MINI PROJECT REPORT

SIGN LANGUAGE RECOGNITION AND MULTILINGUAL TRANSLATION TO INDIAN LANGUAGES

SUBMITTED TO:
DR SONALI AGARWAL

SUBMITTED BY:

PREETI	(IIB2021042)
GIRISHA VASHISHT	(IIT2021183)
KANISHKA SINGH	(IIT2021219)
AMIT KUMAR	(IIT2021054)
CHIRAG	(IIT2021035)

TABLE OF CONTENTS

ABSTRACT:

We have designed an application designed for sign language detection and translation, aiming to contribute to enhanced accessibility and inclusivity. To improve generalization, the core methodology employs a Convolutional Neural Network (CNN) trained on a dataset enhanced with Gaussian blur.

The initial phase of our application involves robust sign detection, where the CNN effectively identifies and classifies hand gestures, capturing the subtle gestures and contours inherent in sign language expression. Subsequently, the recognized hand gestures are converted into English text using the trained CNN, thereby helping in the transition from visual language to written language.

In a notable extension, we address the requirement for linguistic diversity by translating the English text into 21 Indian languages. This multilingual functionality is achieved through the integration of the Google Translate API into our application, ensuring precise and culturally relevant translations. The application is a practical tool for overcoming communication barriers, catering to India's various linguistic communities. This methodological approach not only advances the field of sign language technology, but also highlights the power of combining computer vision, machine learning, and language translation to create inclusive solutions for diverse populations.

Keywords: CNN, PyspellChecker Library, Keras, Tensor Flow, Google Translate API

OBJECTIVES:

The primary goal is to improve the efficiency of sign language technology by addressing the challenges posed by regional linguistic variations. We are dedicated to creating an intelligent structure that effortlessly incorporates these variations, resulting in a highly flexible and inclusive solution. We have used CNN convolutional neural networks to attain high efficiency. Our designed structure involves the use of 3 convolution layers with 3 max-pooling layers, 2 dense layers, and 1 dropout layer trained on a Gaussian blur dataset to achieve maximum accuracy.

Another key objective is the integration of the Google Translate API into our system to expand the utility of sign language translation across a broader linguistic spectrum. This integration serves as a concrete step towards addressing the linguistic complexities we've identified. We aim to make a meaningful contribution to the well-being of people with hearing impairment by promoting inclusivity and self-reliance, going beyond technological advancements.

INTRODUCTION:

American Sign Language (ASL) is a crucial means of communication for individuals with hearing impairments, providing a visual medium for expressing thoughts and messages. Despite strides in converting sign language to text, our project acknowledges the imperative for further innovation. Given the primary communication challenge faced by Deaf and Dumb (D&M) individuals, our objective is to enhance the accessibility and inclusivity of sign language translation.

Traditional methods of sign language translation have typically focused on converting visual gestures into textual representations. However, our distinctive approach involves integrating the Google Translate API into our existing application. This integration enables the direct translation of sign language into various languages, broadening the audience's reach. Our goal is not only to recognize Fingerspelling-based hand gestures but also to establish an inclusive platform where the interpreted content is available in multiple languages.

We recognize the regional diversity of sign language and underscore the importance of linguistic inclusivity. By incorporating the Google Translate API into our application, we aim to facilitate seamless communication between D&M individuals and those who communicate verbally, transcending linguistic barriers.

Our project surpasses conventional sign language recognition, embracing the potential for broader societal impact by ensuring that translated content is globally accessible. Through this integration, we contribute to the ongoing efforts to make technology a more effective tool for bridging communication gaps and fostering understanding among diverse linguistic communities.

EXISTING WORKS IN DOMAIN

Literature Review on Sign Gesture Recognition: A Formal Analysis**

This literature review provides a formal analysis of noteworthy contributions to the field.

1. Siming He's Comprehensive Framework [1]
Siming He created a system that used a dataset of 40 common words and 10,000 sign language images. The system's accuracy was improved through the integration of Faster R-CNN with an embedded RPN module for precise hand region localization. The combination of a 3D CNN for feature extraction along with a sign-language recognition framework including LSTM yielded a 99% accuracy rate for the dataset with commonly used vocabulary.

2. Cost-Effective Image Processing:[2]
In this approach, pictures were taken against a green background, which helped in streamlining the processing in the RGB color space. When it was applied to Sinhala sign language, the suggested method, incorporating a centroid mapping technique, attained a 92% recognition rate for sign gestures, regardless of the hand size and position.

3. Pigou's CNN Model and J Huang's 3D CNN Model:[3]
Pigou employed a 2D CNN model for 20 Italian sign gestures, achieving 91.70% accuracy. J Huang, on the other hand, created a dataset using Kinect, applying a 3D CNN with a notable average accuracy of 94.2%. Pigou's model demonstrated the efficacy of 2D CNNs, while J Huang showcased the advantages of 3D CNNs, considering multiple channels including color and depth.

4.Transfer Learning Approach by J. Carriera:[4]
In his framework, J. Carriera made use of a transfer learning approach by capitalizing on pre-trained datasets from ImageNet and the Kinetic Dataset. By combining RGB models, and flow models, and incorporating both pre-trained Kinetic and ImageNet data, he achieved an impressive accuracy rate of 98.0% on UCF-101 and 80.9% on HMDB-51 datasets.

In summary, these research works have collectively contributed to the dynamic field of sign gesture recognition, highlighting progress in terms of accuracy, efficiency, and versatility across diverse languages and datasets.

