EXPLORATORY PROJECT REPORT

PROJECT AIM

To design a Machine Learning Model to detect characters from American Sign Language.

Submitted by:

Ajinkya Rajendra Mohale (20095067)

Sanidhya Verma (20095101)

Udit Agarwal (20095118)

Vedant Gupta (20095135)

Under the Guidance of

Dr. Sanjeev Sharma

(Department of Electronics Engineering)

Department Of Electronics Engineering
IIT (BHU) Varanasi

CERTIFICATE

This is to certify that this project report "**To design a Machine Learning Model to detect characters from American Sign Language"** is submitted by **Sanidhya Verma, Udit Agarwal, Vedant Gupta and Ajinkya Rajendra Mohale**

who carried out the project work under the supervision of

Dr. Sanjeev Sharma.

We approve this project for submission of the Exploratory Project, IIT(BHU) Varanasi.

Signature of Supervisor
Dr. Sanjeev Sharma
Department of Electronics Engineering
IIT(BHU) Varanasi

INDEX

Acknowledgment

It gives us immense pleasure to express our deepest sense of gratitude and sincere thanks to our highly respected and esteemed guide, Dr. Sanjeev Sharma, for his valuable support, guidance, encouragement, and help for accomplishing this work. His useful suggestions for this whole project are sincerely acknowledged.

We would also like to express our sincere thanks to all others who helped us directly or indirectly during this project work.

Date: 3-5-2022

Students Names:

Ajinkya Rajendra Mohale (20095067)
Sanidhya Verma (20095101)
Udit Agarwal (20095118)
Vedant Gupta (20095135)

ABSTRACT

This report presents the design, and prototype implementation and testing of Machine Learning Model that detect characters from American Sign Language and outputs the corresponding English Characters on the screen.

This project went through Three Phases:

Phase One:

Studying about all the various sign language systems already in place and selecting a more popular and user-friendly language system to translate.
Then, finding a data set of that particular sign language to train the machine learning model.

Phase Two:

Figuring out all the methods, software, functionalities, and extra learning required to implement our proposal via books, the internet, help from colleagues, guide, etc.

Phase Three:

Training the machine learning Model on the given dataset via Convolutional neural Networks, and using build a program using OpenCV to detect sign language characters from live video feed and output the corresponding English character.

INTRODUCTION

Sign language is a language through which communication is possible without the means of acoustic sounds. Instead, sign language relies on sign patterns, i.e., body language, orientation and movements of the arm to facilitate understanding between people. There are perhaps around two hundred sign languages in use around the world today. It has been estimated that there are between 0.9 and 14 million hearing impaired in India and perhaps "one of every five people who are deaf in the world, lives in India", making it the country with the largest number of Deaf, and thus also the largest number of sign language users.

But on contrary, there has always been a communication barrier between deaf and mute people and the speaking community, as apart from the deaf and mute, only few people know sign language. Many times, a human translator is employed for the translation. But every time a translator may not be available and even available.

Also, this factor limits the interaction between the deaf and mute community and the rest of society. Our project, "**To design a Machine Learning Model to detect characters from Sign Language**" aims to eliminate this barrier. The task is accomplished using Image processing in OpenCV and Machine learning. Machine-learning and Image processing are very powerful tools often used for image classification and recognition. Image Processing deals with the image, its properties, and the operations performed on it for getting some information from the images. Machine learning is the study of algorithms and statistical data used to perform tasks using various data patterns and inferences. In this project, the collection of the images of the sign language are to be done using a camera. The images are then processed and the features are extracted using image processing. These images are them compared from the available datasets and by implementing deep learning, the signs are interpreted. The data is displayed on a display that helps the person in front of a deaf/mute person understand the sign language.

Thus, we have developed a simple and lightweight deep learning algorithm which can detect static American sign-language gestures.

