Mini Project Report

on

REAL TIME SIGN LANGUAGE CONVERSION TO TEXT

COMPUTER ENGINEERING

Submitted by

SHREEYASH GAIKI

NAINA MAKHIJANI

YAMINI MIRASHE

SALONI SABLE

Under the guidance of

DR. KAPIL GUPTA

Associate Professor

Academic Year 2023-24

Department of Computer Engineering

ARISE & SHINE

ST. VINCENT PALLOTTI COLLEGE OF ENGINEERING AND TECHNOLOGY

Wardha Road, Gavsi Manapur, Nagpur

ST. VINCENT PALLOTTI COLLEGE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institute Affiliated to RTM University, Nagpur)
NAAC Accredited with 'A' Grade
Gavsi Manapur, Wardha Road, Nagpur – 441108

CERTIFICATE

This is to certify that **SHREEYASH GAIKI, NAINA MAKHIJANI, YAMINI MIRASHE AND SALONI SABLE** have completed mini project on "**REAL TIME SIGN LANGUAGE CONVERSION TO TEXT"** under my supervision for partial fulfilment of VI Semester, Bachelor of Technology COMPUTER ENGINEERING of ST. VINCENT PALLOTTI COLLEGE OF ENGINEERING AND TECHNOLOGY, NAGPUR.

Dr. S. M. Wanjari
Associate Professor
Head of the Department
Computer Engineering

Dr. Kapil Gupta
Assistant Professor
Project Guide
Computer Engineering

CONTENTS

ABSTRACT

Sign language serves as a vital means of communication for the hearing-impaired community, yet the lack of understanding among non-signers often erects barriers to inclusive interaction. Our project aims to dismantle these barriers by crafting a real-time sign language conversion system. By seamlessly translating sign language gestures into text, we empower both sign language users and non-signers to engage effortlessly in meaningful dialogue.[7] This initiative not only facilitates communication but also fosters empathy and understanding across diverse linguistic communities. Our user interface will be intuitively designed, featuring a camera screen to capture real-time hand gestures and a textbox to display the corresponding text output [3]. This interface will serve as a gateway to bridging the communication chasm, fostering empathy, and fostering a more inclusive society where all voices are heard and understood. For this, we are building a Convolutional Neural Network (CNN) model capable of accurately recognizing and interpreting sign language gestures in real-time. This machine learning model which will identify Numbers (0 to 9) and Black space.

Chapter 1:

INTRODUCTION

1.1 INTRODUCTION

The proposed project aims to solve the communication barrier between people who have communication related problem use the sign language. Hence, we have come up with a real time ML model using Neural Networks for hand based American sign language. Automatic hand gestures recognition from video converted into stream of frames.

We propose a Convolution Neural Network (CNN) method to recognize to hand gestures. Each frame form video will be feed to model to recognize the corresponding letter or number. Frames are first passed through the filter and then passed to the CNN classifier which classifies the frame based on its properties and then predicts the class of the hand gesture. This system introduces efficient and fast techniques for identifying the hand gestures representing sign language meaning in real time.

Our proposal is a Convolution Neural Network CNN-based technique for hand gesture recognition. The model will be fed each frame from the video in order to identify the appropriate letter or number. After going through the filter, frames are sent to the CNN classifier, which uses its features to classify the frame and then guesses which hand gesture class it belongs to. This system presents quick and easy methods for recognizing the hand motions that indicate the meaning of sign language in real time gesture is the motion of a hand. Here, we employ computer vision and image processing to identify point recognized gestures, which let computers comprehend human behavior and serve as a mediator between them. This could enable users to communicate with computers without having to physical interaction with any piece of machinery. The community uses sign language, which is created in the congregation's language for the deaf, in circumstances when voice is not an option for communication or where writing and typing are challenging yet visible.

At that time, people could only communicate with one another through words. Although most individuals only utilize their languages as a means of communication when they are unable to speak, this is the case for the deaf population. The meaning of symbols is identical to that of spoken words. The deaf use this all over the world, but is l, as l, etc. One or both hands might be used to move the sign language when it is employed in regional variants. It comes with expanded instructions and two distinct sign languages.

In isolated sign language, a word is accompanied by a gesture, while in continuous sign language, the main statement consists of a sequence of movements. We employ typical representational asl hand motions in this study.