GAPS IDENTIFIED IN EXISTING WORK

Our project is motivated by a thorough examination of the existing landscape in sign language translation, revealing a significant gap that needs attention. While considerable progress has been made in translating sign language into American languages, there exists a notable void when it comes to translating sign language into multiple Indian languages. The predominant focus in this field has leaned towards Western linguistic contexts, overlooking the linguistic diversity present in India.

India is home to a rich tapestry of languages spoken across its diverse regions, presenting a unique challenge for individuals with hearing impairments who use sign language for communication. Unfortunately, there is a lack of technologies catering to the Deaf and Dumb (D&M) community in their native Indian languages.

Recognizing this gap, our project aims to address this specific challenge by integrating the Google Translate API into our application. This intentional integration not only addresses the current limitation but also empowers D&M individuals in India to communicate seamlessly in their preferred sign language, regardless of regional linguistic variations. By concentrating on sign language translation into multiple Indian languages, our goal is to contribute to a more inclusive technological landscape that acknowledges and embraces the linguistic diversity inherent in our global community.

METHODOLOGIES:

Dataset:

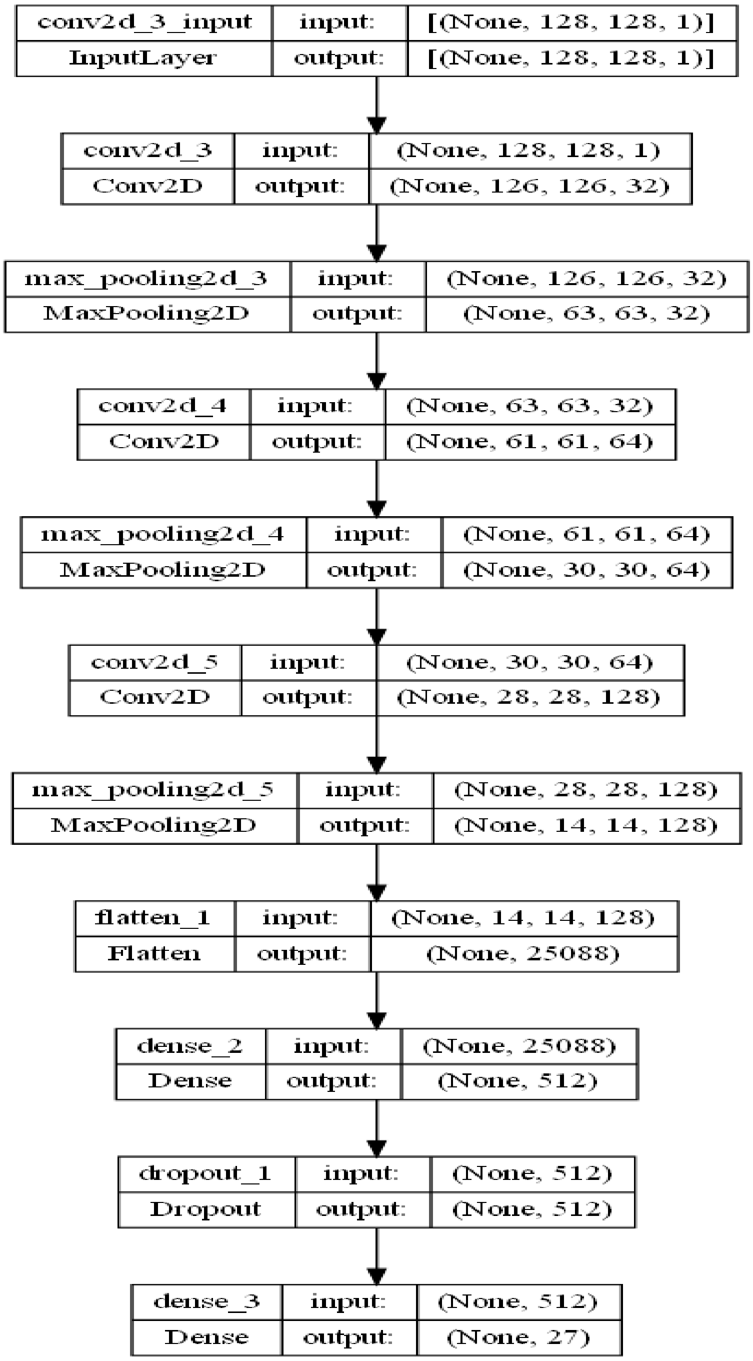
Our dataset consisted of 17113 images belonging from 27 different classes which consisted of alphabets(A-Z) and blank images used for representing the space. We used the Gaussian blur image dataset which is formed by converting RGB images to grey scale and then passing them through the Gaussian filter. We used Gaussian blur images for following reasons:

1. In the context of sign language detection, we found Gaussian blur a powerful tool. It helped to reduce image noise, creating a cleaner image that was crucial for accurate recognition of signs.

2. It is particularly good at preserving the edges of images, which is important as the edges of hands and fingers are vital in sign language recognition.

