<u>Final Project Report</u>

Team 33

Manasi Jadhav - mjadhav

Sneha Singh - singh43

Sneha Yadav - snehayad

Signify - Real-time ASL Detection Application

Background -

American Sign Language (ASL) is a critical communication tool for many people, particularly within the deaf and hard-of-hearing community in the United States. According to the commission on the Deaf and Hard of Hearing - *Approximately more than a half-million people throughout the US(1) use ASL to communicate as their native Language. First appeared in the 1800s, ASL is the third most commonly used language in the United States, after English and Spanish.*

Need for ASL -

1. **Communication Access** - ASL is the primary language for many people who are deaf or hard of hearing in the U.S. This visual language comprises signs made by moving the hands combined with facial expressions and postures of the body. It is not a derivative of English; it has its own unique grammar and syntax. Everyone needs to have a basic knowledge of ASL in order to communicate with deaf and hard-of-hearing communities.
2. **Cultural Identity** - ASL is not just a means of communication but also an essential part of the cultural identity and community interactions among deaf individuals. It plays a crucial role in the social integration and personal development of people who use it.
3. **Education and Learning** - Access to education in ASL can be crucial for the cognitive and social development of deaf children. Studies suggest that early exposure to ASL can significantly benefit educational outcomes for children who are deaf or hard of hearing.
4. **Legal and Health Access** - There is a legal requirement in many contexts to provide ASL interpretation to ensure that deaf and hard-of-hearing individuals have equal access to legal and health services. This helps prevent misunderstandings and ensures that individuals can make informed decisions about important aspects of their lives.

ASL is the need for many people now-a-days. It is also important that each individual knows this language very well in the same manner as the English Language is needed to survive in the community. Computer Scientists and Engineers are now developing multiple tools to help in this matter. Continued investment in research and development in these areas not only supports the ASL-using community but also advances the field of human-computer interaction and machine learning, contributing to more universal design principles in technology.

Research Needs in Computer Science

1. **ASL Recognition and Translation Technologies -** Developing robust machine learning models that can accurately recognize ASL through image/video inputs can enhance communication between ASL users and those who do not know ASL. This includes real-time translation services and improved accessibility in public services and media.
2. **Enhanced Learning Tools -** There is a growing need for educational technologies that can aid in teaching and learning ASL. We can build interactive apps that make learning ASL more engaging and effective.
3. **Accessibility in Digital Media -** Ensuring that digital content is accessible to ASL users involves research into automatic sign language generation for live broadcasts, online videos, and virtual meetings. This would greatly increase the inclusivity of digital media platforms.
4. **Human-Computer Interaction (HCI) -** Research in HCI can explore more intuitive ways for ASL users to interact with technology using gesture-based inputs. This can include improvements in sensor technology, wearable technology designed specifically for ASL users.
5. **Ethical AI Use -** As with any application of AI, ethical considerations are important. Ensuring that the development of ASL-related technologies respects privacy, consent, and accuracy can help build trust and utility in these systems.

In this project, we have focused on the above mentioned first two points.

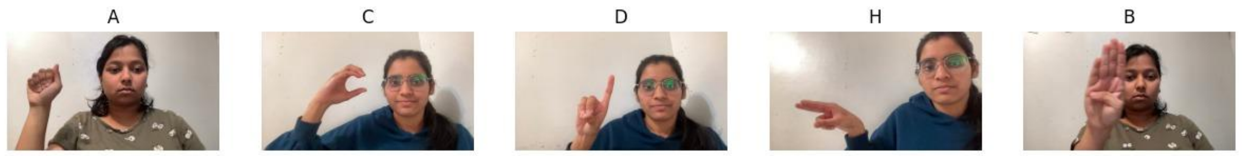
Data Used in this Project

The data we are using in this project are images captured using the opencv library of python.

Data Collection -

We have created this dataset on our own. We tried to run our model on two different datasets but the accuracy obtained was low. So, we shifted to create our own dataset with clear labeling. We have captured data for 24 different classes of alphabets (j and z are excluded as they require actions). There are 400 images for each class in the MNIST dataset format. We are using 9600 images in total. We are using the opencv library to capture these images through our webcam and os library for directory management and navigation. Images are captured as frames and stored within their respective directories based on the class.