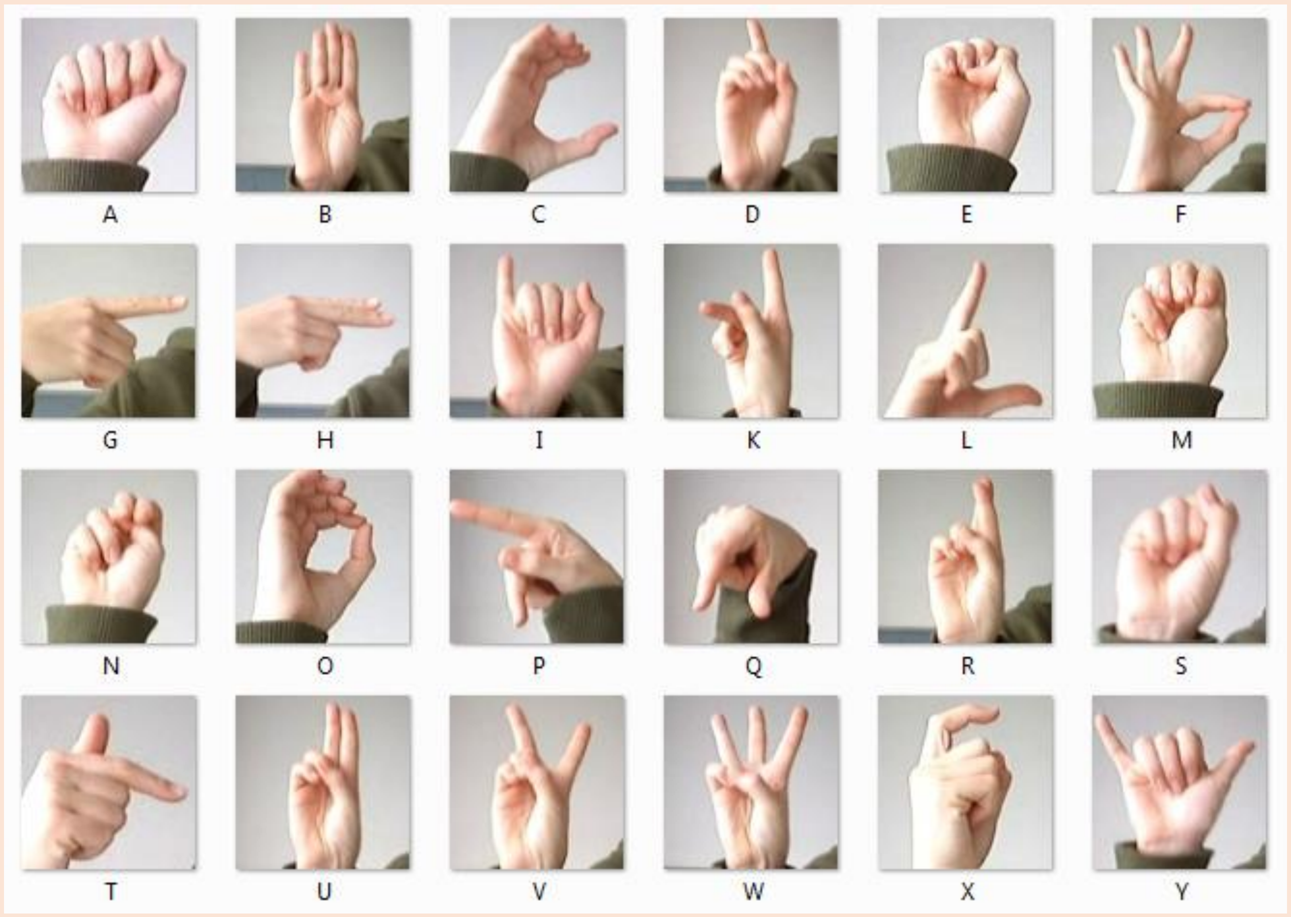
The Dataset Used

The original MNIST image dataset of handwritten digits is a popular benchmark for image-based machine learning methods but researchers have renewed efforts to update it and develop drop-in replacements that are more challenging for computer vision and original for real-world applications.

