# Real Time Indian Sign Language Recognition Using Nvidia Jetson Nano

## Minor Project


Submitted by:

**Ananya Kapoor (20103104)**

**Dhairya Sachdeva (20103098)**

**Sparsh Celly (20103102)**

Under the supervision of:

**Dr. Ashish Mishra**




**Department of CSE**

**Jaypee Institute of Information Technology University, Noida**


**December, 2022**

# DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and beliefs, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma from a university or other institute of higher learning, except where due acknowledgment has been made in the text.

Place: Jaypee Institute of Information Technology, Noida
Date: 2nd December, 2022

Name: Ananya Kapoor

Enrolment No.: 20103104

Name: Dhairya Sachdeva

Enrolment No.: 20103098

Name: Sparsh Celly

Enrolment No.: 20103102

# CERTIFICATE

This is to certify that the work titled “Real Time Indian Sign Language Recognition Using Nvidia Jetson Nano” submitted by Ananya Kapoor, Dhairya Sachdeva, and Sparsh Celly of B.Tech of Jaypee Institute of Information Technology, Noida has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of any other degree or diploma.

Digital Signature of Supervisor
Name of Supervisor
Designation
Date

## Abstract

Communication is very significant to human beings as it facilitates the spread of knowledge and forms relationships between people. We communicate through speech, facial expressions, hand signs, reading, writing or drawing etc. But speech is the most commonly used mode of communication. However, people with hearing and speaking disability only communicate through signs, which makes them highly dependent on non-verbal forms of communication. India is a vast country, having nearly five million people deaf and hearing impaired. Still very limited work has been done in this research area, because of its complex nature. Indian Sign Language is predominantly used in South Asian countries.

The purpose of this project is to recognize all the alphabets (A-Z) and digits (0-9) of Indian sign language using a machine learning model and convert them to text. The dataset for this system is created manually in different hand orientations and lighting. Pre-processing of images has been done for better results. Our model has then been deployed on Nvidia Jetson Nano, thus allowing us to work in real time.

# ACKNOWLEDGEMENT

I would like to place on record my deep sense of gratitude to Dr. Ashish Mishra, Jaypee Institute of Information Technology, India for his generous guidance, help and useful suggestions.

Ananya Kapoor (20103104)

Dhairya Sachdeva (20103098)

Sparsh Celly (20103102)

# List of Figures

# Abbreviations

| | |
|---|---|
| CNN | Convolution Neural Network |
| DHH | Deaf or Hard-of-Hearing |
| ISL | Indian Sign Language |

# Table of Contents

# Chapter 1: INTRODUCTION

Communication is very crucial to human beings, as it enables us to express ourselves. We communicate through speech, gestures, body language, reading, writing or through visual aids, speech being one of the most commonly used among them. However, unfortunately, for the speaking and hearing-impaired minority, there is a communication gap. Visual aids, or an interpreter, are used for communicating with them. However, these methods are rather cumbersome and expensive, and can't be used in an emergency.

Sign Language chiefly uses manual communication to convey meaning. This involves simultaneously combining hand shapes, orientations and movement of the hands, arms or body to express the speaker's thoughts.

Sign Language consists of fingerspelling, which spells out words character by character, and word level association which involves hand gestures that convey the word meaning. Fingerspelling is a vital tool in sign language, as it enables the communication of names, addresses and other words that do not carry a meaning in word level association. In spite of this, fingerspelling is not widely used as it is challenging to understand and difficult to use. Moreover, there is no universal sign language and very few people know it, which makes it an inadequate alternative for communication. Hence, we have developed a system which in real time would recognise the gestures of the deaf and hard of hearing people. Our model runs in real time using Nvidia Jetson Nano.

# Chapter 2: LITERATURE REVIEW

Little work has been done previously on ISL. One of the approaches included key point detection of Image using SIFT and then matching the key point of a new image with the key points of standard images per alphabet in a database to classify the new image with the label of one with the closest match [3]. Another one calculated the eigen vectors of covariance matrix calculated from the vector representation of image and used Euclidean distance of new image eigen vector with those in training data set to classify new image [1]. Some of them used Neural networks for training but their dataset comprised of only single-handed images and their feature vectors are based on the angle between fingers, number of fingers [2]. The major problem with all these works that there was no mention of where the dataset was collected from and from the images it appeared as if it was taken from webcam by the people working themselves.