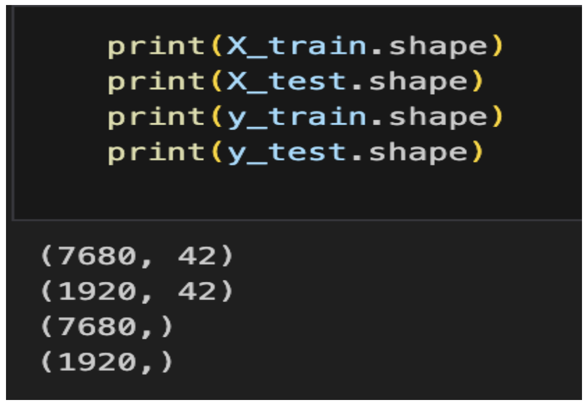
Dataset Creation and Preprocessing -



These images are then converted from BGR to RGB format because MediaPipe requires RGB format. We are using the mediapipe library which is developed by Google to capture the hand movements. We are using this library for real-time hand tracking and gesture recognition. Each directory name is a label. We are using mediapipe and opencv for image preprocessing. Then we are using os and pickle libraries for directory handling and data serialization. Then we are processing the image to detect hands and extract landmarks using hand.process(). If exactly one hand is detected, the x and y coordinates of each landmark (21 in total) are extracted and stored in 'data_aux'. If the number of landmarks in data_aux is not equal to 42 we are padding it with zeros to maintain consistent feature vector lengths. So, if no hand or more than one hand is detected, data_aux is filled with zeros.
After processing all images, we are serializing the collected data and labels into a file using the pickle module. This file we are using for further processing and training the model.