CNN Model (Proposed architecture) [5][11][12]

Below is our architecture of Convolutional Neural Networks that we used to train our model



conv2d_3_input	input:	[(None, 128, 128, 1)]
InputLayer	output:	[(None, 128, 128, 1)]
conv2d_3	input:	(None, 128, 128, 1)
Conv2D	output:	(None, 126, 126, 32)
max_pooling2d_3	input:	(None, 126, 126, 32)
MaxPooling2D	output:	(None, 63, 63, 32)
conv2d_4	input:	(None, 63, 63, 32)
Conv2D	output:	(None, 61, 61, 64)
max_pooling2d_4	input:	(None, 61, 61, 64)
MaxPooling2D	output:	(None, 30, 30, 64)
conv2d_5	input:	(None, 30, 30, 64)
Conv2D	output:	(None, 28, 28, 128)
max_pooling2d_5	input:	(None, 28, 28, 128)
MaxPooling2D	output:	(None, 14, 14, 128)
flatten_1	input:	(None, 14, 14, 128)
Flatten	output:	(None, 25088)
dense_2	input:	(None, 25088)
Dense	output:	(None, 512)
dropout_1	input:	(None, 512)
Dropout	output:	(None, 512)
dense_3	input:	(None, 512)
Dense	output:	(None, 27)

Now let’s Understand how this architecture works
1st Convolution Layer: The input picture in our has a resolution of 128x128 pixels. It will be first processed in the first convolutional layer using 32 filter weights (3x3 pixels each) resulting in a 126x126 pixel image, one for each filter weight.Then it is passed through the ReLU function(max(0,x)) to obtain the output from this layer.

1st Pooling Layer: Then the images are downsampled using max pooling of 2x2 containing the Stride of 2, resulting in a 63x63 pixel image.

2nd Convolution Layer: The output from the 1st pooling layer is served as an input to the second convolutional layer. It will be processed into the second convolutional layer using 64 filter weights (3x3 pixels each), resulting in a 61x61 pixel image.Then it is passed through the ReLU function(max(0,x)) to obtain the output from this layer.

2nd Pooling Layer: Then the images are downsampled using max pooling of 2x2 containing the Stride of 2, resulting in a 30x30 pixel image.

3rd Convolution Layer: The output from the second pooling layer is served as an input to the third convolutional layer. It is then processed in the third convolutional layer using 128 filter weights (3x3 pixels each), resulting in a 28x28 pixel image.Then it is passed through the [1]ReLU function(max(0,x)) to obtain the output from this layer.

3rd Pooling Layer: The images are downsampled using max pooling of 2x2, resulting in a 14x14 pixel image.

Flatten Layer: The output from the third pooling layer is then flattened into a 1D array of size 14x14x128 = 25088.

1st Densely Connected Layer: The flattened array is fed to a fully connected layer with 512 neurons.

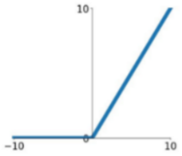
Dropout: A dropout layer with a dropout rate of 0.5 is applied to prevent overfitting.

2nd Densely Connected Layer: The output from the dropout layer is then fed to a fully connected layer with a number of neurons equal to the number of labels/classes which is 27.

Final layer: The output of the second densely connected layer serves as an input for the final layer, which uses the softmax activation function to produce the predicted probabilities for each label/class.

$$\text{softmax}(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ for } j = 1,\ldots,K$$

ReLU
$\max(0, x)$



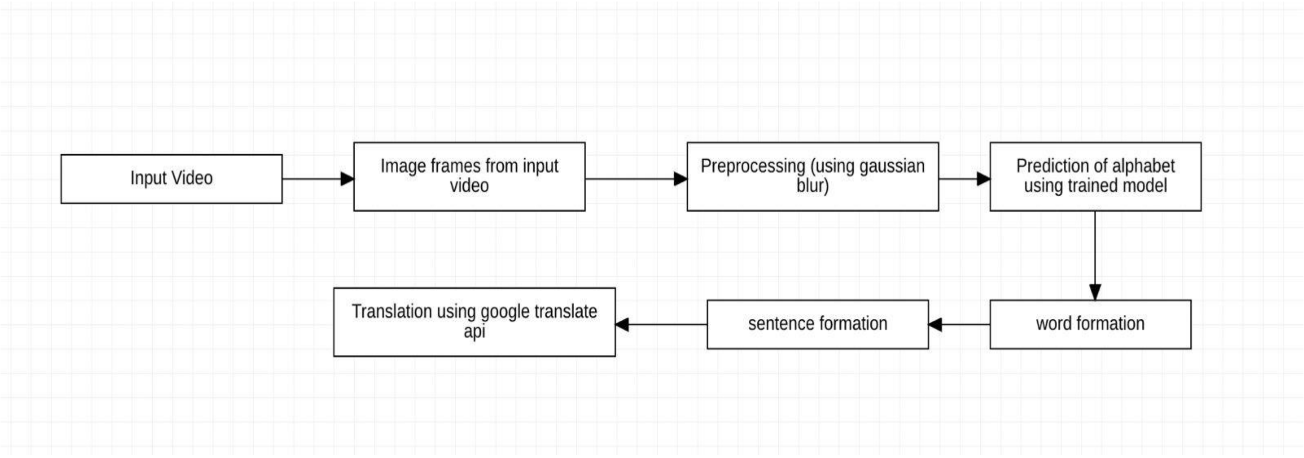
Then to train this model we have used the Adam optimizer with a learning rate of 0.001 which is used for updating the model in response to the output of the loss function during training. It combines the advantages of two stochastic gradient descent algorithms: adaptive gradient algorithm (AdaGrad) and root mean square propagation (RMSProp).