Sanil Jain and KV Sameer Raja [4] worked on Indian Sign Language Recognition, using coloured images. They used feature extraction methods like bag of visual words, Gaussian random and the Histogram of Gradients (HoG). Three subjects were used to train SVM, and they achieved an accuracy of 54.63% when tested on a totally different user.

# Chapter 3: BACKGROUND STUDY

## 3.1 SIGN LANGUAGE

Sign languages are developed primarily to aid deaf and dumb people. Sign language involves non-vocal communication with a combination of hand movements, lip patterns and facial expressions in order to identify as a meaningful expression. They use a concurrent and specific combination of hand movements, hand shapes and orientation in order to convey particular information. One such set of language is the Indian Sign Language (ISL) system which is predominantly used in south Asian countries. Certain aspect that distinguishes ISL from other sign languages is that ISL devoid of any temporal inflections in its finger spelling chart.

According to WHO around 466 million people in world are dumb or deaf i.e., approximately 5% of total's world population. Sign Language is a language like any other globally recognized language. It is a non-verbal means of communication used by the deaf and mute to communicate amongst each other and with others. It allows people to express themselves and understand each other without speaking. There are many different sign languages, for example Indian Sign Language, American Sign Language, Chinese Sign Language.

## 3.2 GESTURE RECOGNITION

In gesture recognition technology a camera reads the movements of the human body to communicate with the device that uses gestures as input to control the functions of that device.

The basic steps involved in hand gesture recognition as shown in Figure 2.1 are:

1. Image Acquisition: User input is captured as a single image or sequence of frames with single camera.
2. Pre-processing: Pre-processing is the second phase of hand gesture recognition. In this process, the input image is divided into regions separated by boundaries by eliminating the noise.
3. Feature Extraction: Relevant data from the input are extracted as feature vectors and these features can be used for classification instead of using the entire input data.
4. Classification: It is the process of separating the features collected into predefined categories. Proper selection of the feature set and recognition algorithms can

result in good performance of the classifier (accuracy in terms of identifying a gesture).

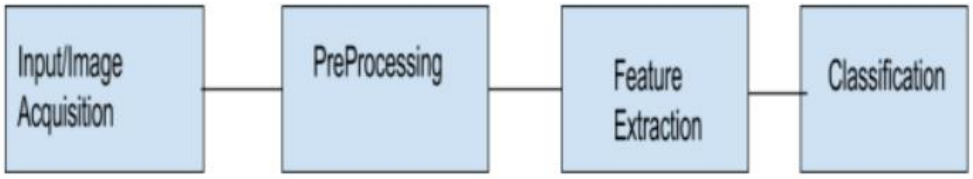


Figure 2.1 Basic Steps Involved in Gesture Recognition.

## 3.3 MACHINE LEARNING

Machine learning is defined as the methodology that can "detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty". It has been a significant subfield of Computer Science derived from pattern recognition and learning methodologies in Artificial Intelligence. n this research work, we have generated the input dataset from different users by extracting the joint information with the help of the Intel RealSense camera. A collection of feature sets should be extracted from the data in order to match the prediction process for the expected output. Selecting the dataset that is significant to the study out of huge amount of data is a critical task till date in Machine Learning. Once the feature set is selected the whole dataset is divided into different percentages of training and testing datasets. The training dataset is used to build the classification model to learn from the labelled data, and the classifier will be able to make predictions based on the given input samples. During the training phase, the classifier tries to learn from features as well as the labels and uses this learning in the testing phase to predict the samples based on the previous knowledge it has gained. These predictions are compared to the actual output to verify the accuracy.