Model Definition and Training

Now to pass the data generated from the previous step, we are using pickle to deserialize the data stored. Then we are extracting data and labels from the loaded dictionary (data_dict) and converting them to NumPy arrays for easier manipulation and compatibility. Then we are using the train_test_split() function from scikit-learn to divide the data into training and testing sets. This is crucial for evaluating the model on unseen data, thus ensuring that the model generalizes well to new inputs. We are dividing this data into an 80:20 ratio where 20% is test data and 80% is training data.



```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

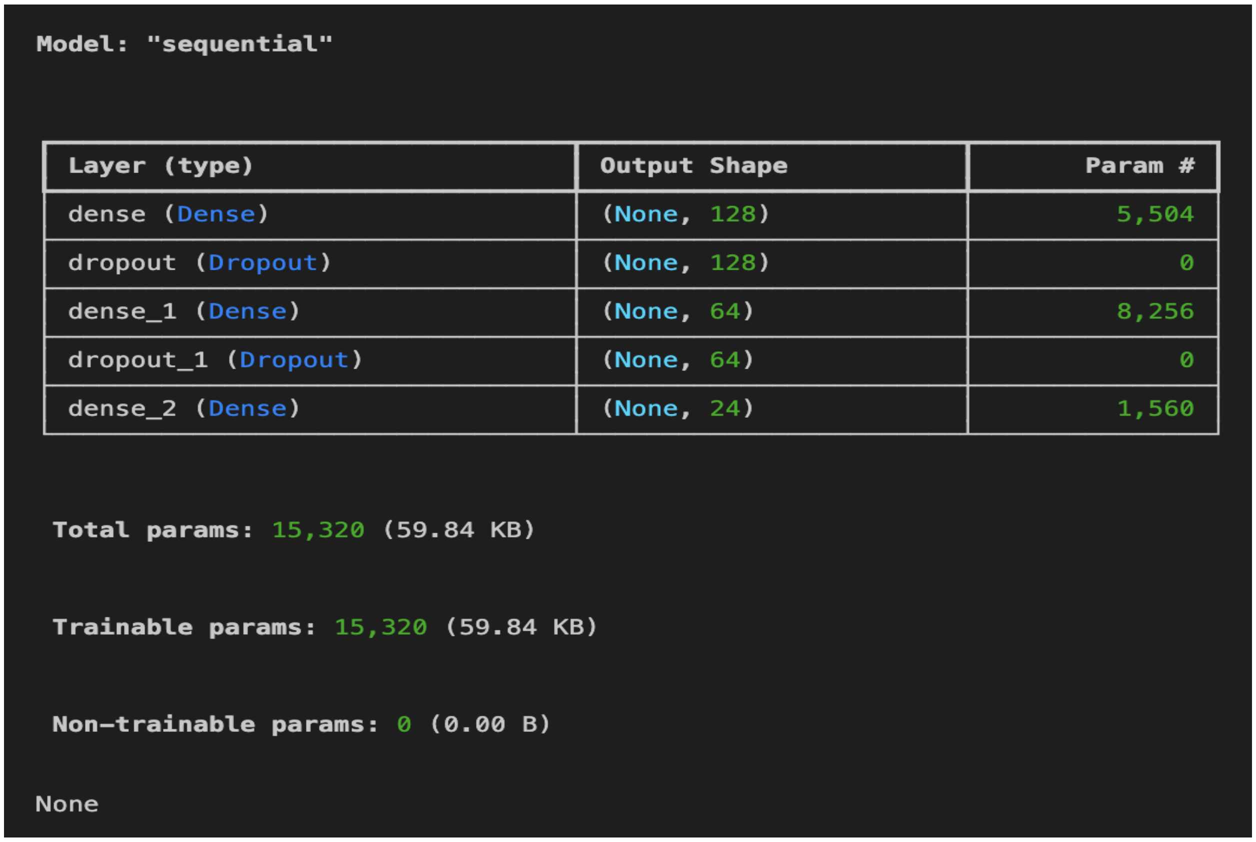
```
(7680, 42)
(1920, 42)
(7680,)
(1920,)
```

Model Definition -

1. **Dense Layers -** Our network starts with a dense layer of 128 neurons, followed by another dense layer of 64 neurons, and ends with a dense layer of 24 neurons. Each

dense layer includes an activation function ReLU, which helps the model learn non-linear patterns. At the end we are using the softmax function for the classification.

2. **Dropout Layers** - Between the dense layers, there are dropout layers with a dropout rate of 0.5. Dropout is a regularization technique used to prevent overfitting by randomly setting a fraction of the input units to 0 at each update during training time.