Hyperparameters: Hyperparameters used in our CNN training model are:
Learning rate.
Number of filters and size of filter used in each layer.
Activation Function used in Each Layer i.e., ReLU function and SoftMax functions.
Size of max pooling Layers which also sets the stride as the size of max pooling layer by default equal to the size of kernel used in our model.
Dropout Layer.

Pseudo Code (Architecture of CNN Used)

```
model = Sequential ([
    Conv2D (32, (3, 3), activation='relu', input_shape= (img_width, img_height, 1)),
    MaxPooling2D(2, 2),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Flatten(),
    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(len(labels), activation='softmax')
```

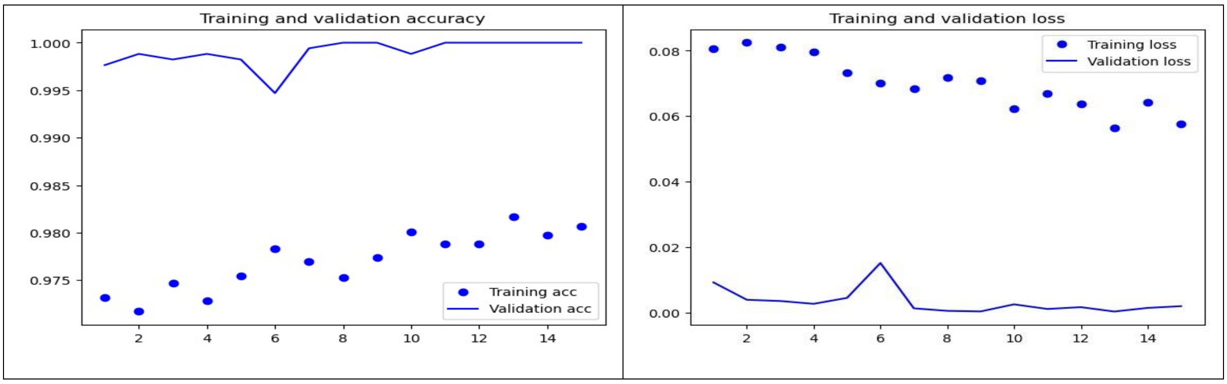
GUI FLOW DIAGRAM


Input Video
Image frames from input video
Preprocessing (using gaussian blur)
Prediction of alphabet using trained model
word formation
sentence formation
Translation using google translate api