To stimulate the community to develop more drop-in replacements, the Sign Language MNIST was formed and it follows the same CSV format with labels and pixel values in single rows. The American Sign Language letter database of hand gestures represent a multi-class problem with 24 classes of letters (excluding J and Z which require motion, which will be thus out of scope of this project).

The dataset format is patterned to match closely with the classic MNIST. Each training and test case represents a label (0-25) as a one-to-one map for each alphabetic letter A-Z (and no cases for 9=J or 25=Z because of gesture motions). The training data (27,455 cases) and test data (7172 cases) are approximately half the size of the standard MNIST but otherwise similar with a header row of label, pixel, pixel2….pixel784 which represent a single 28x28 pixel image with grayscale values between 0-255. The original hand gesture image data represented multiple users repeating the gesture against different backgrounds.

The Sign Language MNIST data came from greatly extending the small number (1704) of the color images included as not cropped around the hand region of interest. To create new data, an image pipeline was used based on ImageMagick and included cropping to hands-only, gray-scaling, resizing, and then creating at least 50+ variations to enlarge the quantity. The modification and expansion strategy were filters ('Mitchell', 'Robidoux', 'Catrom', 'Spline', 'Hermite'), along with 5% random pixelation, +/- 15% brightness/contrast, and finally 3 degrees rotation. Because of the tiny size of the images, these modifications effectively alter the resolution and class separation in interesting, controllable ways.



This dataset was inspired by the Fashion-MNIST 2 and the machine learning pipeline for gestures by Sreehari 4.

A robust visual recognition algorithm could provide not only new benchmarks that challenge modern machine learning methods such as Convolutional Neural Nets but also could pragmatically help the deaf and hard-of-hearing better communicate using computer vision applications.

The National Institute on Deafness and other Communications Disorders (NIDCD) indicates that the 200-year-old American Sign Language is a complete, complex language (of which letter gestures are only part) but is the primary language for many deaf people. One could implement computer vision in an inexpensive board computer like Raspberry Pi with OpenCV, and some Text-to-Speech to enabling improved and automated translation applications.

Total number of images: 90380.

Hence this data set was a preferred our choice.

Theory

CONVOLUTIONAL NEURAL NETWORK (CNN)

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 tolearn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in theHuman 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. Acollection of such fields overlap to cover the entire visual area.

Convolutional layers

Convolutional layers are named after the convolution operation. In mathematics convolution is an operation on two functions that produces a third function that is the modified (convoluted) version of one of the original functions. The resulting function gives in integral of the pointwise multiplication of the two functions as a function of the amount that one of the original functions is translated.

A convolutional layer consists of groups of neurons that make up kernels. The kernels have a small size butthey always have the same depth as the input. The neurons from a kernel are connected to a small region of the input, called the receptive field, because it is highly inefficient to link all neurons to all previous outputs in the case of inputs of high dimensions such as images. For example, a 100 x 100 image has 10000 pixels and if the first layer has 100 neurons, it would result in 1000000 parameters. Instead of each neuron having weights for the full dimension of the input, a neuron holds weights for the dimension of the kernel input.

The kernels slide across the width and height of the input, extract high level features and produce a 2 dimensional activation map. The stride at which a kernel slides is given7 as a parameter. The output of a convolutional layer is made by stacking the resulted activation maps which in turned is used to define the input of the next layer. Applying a convolutional layer over an image of size 32 X 32 results in an activation map of size 28 X 28. If we apply more convolutional layers, the size will be further reduced, and, as a result the image size is drastically reduced which produces loss of information and the vanishing gradient problem. To correct this, we use padding.

Padding increases the size of a input data by filling constants around input data. In most of the cases, this constant is zero so the operation is named zero padding. ”Same” padding means that the output feature map has the same spatial dimensions as the input feature map. This tries to pad evenly left and right, but if the number of columns to be added is odd, it will add an extra column to the right. ”Valid” padding is equivalent to no padding. The strides causes a kernel to skip over pixels in an image and not include them in the output. The strides determines how a convolution operation works with a kernel when a larger image and more complex kernel are used.