3. **Parameters** - The total number of trainable parameters in the model is 15,320. These parameters are learned during the training process to minimize the loss function.
4. **Output** - The final output of the model comes from the last dense layer with 24 neurons as we have 24 classes.
5. **Use Case** - We are using this architecture as a baseline to solve this complex problem. That is why we have kept the architecture of the given model with straightforward design and moderate number of parameters.



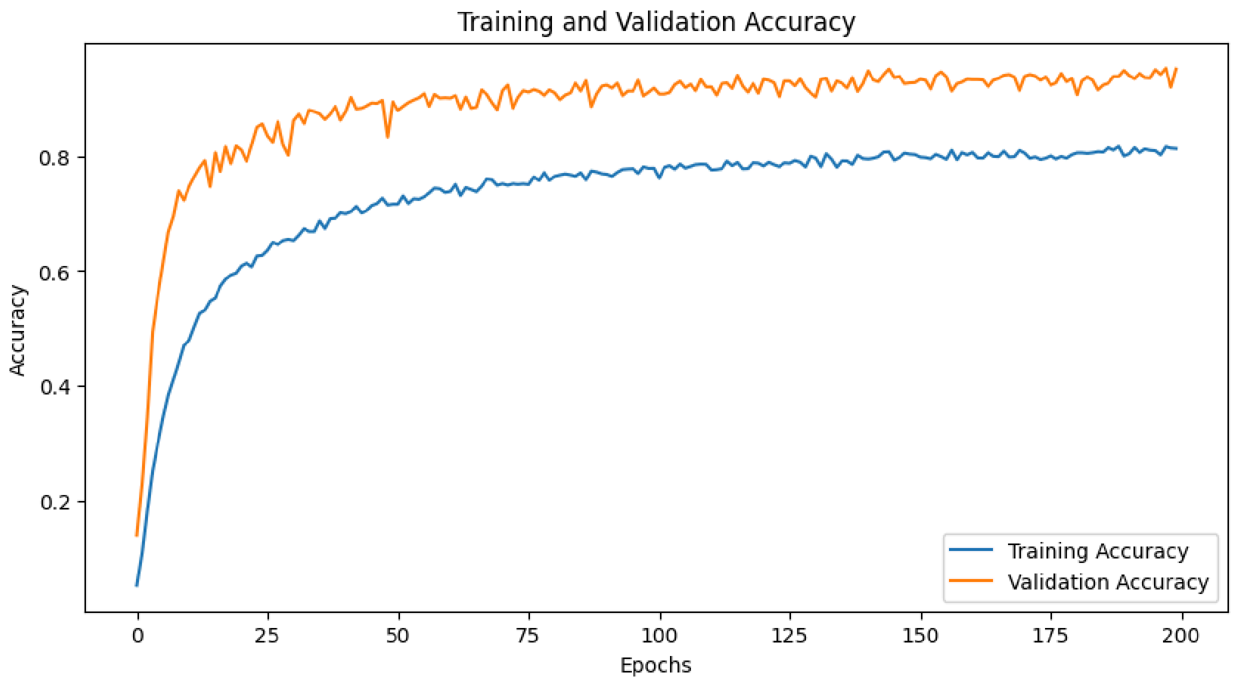
```
Model: "sequential"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━┓
┃ Layer (type)                     ┃ Output Shape            ┃        Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━┩
│ dense (Dense)                    │ (None, 128)             │          5,504 │
├──────────────────────────────────┼─────────────────────────┼────────────────┤
│ dropout (Dropout)                │ (None, 128)             │              0 │
├──────────────────────────────────┼─────────────────────────┼────────────────┤
│ dense_1 (Dense)                  │ (None, 64)              │          8,256 │
├──────────────────────────────────┼─────────────────────────┼────────────────┤
│ dropout_1 (Dropout)              │ (None, 64)              │              0 │
├──────────────────────────────────┼─────────────────────────┼────────────────┤
│ dense_2 (Dense)                  │ (None, 24)              │          1,560 │
└──────────────────────────────────┴─────────────────────────┴────────────────┘

