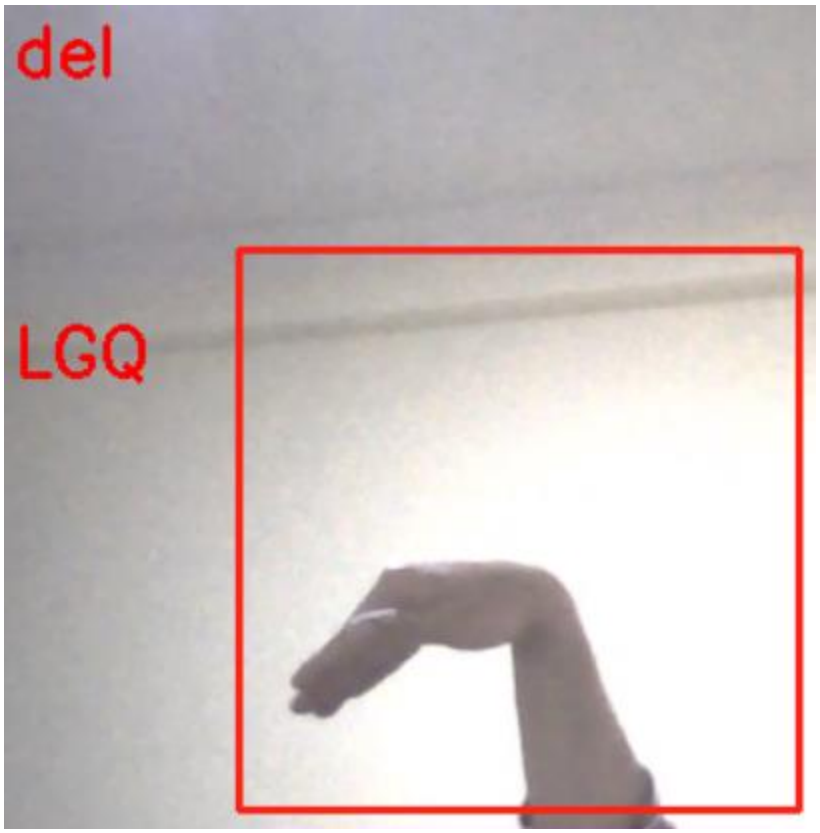
[11] Evaluation of surface EMG features for the recognition of American Sign Language gestures. V. E. Kosmidou, L. J. Hadjileontiadis, and S. M. Panas. s.l. : Annu. Int. Conf. IEEE Eng. Med. Biol. .

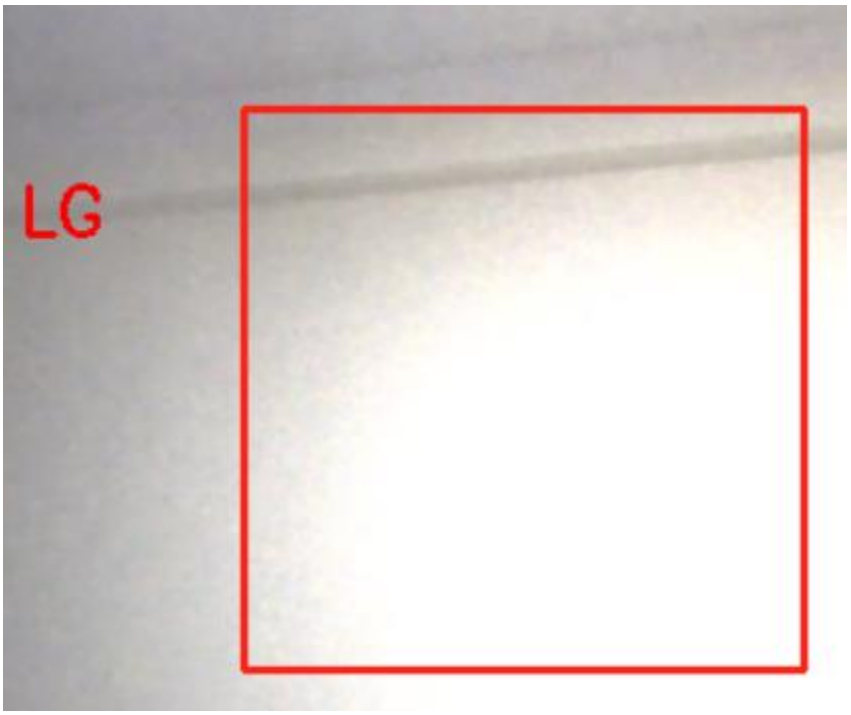
SCREENSHOTS




B
BB

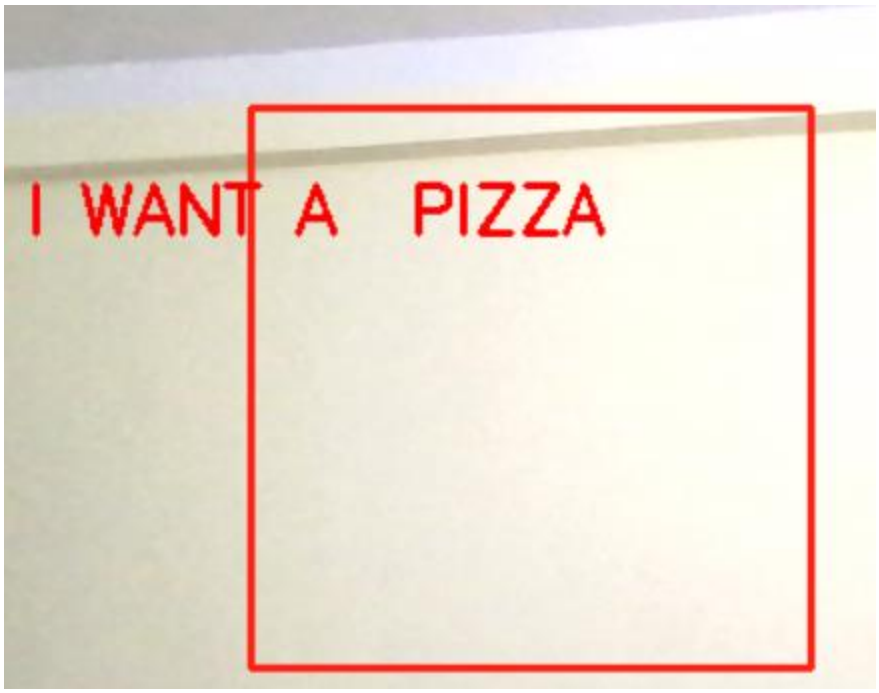

L
L


del
LGQ


LG


nothing
SANKALPA


SANKALPA IS A GOOD BOY


I WANT A PIZZA