As a kernel is sliding the input, it is using the strides parameter to determine how many positions to skip. ReLU layer, or Rectified Linear Units layer, applies the activation function max(0, x). It does not reduce the size of the network, but it increases its nonlinear properties.

Pooling layers

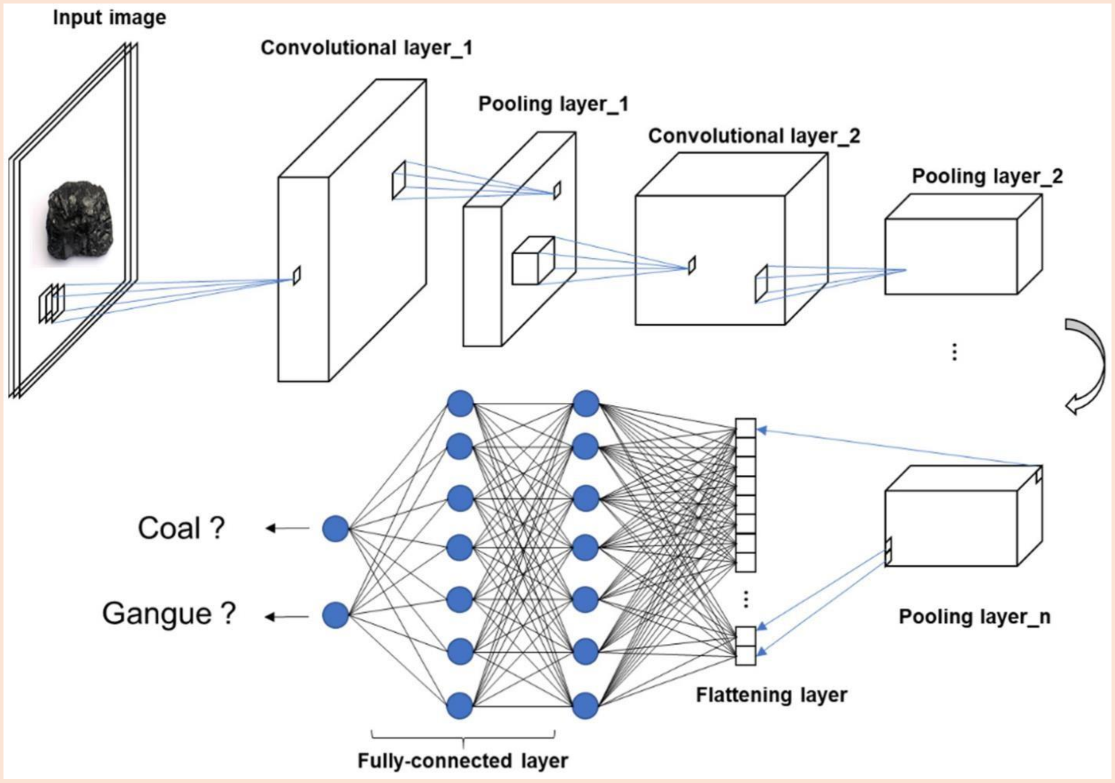
Pooling layers are used on one hand to reduce the spatial dimensions of the representation and to reduce the amount of computation done in the network. The other use of pooling layers is to control overfitting. The most used pooling layer has filters of size 2 x 2 with a stride 2. This effectively reduces the input to a quarter of its original size.

Fully connected layers

Fully connected layers are layers from a regular neural network. Each neuron from a fully connected layer is linked to each output of the previous layer. The operations behind a convolutional layer are the same as in a fully connected layer. Thus, it is possible to convert between the two.

Loss layers

Loss layers are used to penalize the network for deviating from the expected output. This is normally the last layer of the network. Various loss function exist: softmax is used for predicting a class from multiple disjunct classes, sigmoid cross-entropy is used for predicting multiple independent probabilities (from the [0, 1] interval).