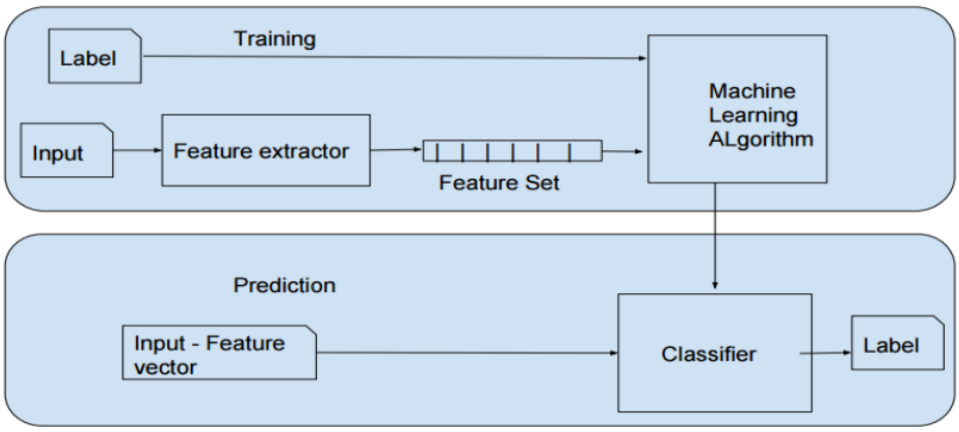


Figure 2.2 Supervised Learning Model Workflow

## 3.4 CLASSIFIERS

Classification is the process of categorizing samples into categories that they belong to. Classifiers are the algorithms that learn from the previous observations and use that knowledge to do further classification. Classification can be of two types: First, binary classification categorizes the samples into two classes, while multi-class classification categorizing the samples to multiple classes. We have used the Cvzone Classifier in our work.

## 3.5 NEURAL NETWORKS

A neural network is a series of algorithms that endeavours to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature. Neural networks can adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria. The concept of neural networks, which has its roots in artificial intelligence, is swiftly gaining popularity in the development of trading systems. A neural network works similarly to the human brain's neural network. A "neuron" in a neural network is a mathematical function that collects and classifies information according to a specific architecture. The network bears a strong resemblance to statistical methods such as curve fitting and regression analysis. A neural network contains layers of interconnected nodes. Each node is a perceptron and is similar to a multiple linear regression. The perceptron feeds the signal produced by a multiple linear regression into an activation function that may be nonlinear. In a multi-layered perceptron (MLP), perceptron's are arranged in interconnected layers. The input layer collects input patterns. The

output layer has classifications or output signals to which input patterns may map. Hidden layers fine-tune the input weightings until the neural network's margin of error is minimal. It is hypothesized that hidden layers extrapolate salient features in the input data that have predictive power regarding the outputs. This describes feature extraction, which accomplishes a utility similar to statistical techniques such as principal component analysis.

### 3.5.1 CONVOLUTION NEURAL NETWORKS

Convolutional neural networks (CNN) is a special architecture of artificial neural networks, proposed by Yann LeCun in 1988. CNN uses some features of the visual cortex. One of the most popular uses of this architecture is image classification. For example, Facebook uses CNN for automatic tagging algorithms, Amazon — for generating product recommendations and Google — for search through among users' photos. Instead of the image, the computer sees an array of pixels. For example, if image size is 300 x 300. In this case, the size of the array will be 300x300x3. Where 300 is width, next 300 is height and 3 is RGB channel values. The computer is assigned a value from 0 to 255 to each of these numbers. This value describes the intensity of the pixel at each point. To solve this problem the computer looks for the characteristics of the baselevel. In human understanding such characteristics are for example the trunk or large ears. For the computer, these characteristics are boundaries or curvatures. And then through the groups of convolutional layers the computer constructs more abstract concepts. In more detail: the image is passed through a series of convolutional, nonlinear, pooling layers and fully connected layers, and then generates the output.

## 3.6 CVZONE CLASSIFIER

Cvzone [5] is a computer vision package that makes it easy to run Image processing and AI functions. At its core it uses OpenCV and Mediapipe libraries.

## 3.7 GOOGLE TEACHABLE MACHINE

Teachable Machine [6] is a web-based tool that makes creating machine learning models fast, easy, and accessible to everyone. You train a computer to recognize your images, sounds, and poses without writing any machine learning code. Then, use your model in your own projects, sites, apps,

and more. You can currently train Teachable Machine with images (pulled from your webcam or image files), sounds (in one-second snippets from your mic), and poses (where the computer guesses the position of your arms, legs, etc from an image). Teachable Machine uses TensorFlow.js, a library for machine learning in Javascript, to train and run the models you make in your web browser. There's a pretrained neural network, and when you create your own classes, you can sort of picture that your classes are becoming the last layer or step of the neural net. Specifically, both the image and pose models are learning off of pretrained mobilenet models

## 3.8 REAL TIME VIDEO PROCESSING

Video data is a common asset which is used every day, whether it is a live stream in a personal blog or security camera in a manufacturing building. A machine learning appliance is becoming a normal tool for processing video in a variety of tasks.