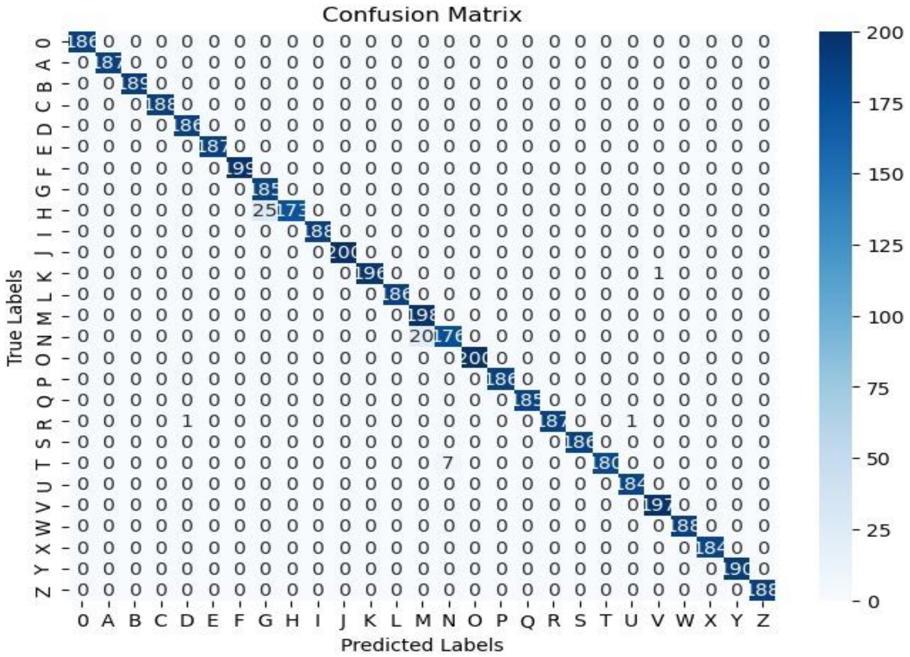
RESULTS AND DISCUSSION

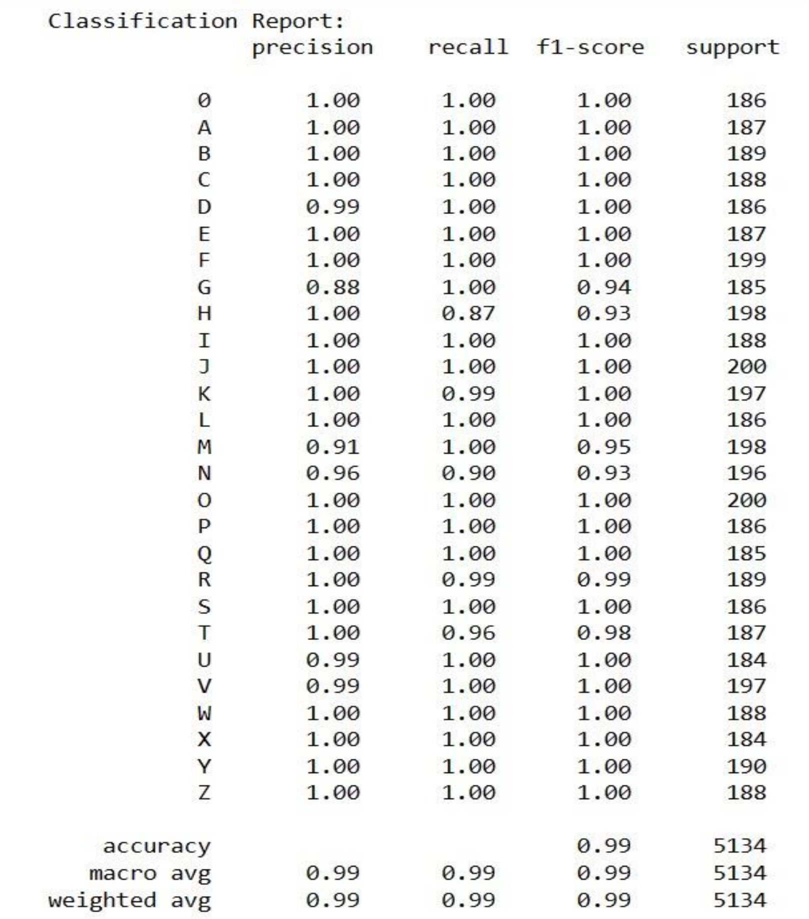
Training and Validation Accuracy:

Upon splitting our dataset into 90% for training and 10% for testing, we attained a peak training accuracy of 98.17% after completing 50 epochs of training.



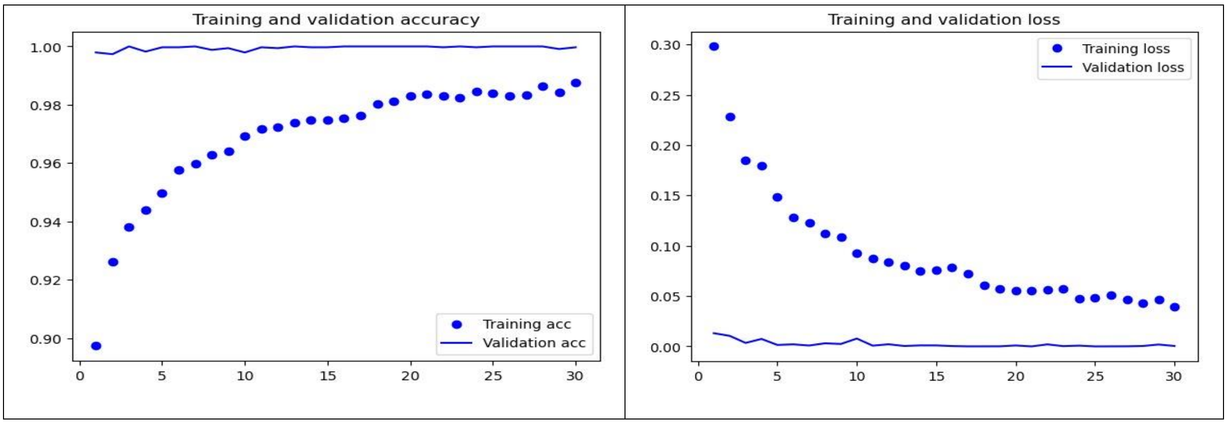
Confusion matrix and classification report



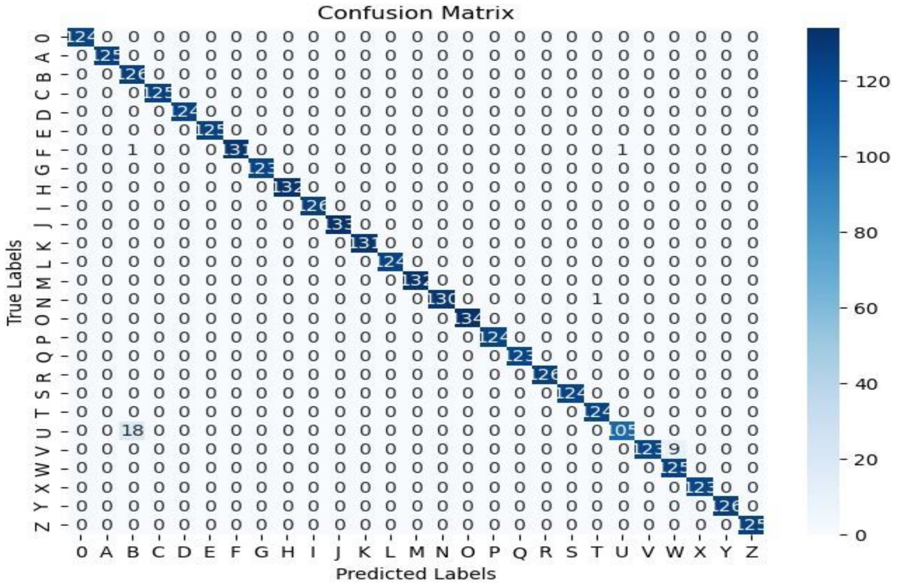


```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       186
           A       1.00      1.00      1.00       187
           B       1.00      1.00      1.00       189
           C       1.00      1.00      1.00       188
           D       0.99      1.00      1.00       186
           E       1.00      1.00      1.00       187
           F       1.00      1.00      1.00       199
           G       0.88      1.00      0.94       185
           H       1.00      0.87      0.93       198
           I       1.00      1.00      1.00       188
           J       1.00      1.00      1.00       200
           K       1.00      0.99      1.00       197
           L       1.00      1.00      1.00       186
           M       0.91      1.00      0.95       198
           N       0.96      0.90      0.93       196
           O       1.00      1.00      1.00       200
           P       1.00      1.00      1.00       186
           Q       1.00      1.00      1.00       185
           R       1.00      0.99      0.99       189
           S       1.00      1.00      1.00       186
           T       1.00      0.96      0.98       187
           U       0.99      1.00      1.00       184
           V       0.99      1.00      1.00       197
           W       1.00      1.00      1.00       188
           X       1.00      1.00      1.00       184
           Y       1.00      1.00      1.00       190
           Z       1.00      1.00      1.00       188

    accuracy                           0.99      5134
   macro avg       0.99      0.99      0.99      5134
weighted avg       0.99      0.99      0.99      5134
```