Libraries Used:

TensorFlow:

TensorFlow is an open-source library for fast numerical computing.

Unlike other numerical libraries intended for use in Deep Learning, TensorFlow was designed for use both in research and development and in production systems.

Keras:

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation.

OpenCV:

OpenCV is the huge open-source library for computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it is integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

OpenCV Functionality:

1. Object/feature detection

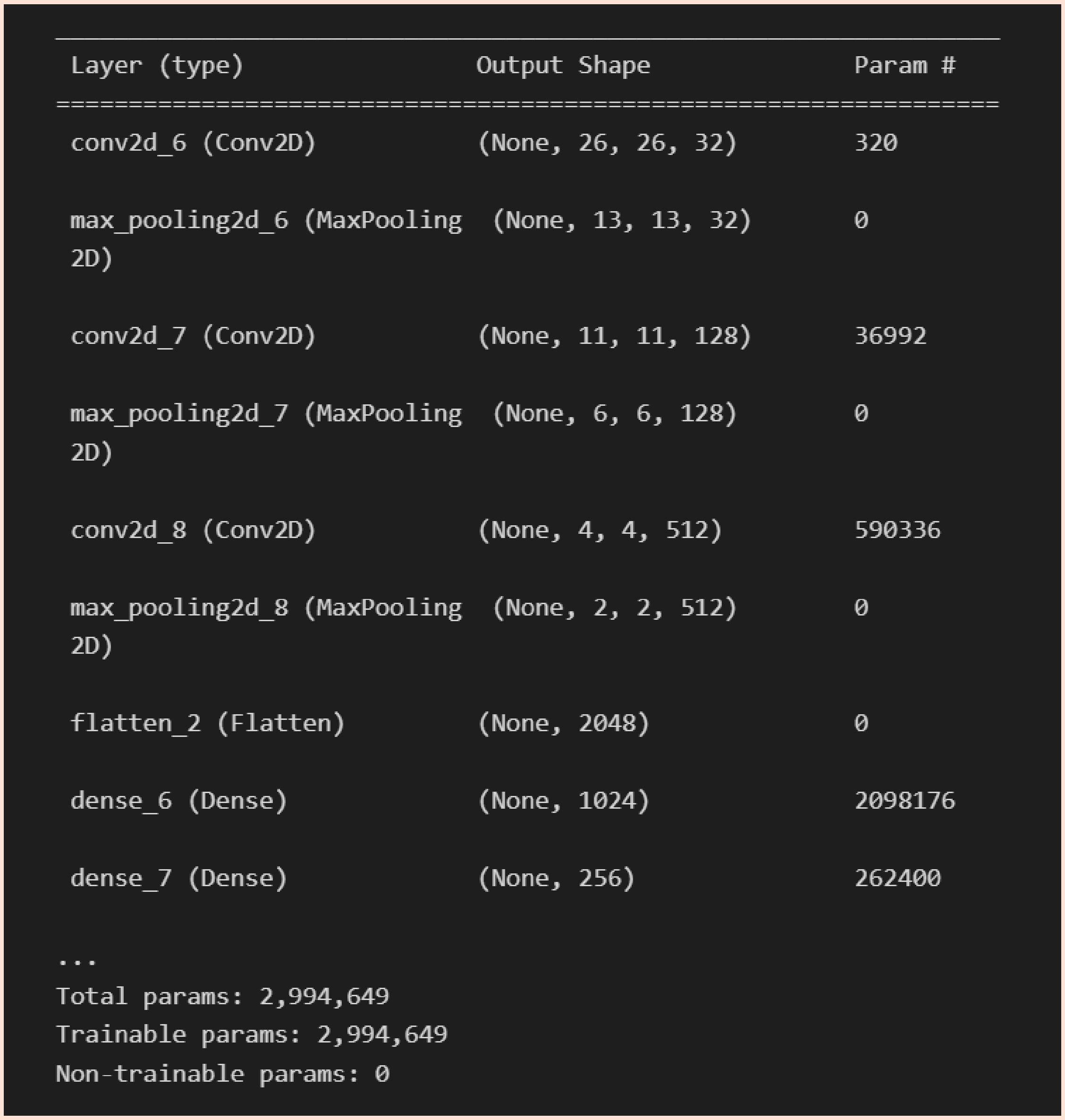
2. Image/video I/O, processing, display

3. Geometry-based monocular or stereo computer vision

4. Computational photography

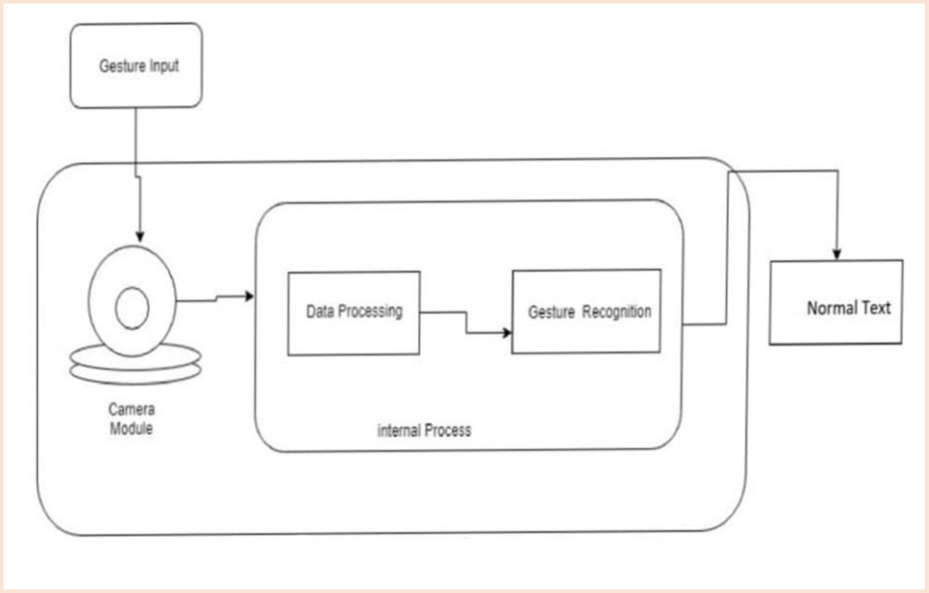