In a nutshell, video processing can be seen as a sequence of operations done for each frame. Each frame includes processes of decoding, computation and encoding. Decoding is a conversion of the video frame from compressed format to the raw format. Computation is a certain operation which we need to do with the frame. And encoding is conversion of the processed frame back to the compressed state. This is a serial process. Although such an approach looks straightforward and easy to implement, it is not practical in most cases. It makes the above-mentioned approach a workable one, it is necessary to set measurement criteria such as, speed, accuracy and flexibility. However, to get all these in one system might be impossible. Usually there is a trade-off between speed and accuracy in such cases. If we choose not to pay attention to speed, we'll have a nice algorithm, but it will be extremely slow, and as a result, useless. If we simplify the algorithm too much, then the results may not be as good as expected. Consequently, a fast and accurate white elephant is impossible to integrate. The solution is supposed to be flexible in terms of input, output and configuration.

So, how to make the processing faster, while keeping the accuracy at a reasonably high level? There are two possible ways to solve the issue. The first one is to make the algorithms run in parallel, so they might be possible to keep using slower accurate models. The second one is to make a certain effort to accelerate the algorithms themselves or their parts with no significant loss of the accuracy.

## 3.9 NVIDIA JETSON NANO

NVIDIA® Jetson Nano™ Developer Kit is a small, powerful computer that lets you run multiple neural networks in parallel for applications like image classification, object detection, segmentation,

and speech processing. All in an easy-to-use platform that runs in as little as 5 watts. JetPack is compatible with NVIDIA's world-leading AI platform for training and deploying AI software, reducing complexity and effort for developers.


Figure 2.3 Nvidia Jetson Nano

JetPack enables end-to-end acceleration for your AI applications, with NVIDIA TensorRT and cuDNN for accelerated AI inferencing, CUDA for accelerated general computing, VPI for accelerated computer vision and image processing, Jetson Linux APIs for accelerated multimedia, and libArgus and V4l2 for accelerated camera processing.

NVIDIA container runtime is also included in JetPack, enabling cloud-native technologies and workflows at the edge. Transform your experience of developing and deploying software by containerizing your AI applications and managing them at scale with cloud-native technologies.

Jetson Linux provides the foundation for your applications with a Linux kernel, bootloader, NVIDIA drivers, flashing utilities, sample filesystem, and toolchains for the Jetson platform. It also includes security features, over-the-air update capabilities and much more.

## Chapter 4. REQUIREMENT ANALYSIS

We couldn’t find any reliable datasets on the internet as most of the videos of the sign language we found on the internet were of people who were demonstrating what sign language looked like and not those who actually spoke the language.
So, we created our own dataset which has approximately 300 images for letter A-Z.
Besides the dataset, our 4GB Jetson Nano Developer Kit had few dependencies, which included a 32GB MicroSD Card for the operating system and main storage, an Ethernet Cable to enable internet connection on the Jetson Nano, Logitech c270 USB Webcam to capture frames, A USB Data Cable (Micro-B to Type-A) to directly connect out laptops to the Jetson Nano in the beginning stages to enable headless device mode, a 5V 4A DC barrel jack power supply to power our device, a desktop, and an HDMI cable.