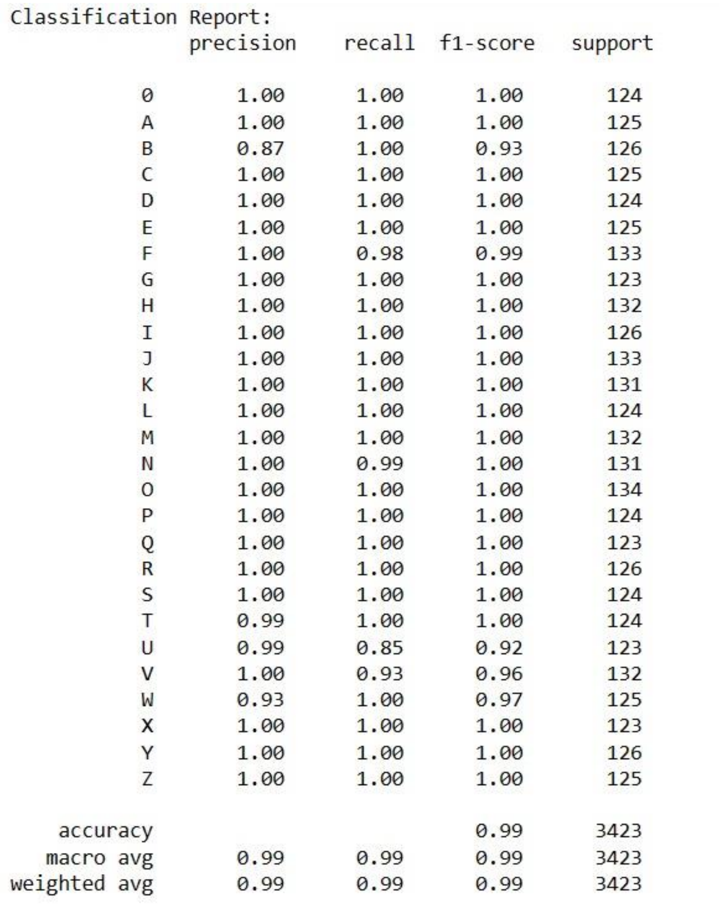
Upon splitting our dataset into 80% for training and 20% for testing, we attained a peak training accuracy of 98.75% after completing 50 epochs of training.



Confusion matrix and classification report



	0	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	124	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	125	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	126	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	125	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	124	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	125	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	1	0	0	0	131	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
G	0	0	0	0	0	0	0	123	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	0	132	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	0	126	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	0	133	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	0	0	131	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0	0	0	124	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	0	132	0	0	0	0	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	130	0	0	0	0	0	1	0	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	134	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	124	0	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	123	0	0	0	0	0	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	126	0	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	124	0	0	0	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	124	0	0	0	0	0	0
U	0	0	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	105	0	0	0	0	0
V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	123	9	0	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	125	0	0	0
X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	123	0	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	126	0
Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	125

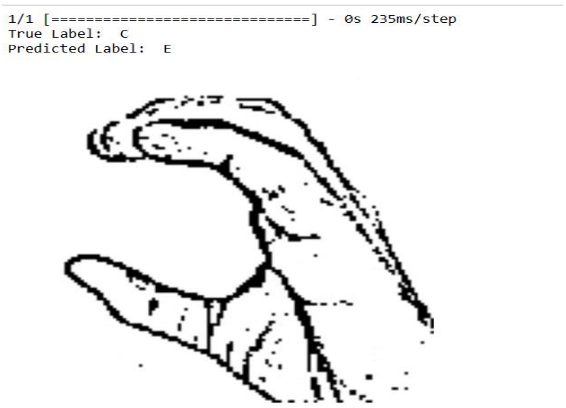


```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       124
           A       1.00      1.00      1.00       125
           B       0.87      1.00      0.93       126
           C       1.00      1.00      1.00       125
           D       1.00      1.00      1.00       124
           E       1.00      1.00      1.00       125
           F       1.00      0.98      0.99       133
           G       1.00      1.00      1.00       123
           H       1.00      1.00      1.00       132
           I       1.00      1.00      1.00       126
           J       1.00      1.00      1.00       133
           K       1.00      1.00      1.00       131
           L       1.00      1.00      1.00       124
           M       1.00      1.00      1.00       132
           N       1.00      0.99      1.00       131
           O       1.00      1.00      1.00       134
           P       1.00      1.00      1.00       124
           Q       1.00      1.00      1.00       123
           R       1.00      1.00      1.00       126
           S       1.00      1.00      1.00       124
           T       0.99      1.00      1.00       124
           U       0.99      0.85      0.92       123
           V       1.00      0.93      0.96       132
           W       0.93      1.00      0.97       125
           X       1.00      1.00      1.00       123
           Y       1.00      1.00      1.00       126
           Z       1.00      1.00      1.00       125

    accuracy                           0.99      3423
   macro avg       0.99      0.99      0.99      3423
weighted avg       0.99      0.99      0.99      3423
```