5. Machine learning & clustering

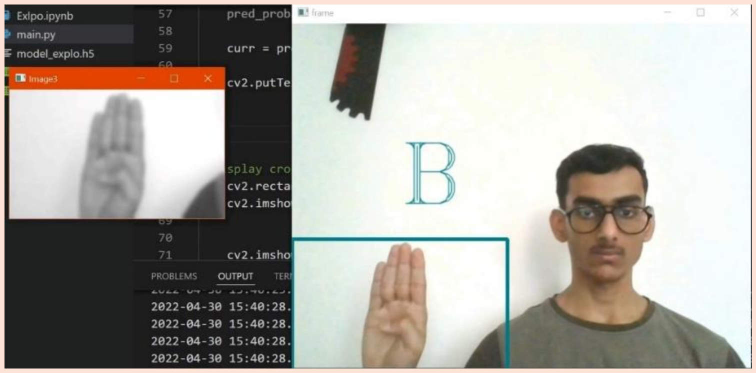
Model Proposed:



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 26, 26, 32)        320

 max_pooling2d_6 (MaxPooling  (None, 13, 13, 32)       0
 2D)

 conv2d_7 (Conv2D)           (None, 11, 11, 128)       36992

 max_pooling2d_7 (MaxPooling  (None, 6, 6, 128)        0
 2D)

 conv2d_8 (Conv2D)           (None, 4, 4, 512)         590336

 max_pooling2d_8 (MaxPooling  (None, 2, 2, 512)        0
 2D)

 flatten_2 (Flatten)         (None, 2048)              0

 dense_6 (Dense)             (None, 1024)              2098176

 dense_7 (Dense)             (None, 256)               262400

...
Total params: 2,994,649
Trainable params: 2,994,649
Non-trainable params: 0
```

Working of the model