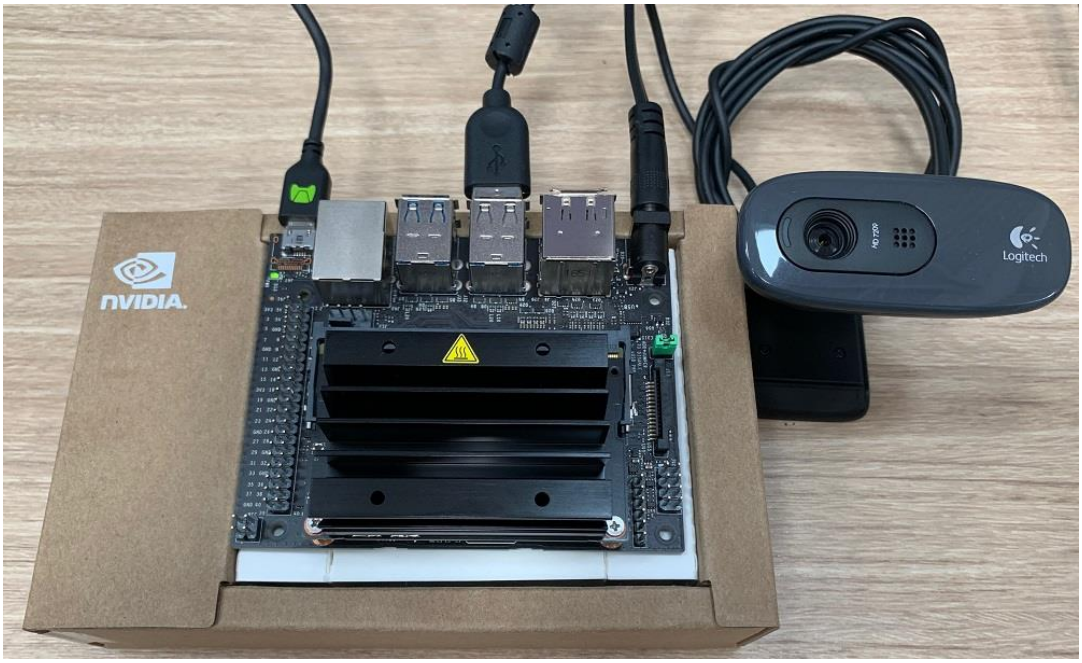


Figure 2.4 Setup

# Chapter 5. DETAILED DESIGN

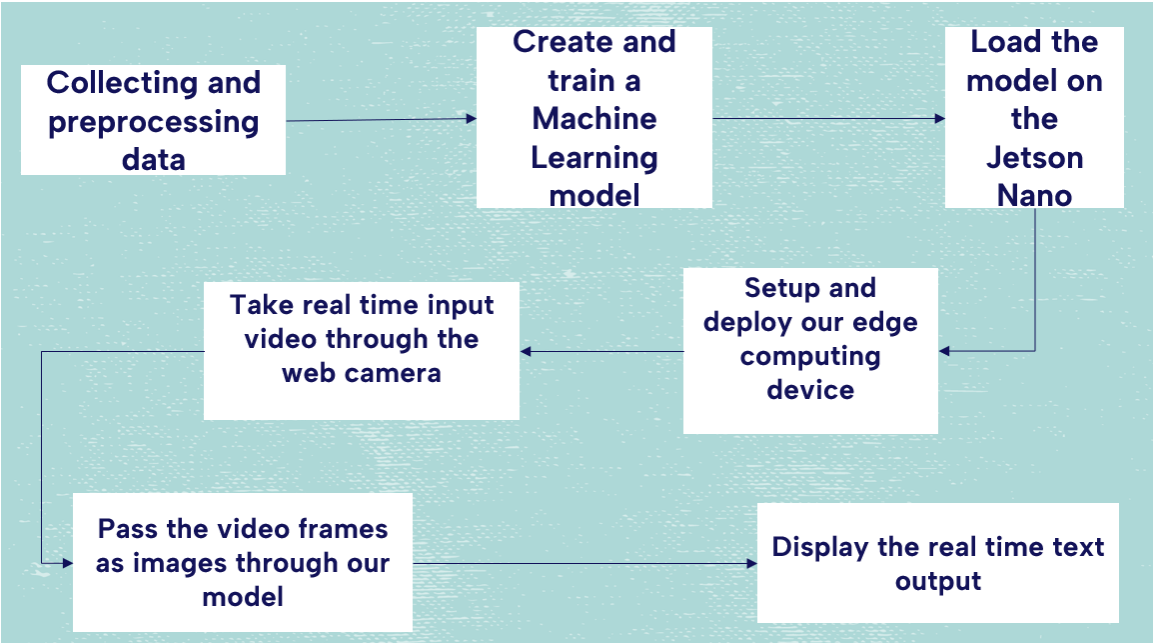


Figure 2.5 Workflow

Today the technology is evolved and has potential to help the dumb and deaf people. We propose a model that can recognize sign language. The predicted alphabets will be visible on the real time basis.

## 5.1 DATASET COLLECTION AND PREPROCESSING

We searched for sign language image datasets and found a few. But we needed a dataset that had a good amount of variation and which was recorded in different lighting conditions, to maximise our accuracy in different locations and subjects. Upon searching for some time, we realised that such a dataset wasn't available. So, the best solution was to create our own dataset for each and every sign.

We captured images of signs through our camera by writing a python script which tracked our hand, cropped and discarded the rest of the area. But since different signs had different shapes, this meant that different letters' height to width ratio would be different. To make the aspect ratio equal the same for all signs we added white padding to make all the images equal squares. On top of that to tackle the problem of different skin tones and background we needed to apply filters on the image. We converted the RBG image to grayscale image, applied Gaussian blur and used Adaptive Thresholding to get uniform data across for different subjects.

We collected about 320 images per sign, where we had unique signs for each alphabet and signs for backspace and spacebar.