 Total params: 15,320 (59.84 KB)

 Trainable params: 15,320 (59.84 KB)

 Non-trainable params: 0 (0.00 B)

None
```

We are using an ‘Adam’ optimizer that will adjust the weights of the network during training. Then we are using the ‘sparse_categorical_crossentropy’ loss function. This loss function is used for multi-class classification problems where the labels are provided as integers. Metric is used to monitor the training and testing steps. "Accuracy" assesses the percentage of correctly predicted instances among the total instances.
We are using the fit() method, which includes loss and accuracy metrics for both training and validation datasets over each epoch with images in batch size of 32.

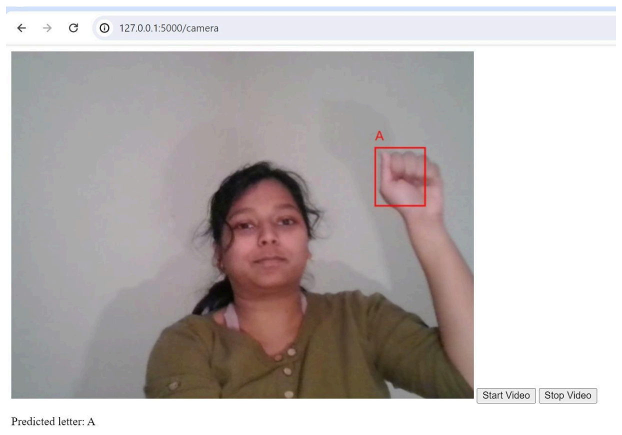
Results

Training vs Accuracy -

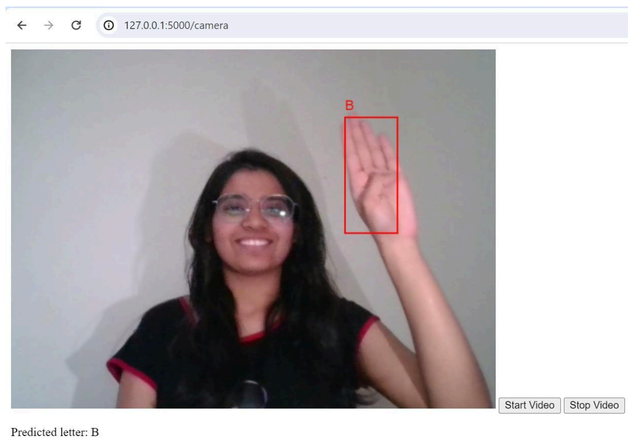