```
pred_prob
curr = pr
cv2.putTe
splay cro
cv2.recta
cv2.imsho
cv2.imsho
```

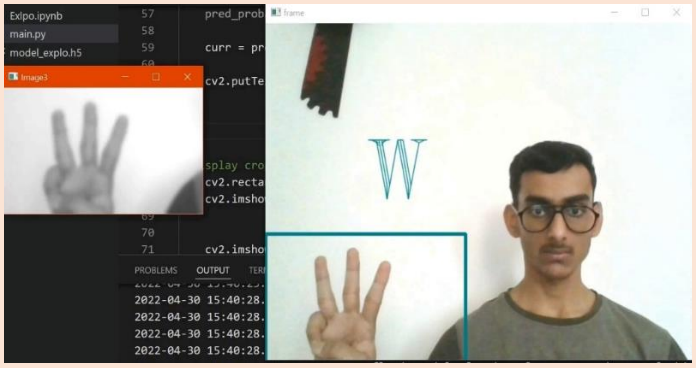
```
2022-04-30 15:40:28.
2022-04-30 15:40:28.
2022-04-30 15:40:28.
2022-04-30 15:40:28.
```

We used OpenCV to capture live image from the device.

We cropped the region from (0,300) pixels to (300,600) pixels for capturing the hand gesture for prediction.

The captured image was pre-processed using the following operations:

- Converting the image to grayscale
- Applying Gaussian Blur
- Resize the image to (28x28) pixels using cv2.INTER_AREA



```
pred_prob
curr = pr
cv2.putTe
splay cro
cv2.recta
cv2.imsho
cv2.imsho
```

```
2022-04-30 15:40:28.
2022-04-30 15:40:28.
2022-04-30 15:40:28.
2022-04-30 15:40:28.
```

The resized image was passed to the trained model for prediction.

Results and Discussions

- This this Machine learning program can bridge the gap between the deaf and mute people and rest of the society.
- Thus, communication gap between the deaf and mute and the people that could not understand the sign language is bridged.
- By making it a portable device the usage will be increased.
- The improved speed and accuracy help us use the device in real-time with less lag.
- It can be deployed in potential places like hospitals and police stations where emergency communication is of at most important.
- It will help us to make a better society.

Future Scope of the Project

- The model thus generated can be used to develop mobile applications, websites, translation devices, etc that can be put to use to improve interaction of the deaf and mute people with the society.
- As for the future, the system recently can recognize the ASL character only and can be enhanced to recognize the digits and the dynamic gesture too.
- Also, the 2 characters that required motion could be added in future.
- The system particularly requires good lighting condition to recognize the gesture and can be enhanced by using a depth-sensing camera which helps and allows to recognize the 3D gestures and the motion gestures.

Summary and Conclusion

- An American Sign Language recognition system using Convolution Neural Network was thus developed using implementing a Machine Learning CNN model.
- It was able to recognize the simple static ASL gestures with accuracy above 98% and classify real-world environment image with accuracy above 85% in good lighting conditions.

GitHub link for the project repository:

https://github.com/vedcer20/EXPLO

References

- https://towardsdatascience.com/introduction-to-convolutional-neural-network-cnn-de73f69c5b83#:~:text=Dense%20Layer%20is%20simple%20layer,on%20output%20from%20convolutional%20layers
- https://keras.io/about/
- Coursera- Machine Learning by Andrew Ng
- https://www.youtube.com/watch?v=Z78zbnLlPUA&list=PLQVvvaa0QuDdttJXlLtAJxJetJcqmqlQq
- https://www.kaggle.com/datasets/datamunge/sign-language-mnist?select=amer_sign2.png
- https://www.nidcd.nih.gov/health/american-sign-language