Figure 2.6 Sample Pre-processed Image

## 5.2 TRAINING A MACHINE LEARNING MODEL

The aim of our project was to recognise the signs displayed in the live video feed. This is done by recognising the sign in each and every frame and then outputting the best match. The recognition of each frame is to be done by a machine learning model. Out of the many deep learning models, we found that Convolutional neural networks were the perfect fit to classify visual data. With the pre-processed data we had recorded, we could train a CNN model using Google's Teachable Machine. Teachable Machine is a platform where anyone can upload their audio or visual data and classify it using CNN on Google's allocated servers.

Upon many trial and error model specifications we found that we got the accuracy when we trained our 26 signs (320 images each) for 35 epochs, with a batch size of 32 and learning rate of 0.005. We exported the model in tensorflow keras format with .h5 file extension along with a text file containing the labels with their respective indexes. This model then classifies each frame and outputs the probabilities of each sign(label) and the index of the sign with the maximum probability.

## 5.3 DEPLOY THE MODEL ON JETSON NANO

Phase 1: Getting started with AI on Jetson nano- to get started with deep learning and AI on Jetson nano we completed a Nvidia course on the fundamentals of Jetson nano and how to initially setup the device and configure it to write Python code in it.

The Setup-for setting up the Jetson Nano we flashed a micro-SD card which is the storage unit for Jetson nano and from which we load up and boot our operating system (Ubuntu v18.04). Then we installed the JetPack on it which comes with a docker container containing all the necessary libraries and modules optimised to run Deep neural networks in real time. We also upgraded our swap file from 2 GB to 4GB to give us faster speed.

Hello AI- After successfully setting up our Nano we ran some test programs on the docket container in headless mode (without the monitor) on a remote laptop to learn about the flow of the Python scripts in the docker container.

Phase 2: open CV on Jetson nano-After setting up the device we started with open cv (cv2) also known as computer vision and other modules and packages like gstream and deepstream (present in the docker container).
We learnt about computer vision in depth as it is one of the main parts of our deployment and real time inference of the machine learning model.

Phase 3: Research-We did a lot of research regarding the issues faced by people while deploying their models for real time use on Jetson nano. Even installing of Python packages and dependencies on the nano were very in-depth and had to be done with a lot of care and after reading a lot of blogs and going through countless resources we figured out ways to install packages that we required to run the model.

Phase 4: Setting up for running our Python script on Jetson nano- As we had learnt about downloading Python packages in phase 3, now came the time to implement it. Firstly, we required Google's open-source framework-Mediapipe. We installed and built this framework on the Nano enabling us to track hands in our ML model and setting up a pipeline for our project. Next, we had to install and build one of the most important components of our project, Tensorflow (v2.5.0 gpu==enabled) which is an open-source software for ML and AI. These two were the most important part of our deployment process as it was only after this that we could run and test our models on the Jetson Nano.

Phase 5: Running and Testing of our model-

As we were done with the toughest part of the deployment process now came the time for actually running our Python script for creating a live camera object and taking frames from the live video and passing it through our model for output.
The model imported from Google's Teachable Machine is a Tensorflow-Keras (.h5) model for which we had earlier installed Tensorflow and built it with gpu support enabled. We have the gpu enabled as the Nano provides us with 128 CUDA cores for faster processing.

Phase 6: OS optimisations- after successfully running our model in the Jetson nano, optimisations had to be made in regard with both Time and Space (memory). For reducing the time, we reduced the size of our model to just (2.5 mb) and minimising our user interface at this time for better inference in real time. For reducing the memory usage, we deleted unnecessary files from Jetson's SD card (storage unit) and we reduced the number of processes running in background to reduce the main memory usage (RAM usage) and running our Python script.

Phase 7: Successfully deploying and Inferencing model in Real Time- After the struggles of new hardware and a new operating systems environment we overcame the problems we initially faced and successfully ran our model and inference the results in real time on an Edge Computing device which can run entirely on a 5V, 4A power source!

## Chapter 6. IMPLEMENTATION

As we had our trained model deployed on the jetson nano, we had to not only make predictions but also give the user a good and smooth experience wherein the user could communicate in full sentences in real time. To implement this, we create an empty string every time the script is run, and append and edit the string according to the users' needs and displayed the same on the live video stream. After all hardware dependencies like the mouse, keyboard, monitor(display) and camera are fulfilled, we can run our project in real time and test it.


Figure 2.7 Sample Output

# Chapter 7. EXPERIMENTAL RESULTS AND ANALYSIS

After trying various datasets, pre-processing techniques and model parameters we could achieve above 90% accuracy on the dataset. But what really mattered is the real time accuracy on different subjects and locations. We could successfully classify the signs most of the times. Some signs which were giving varying and inconsistent accuracy were alphabets which were very similar to the human eyes and resembled a closed fist (A, S, T, M, N). We could tackle this problem by providing good lighting conditions along the camera direction. It is also preferred that the signs are made away from the body and before a fairly blank background to avoid any visual noise and get the best predictions.