Following is the graph which shows Training and Validation accuracy in increasing manner over 200 epochs.



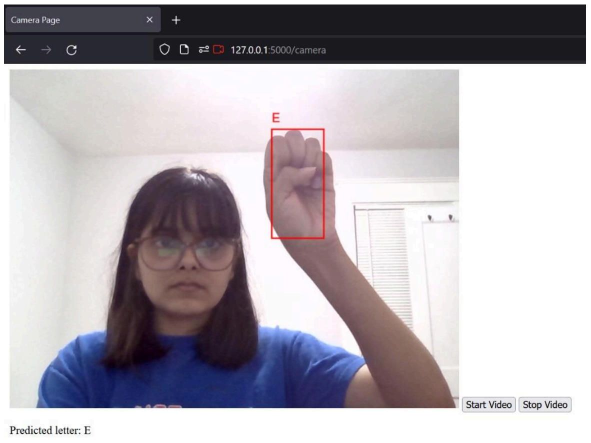
The application detects the word and shows its label (alphabet letter) in real-time.



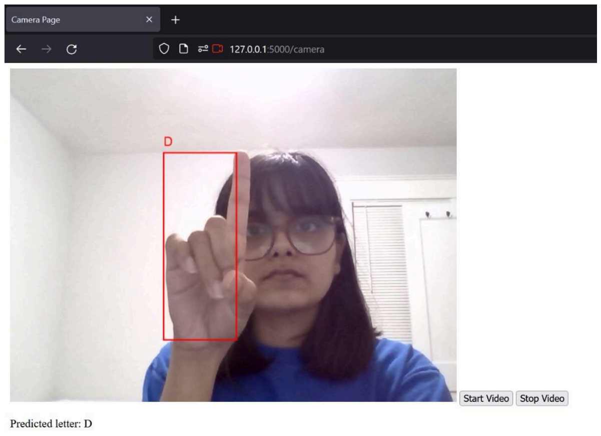
Predicted letter: A



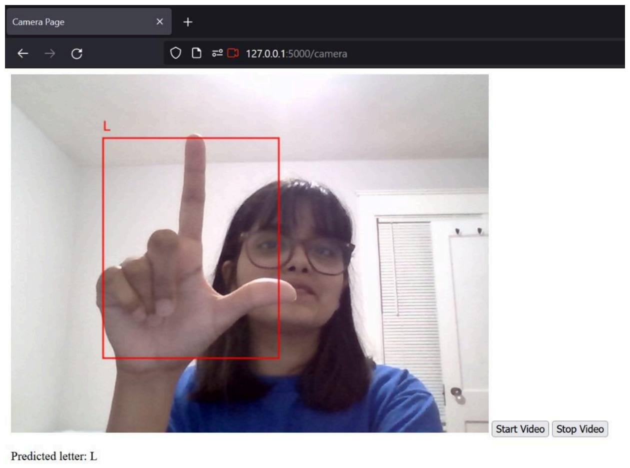
Predicted letter: B



Predicted letter: E



Predicted letter: D



Predicted letter: L

Bonus Tasks

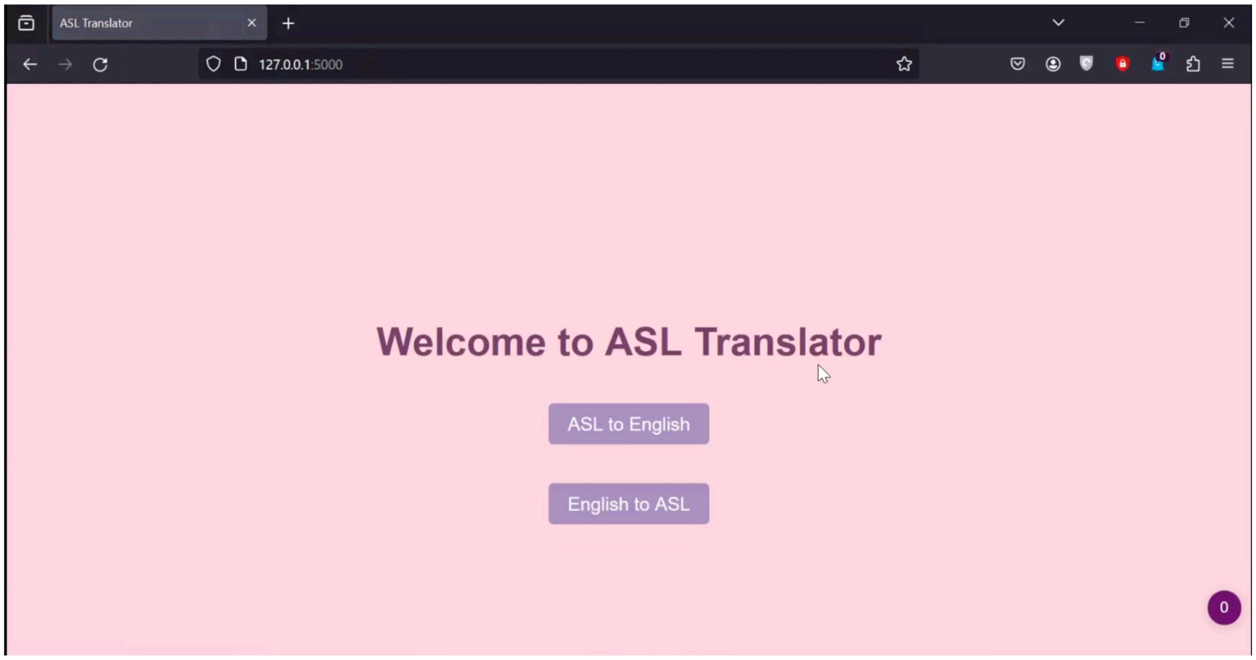
Real-world deep learning application -

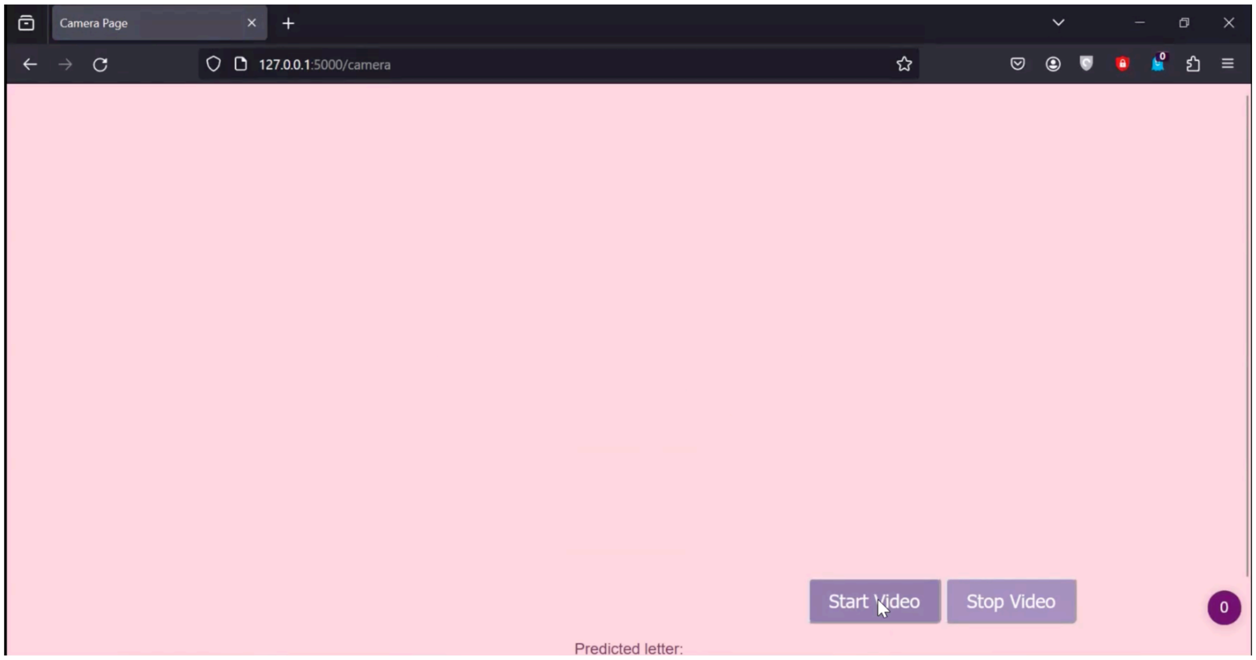
Developing an American Sign Language (ASL) detection and translation application using deep learning offers numerous benefits and practical applications in the real world. Following are the real-world applications of this project -

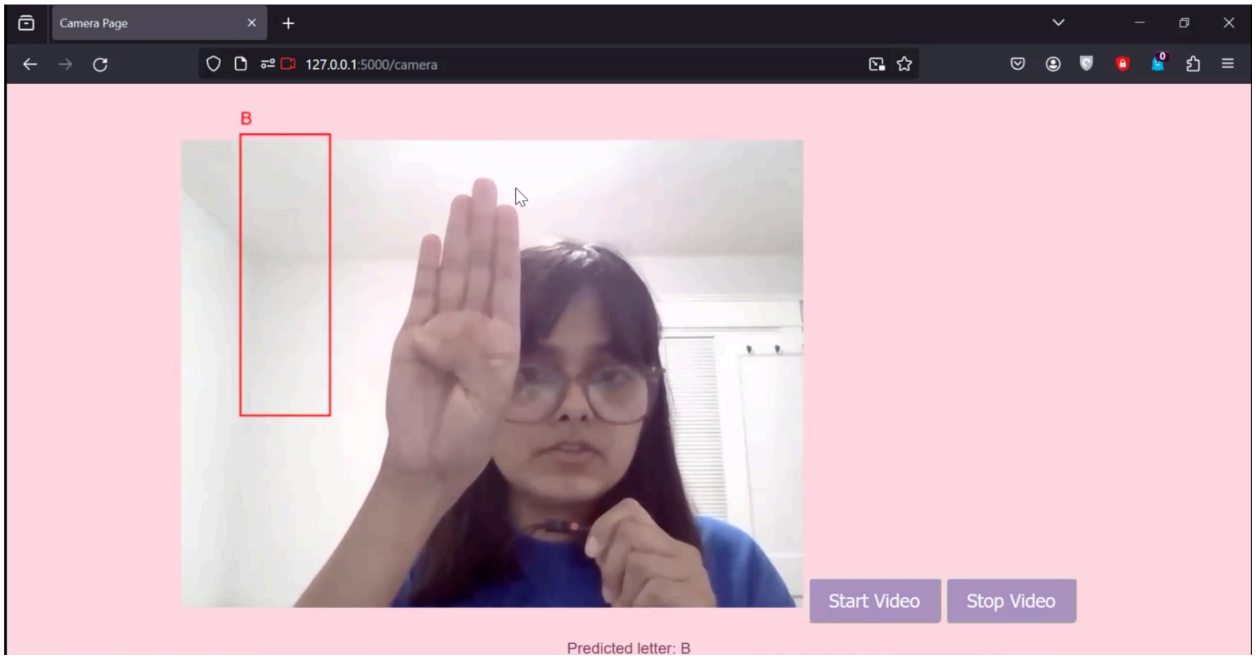