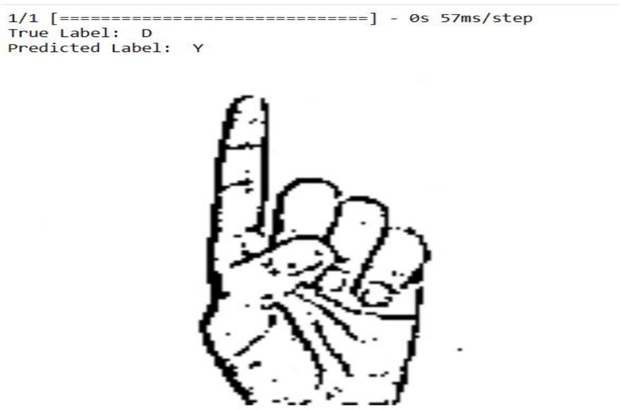
Now to model test the whether the model is actually working on random test data images we did manual testing on testing dataset before and after the training process

Manual testing on test data

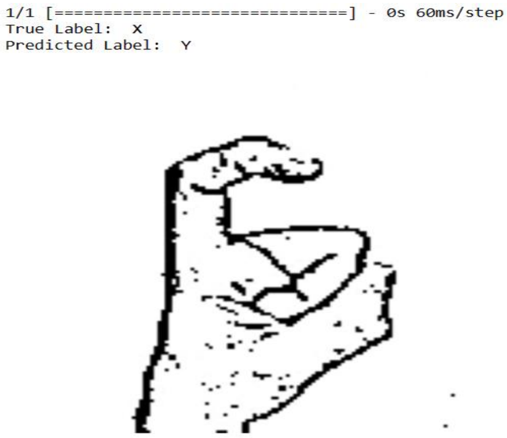
Before training:



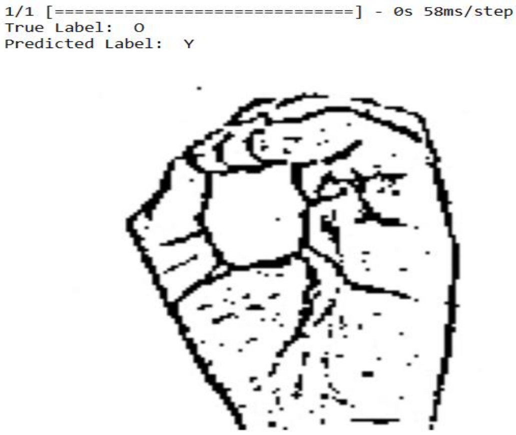
```
1/1 [==============================] - 0s 235ms/step
True Label:  C
Predicted Label:  E
```



```
1/1 [==============================] - 0s 57ms/step
True Label:  D
Predicted Label:  Y
```

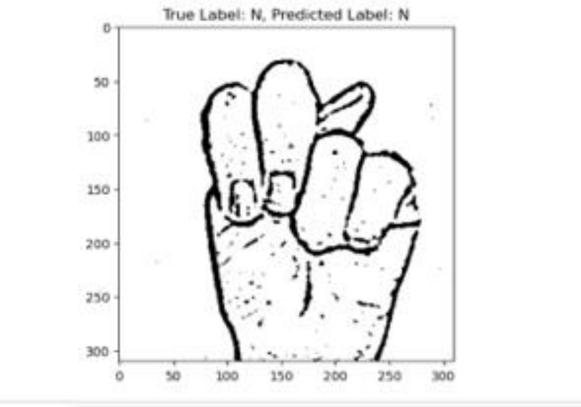


```
1/1 [==============================] - 0s 60ms/step
True Label:  X
Predicted Label:  Y
```

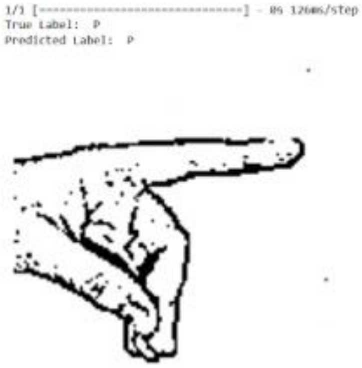


```
1/1 [==============================] - 0s 58ms/step
True Label:  O
Predicted Label:  Y
```

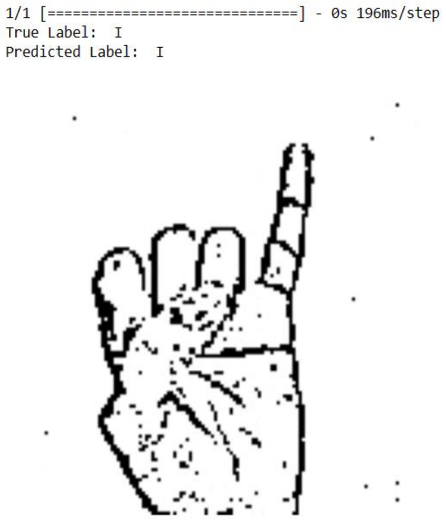
After training our model on our dataset using the CNN model we tested our model by selecting the random images from the test dataset Below are the attached screenshots.