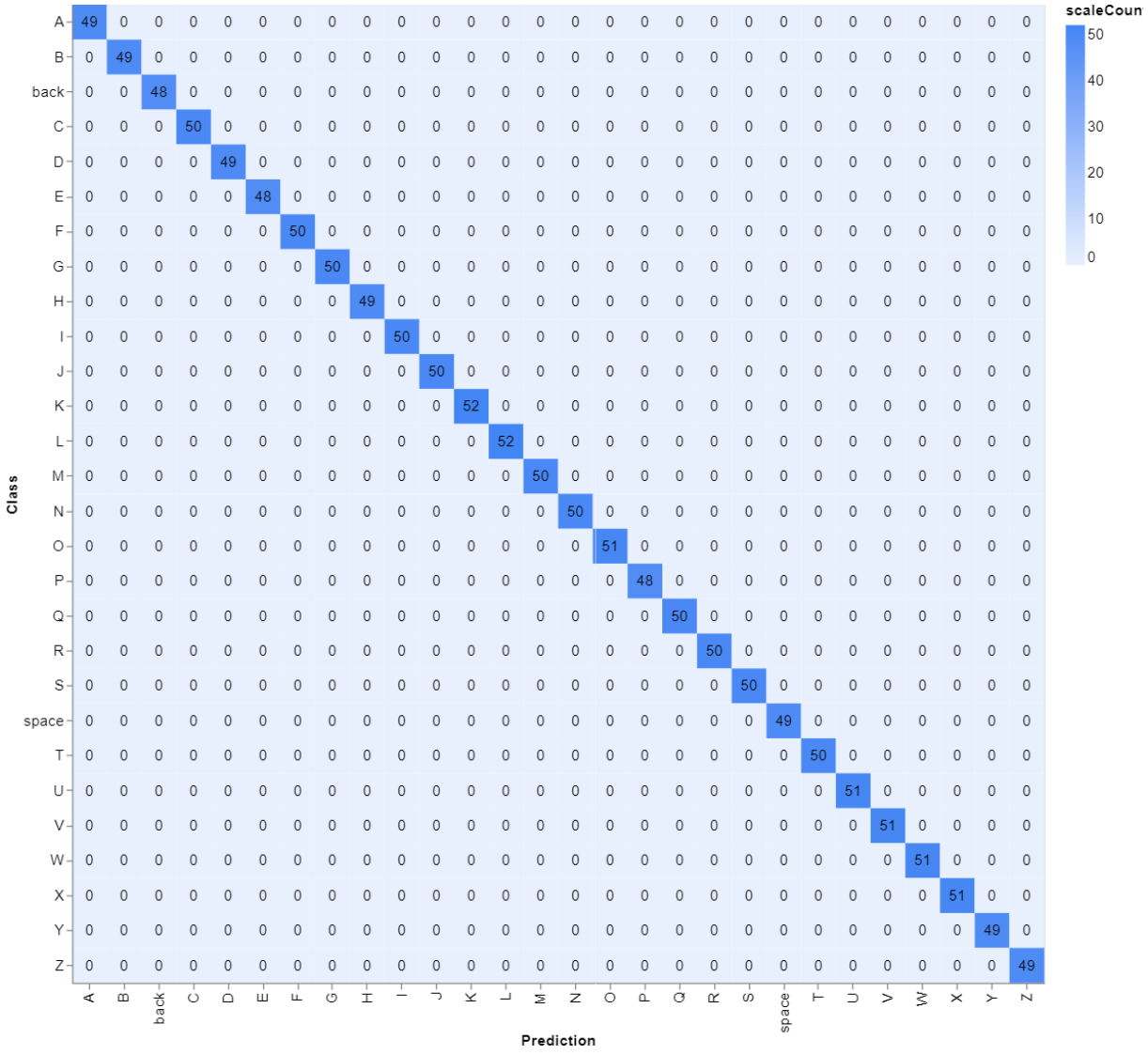


Figure 2.8 Confusion Matrix

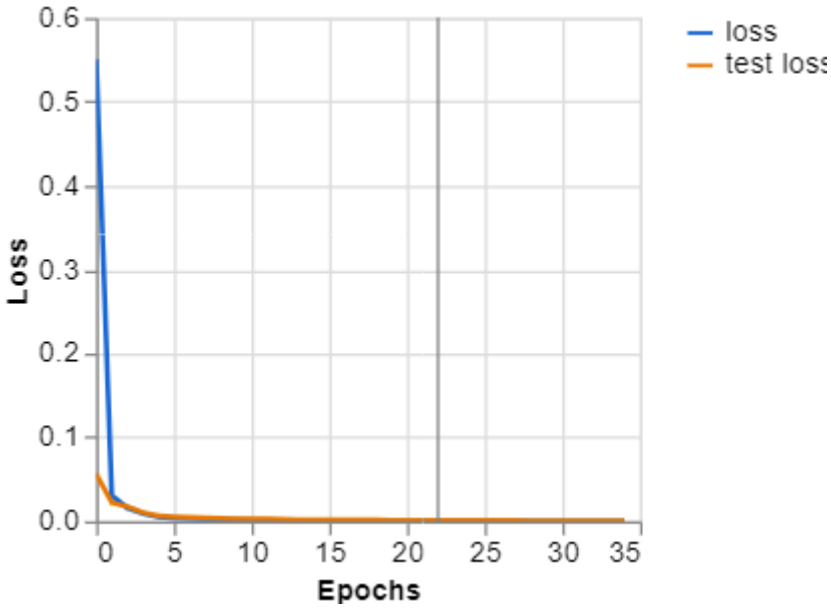


Figure 2.9 Loss vs. Epochs Graph

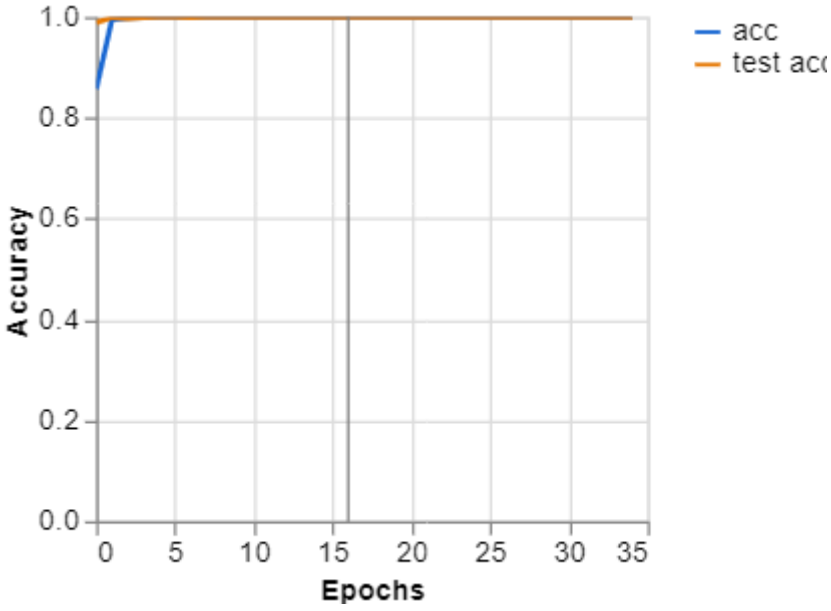


Figure 2.10 Accuracy vs. Epochs Graph

# Chapter 8. CONCLUSION AND FUTURE SCOPE

We successfully built an Indian sign language recognition system which runs in real time.
Scope of improvement and future work for us can include building a complete product that will help the speech and hearing-impaired people, and thereby reduce the communication gap.
This kind of model can also be used to build an application that can predict and autofill the words that the user is trying to communicate based on the previously interpreted signs by the model.

## References:

[1] Joyeeta Singh, K.D. Indian sign language recognition using eigen value weighted Euclidean distance-based classification technique. International Journal of Advanced Computer Science and Applications 4, 2 (2013).

[2] Padmavathi S, Saipreeth M.S, V. Indian sign language character recognition using neural networks. IJCA Special Issue on Recent Trends in Pattern Recognition and Image Analysis, RTPRIA (2013).

[3] Sakshi Goyal, Ishita Sharma. Sign language recognition system for deaf and dumb people. International Journal of Engineering Research Technology 2, 4 (April 2013).

[4] Sanil Jain, KV Sameer Raja. Indian Sign Language Character Recognition. Journal of Computer Science Engineering and Software Testing (e-ISSN: 2581-6969.

[5] Cvzone Github https://github.com/cvzone/cvzone

[6] Google Teachable Machine https://teachablemachine.withgoogle.com/