1. **Enhanced Communication -** The primary benefit of an ASL translation application is improved communication access for the deaf and hard of hearing community. By providing real-time translation of ASL into written language, the application bridges the communication gap between ASL users and those who do not know ASL, facilitating smoother, more inclusive interactions in daily life.
2. **Education -** In educational settings, such an application can significantly enhance the learning experience for deaf students. It can provide real-time translation of lectures and discussions, allowing deaf students to participate more. Additionally, it can serve as a learning aid for individuals interested in learning ASL, promoting wider awareness and understanding of sign language.

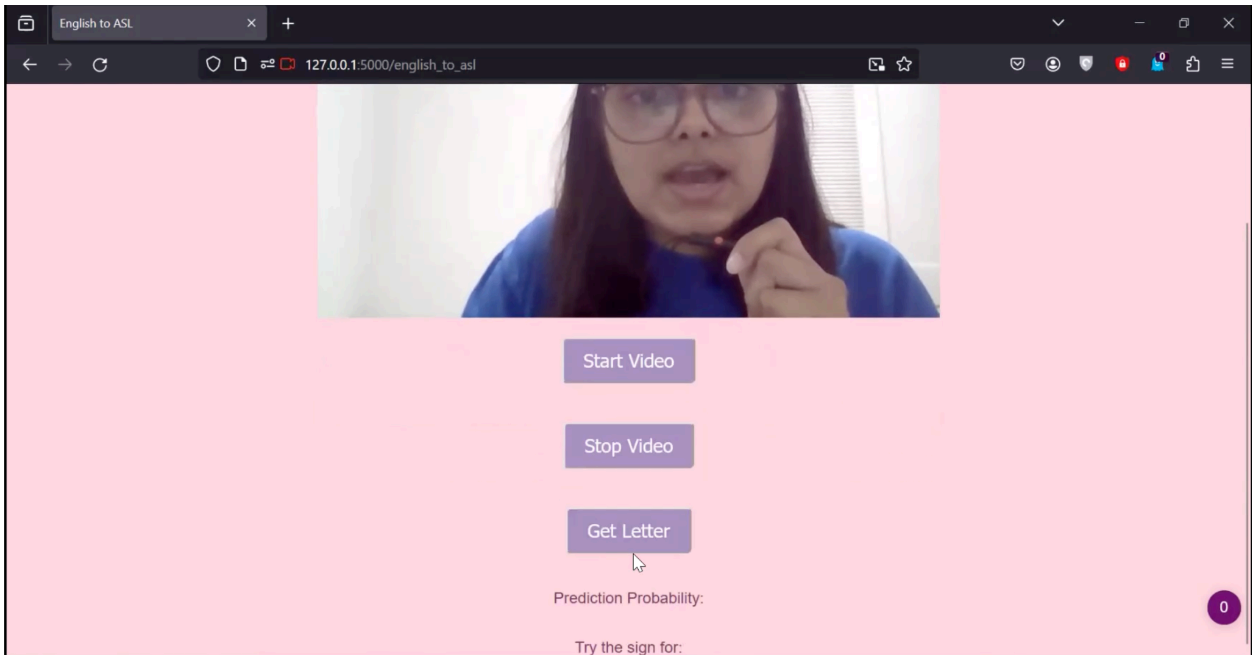
3. **Social Inclusion** - Beyond functional benefits, the application promotes social inclusion, helping to normalize ASL and the participation of deaf individuals in all aspects of social life.

Locally Deployed Application -









<u>Team Contribution</u>

Team Member	Contribution
Manasi Jadhav - mjadhav	33.33%
Sneha Singh - singh43	33.33%

Sneha Yadav - snehayad	33.33%

References

1. How to work with Computer Vision Package code demo shared on CSE 676 B Piazza - https://piazza.com/class/lrsnr0odgns51m/post/554
2. https://mediapipe.readthedocs.io/en/latest/solutions/hands.html
3. https://www.tensorflow.org/guide/keras/sequential_model
4. https://en.wikipedia.org/wiki/American_Sign_Language
5. https://opencv.org/
6. https://www.youtube.com/watch?v=MJCSjXepaAM&ab_channel=Computervisionengineer