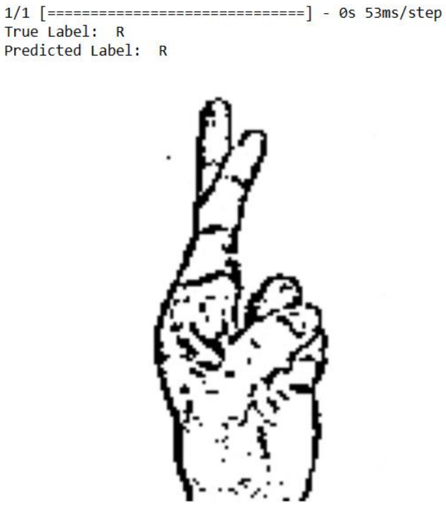
True Label: N, Predicted Label: N



```
1/1 [==============================] - 0s 126ms/step
True Label:  P
Predicted Label:  P
```



```
1/1 [==============================] - 0s 196ms/step
True Label:  I
Predicted Label:  I
```



```
1/1 [==============================] - 0s 53ms/step
True Label:  R
Predicted Label:  R
```

CONCLUSION AND FUTURE DIRECTIONS:

In Conclusion, our methodological framework which is centered on Convolutional Neural Networks (CNNs) that we trained on a dataset enriched with Gaussian blur, has aptly addressed the intricacies of sign language recognition. The integration of the Google Translate API represents a pivotal advancement, seamlessly facilitating translation into diverse languages and significantly enhancing accessibility. The deliberate incorporation of diverse training data, with a specific emphasis on Fingerspelling-based gestures, serves to underscore the robustness of our model.This conscientious approach positions our research at the forefront of advancements in sign
language recognition and translation technology. We
extend our gratitude for the opportunity to contribute to this evolving field of sign language to multilingual Indian Languages.

Future Work

1. Gesture Recognition in Noisy Environments:
We would work to Enhance the system's robustness to operate effectively in noisy environments. This could involve developing algorithms to filter out background noise and distractions that may affect the accuracy of gesture recognition at the same time increasing the robustness of our model.

2. Accessibility Features:
We can work on Integrating additional features such as voice output for translated text and support for visually impaired individuals to enhance the overall user experience.

3. Collaboration with Educational Institutions:
We could also collaborate with educational institutions and organizations to integrate the system into classrooms for deaf and hard-of-hearing students, providing a practical tool for learning and Communication to test our application for real-life scenarios.

LIMITATIONS OF OUR WORK:

1. The CNN model was trained on a dataset consisting of a Gaussian blur dataset. Hence it may not work well in noisy environments.

2. The model may not give accurate results in low-light conditions.

CITATIONS

1.He, Siming. (2019). Research of a Sign Language Translation System Based on Deep Learning. 392-396. 10.1109/AIAM48774.2019.00083.

2.Herath, H.C.M. & W.A.L.V.Kumari, & Senevirathne, W.A.P.B & Dissanayake, Maheshi. (2013). IMAGE BASED SIGN LANGUAGE RECOGNITION SYSTEM FOR SINHALA SIGN LANGUAGE

3.Pigou L., Dieleman S., Kindermans PJ., Schrauwen B. (2015) Sign Language Recognition Using Convolutional Neural Networks. In: Agapito L., Bronstein M., Rother C. (eds) Computer Vision – ECCV 2014 Workshops. ECCV 2014. Lecture Notes in Computer Science, vol 8925. Springer, Cham. https://doi.org/10.1007/978-3-319-16178-5_40

4.Jaoa Carriera, A. Z. (2018). Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on (pp. 4724-4733). IEEE. Honolulu.

5.Kodandaram, Satwik Ram & Kumar, N. & Gl, Sunil. (2021). Sign Language Recognition. 10.13140/RG.2.2.29061.47845.

6. E. S. Gedraite and M. Hadad, "Investigation on the effect of a Gaussian Blur in image filtering and segmentation," Proceedings ELMAR-2011, Zadar, Croatia, 2011, pp. 393-396.

7. John Joseph, Ferdin Joe & Nonsiri, Sarayut & Monsakul, Annop. (2021). Keras and TensorFlow: A Hands-On Experience. 10.1007/978-3-030-66519-7_4.

8. I. Culjak, D. Abram, T. Pribanic, H. Dzapo and M. Cifrek, "A brief introduction to OpenCV," 2012 Proceedings of the 35th International Convention MIPRO, Opatija, Croatia, 2012, pp. 1725-1730.

9. J. Peguda, V. S. S. Santosh, Y. Vijayalata, A. D. R. N and V. Mounish, "Speech to Sign Language Translation for Indian Languages," 2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2022, pp. 1131-1135, doi: 10.1109/ICACCS54159.2022.9784996.

10. R. Vyas, K. Joshi, H. Sutar and T. P. Nagarhalli, "Real Time Machine Translation System for English to Indian language," 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2020, pp. 838-842, doi: 10.1109/ICACCS48705.2020.9074265.

11. https://ambrapaliaidata.blob.core.windows.net/ai-storage/articles/1_hbrZSUS.png

12. https://iq.opengenus.org/content/images/2020/04/relu.png