1.2 OBJECTIVES

- For the community of hearing-impaired people, sign language is an essential form of communication. However, non-signers' ignorance of sign language frequently creates obstacles to inclusive engagement.

- Our project's goal is to create a real-time sign language conversion system in order to remove these obstacles.

- We enable both sign language users and non-signers to easily participate in meaningful dialogue by translating sign language movements into text.

- This program promotes empathy and understanding amongst various language communities in addition to facilitating communication.

- The goal of our project is to improve the lives of those who are deaf and mute. We hope to enable individuals to express themselves freely and interact with others more successfully by providing them with a tool that makes clear communication possible.

- This will encourage social inclusion and break down barriers.

- By providing them with a tool that makes communication easier, we hope to enable individuals to interact with others and express themselves freely, fostering social inclusion and dismantling social barriers.

- With a textbox to show the corresponding text output and a camera screen to record hand motions in real time, our user interface will be intuitively constructed.

- This interface will act as a bridge to improve empathy, close the gap in communication, and create a more welcoming community where everyone's opinions are valued and heard.

- We are developing a Convolutional Neural Network (CNN) model to do this, one that can recognize and interpret sign language motions in real time with accuracy. This machine learning model can distinguish between black space and numbers (0 to 9).

Chapter 2:

LITERATURE REVIEW

2.1 LITERATURE REVIEW

- Data Collection:

  A large and diverse dataset of sign language is needed to make the project. The data collected is used to train and test machine learning models responsible for recognizing gestures and converting them to text [8]. Initially, we gathered data from Kaggle to train our model. But it was not giving good accuracy so we had to create our own dataset with 400 images of each gesture. Total 4.4k images. The images are taken by each possible way in a plain light background and good lighting.

- Data Pre-processing:

  An essential step in developing sign-to-text converters is preprocessing motion pictures. Images are preprocessed in order to facilitate the recognition of gestures and their translation into text by machine learning models. Motion images have first been adjusted in size and normalization to prepare them for use with machine learning models. Reduce the size of the images so that the model can work with them more easily [8]. The purpose of the normalizing stage is to get rid of any color variations in the image, background, or lighting that can have a negative impact on the model's performance. To make sure the model has the same orientation as the original, the image can be subjected to transformation operations like cropping or rotating in addition to scaling and normalizing.

- Labelling the Gestures:

  This stage involves assigning a label to each image in the data that corresponds to the gesture(number) it represents. The information gathered during this procedure is crucial because it gives the machine learning model the tools it needs to identify motions and convert them into text. Machine learning models can be trained and tested using recorded movements. The model can recognize movements and accurately interpret letters as we have already labelled the gestures in the training data. It gives knowledge to the model it to understand the relationship between gestures and text(label).

- Convolutional Neural Networks:

  CNN is primarily used for image classification. CNNs consist of multiple convolutional layers each layer containing numerous “filters” which perform feature extraction. Initially these “filters” are random and by training, the feature extraction gets better. During training, the CNN learns to optimize its internal parameters to accurately map input frames to corresponding gestures. Adjustments are made to improve the model's accuracy and generalization.

- User Interface:

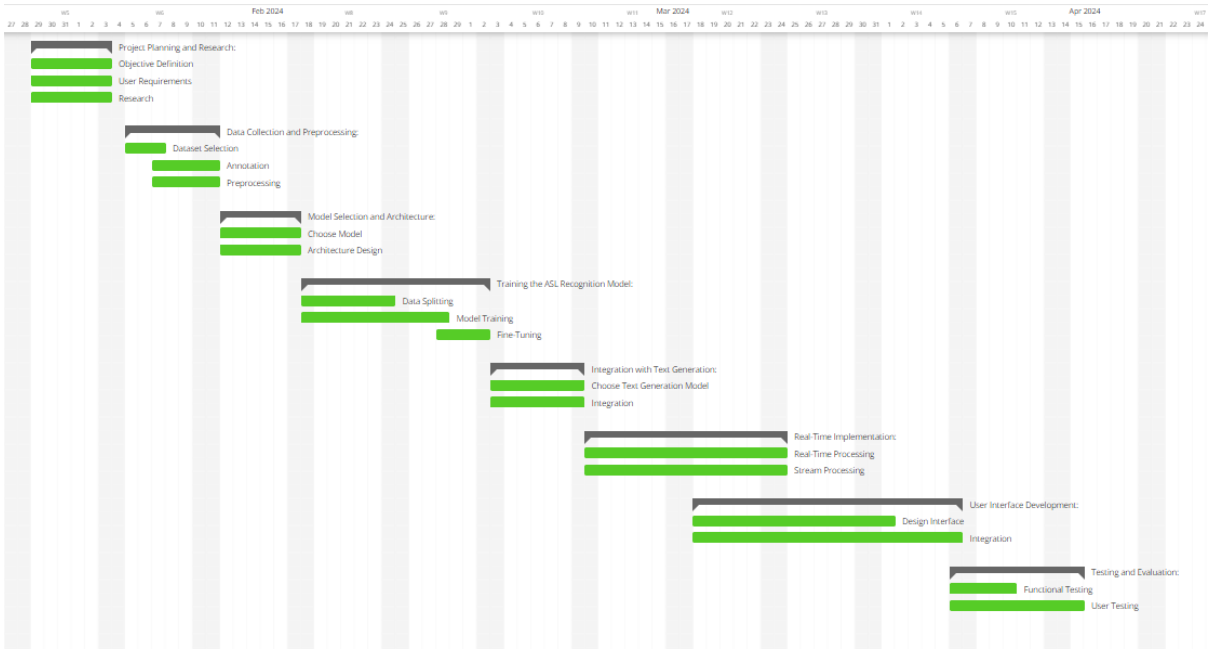
  A user interface is a desktop interface that is used to allow the user to communicate with the model. User Interface can be made in many ways but Python has a Tkinter library which is famous in lot of learners and developers because it is quick and convenient to use. Tkinter offers a number of tools and widgets for making desktop graphical programs. The majority of Python installations come with Tkinter, which makes it simple for developers to use for creating graphical user interfaces (GUIs) without requiring further installations or libraries.

  The User Interface should have a specified box where the user needs to put his hand while doing any hand gesture. Also, a text box should be there to show the output(text).

Chapter 3:

PROJECT PLANNING AND SCHEDULING

3.1 GANTT CHART



This Gantt Chart shows the planning and scheduling of the project.
The project started at the end of January 2024 and was completed in mid-April 2024.
It took approximately 2 ½ - 3 months to complete the project.

The project is mainly divided into 8 stages.

- First is Project Planning and Research in which the objectives and user requirements are identified.
- After that the data is collected and data preprocessing is done in the Data Collection and Preprocessing stage.
- Then the Model selection takes place and Model Architecture is made.
- For Model Training, first we split the data into training and testing datasets. 80% was training data and 20% were testing data. These testing data are randomly selected from the full dataset. Then model is trained and fine-tuned using the training dataset.
- After model training, we choose a text generation model and integrate it with our model.
- Then using real-time video stream, the model is implemented.
- User Interface is developed and integrated.
- Finally functional testing, unit testing and integration testing are done and full project is evaluated.

Chapter 4:

REQUIREMENT ANALYSIS

4.1 SOFTWARE REQUIREMENTS

1. **Python:** Programming Language commonly used in machine learning and deep learning concepts.
2. **Operating System**: Any major operating system supported by the software requirements (e.g., Windows, macOS, or Linux)
3. **TensorFlow:** Machine Learning Framework for implementing Convolutional Neural Networks.
4. **OpenCV:** Computer Vision Library for image processing tasks.
5. **Pandas**: Pandas is an essential library in Python that facilitates data handling, manipulation, and analysis. Used for data loading, exploring, cleaning and preprocessing.
6. **NumPy:** Python library for numerical computing that provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently. Used for data representation, mathematical operations Integration with Machine Learning Libraries.
7. **Keras:** Deep Learning Model Framework that provides a high-level interface to TensorFlow.
8. **Jupyter Notebook or Visual Studio Code**: Development Environment for coding and experimentation.

4.2 HARDWARE REQUIREMENTS

1. **Camera**: High-definition webcam or camera for capturing sign language gestures during model testing and real-time inference.
2. **Memory (RAM)**: 16 GB or more for handling large datasets and model training.
3. **Laptop or Desktop PC:** Laptops and desktop PCs that offers computational power, memory, and storage capabilities necessary for training machine learning models and processing data efficiently.

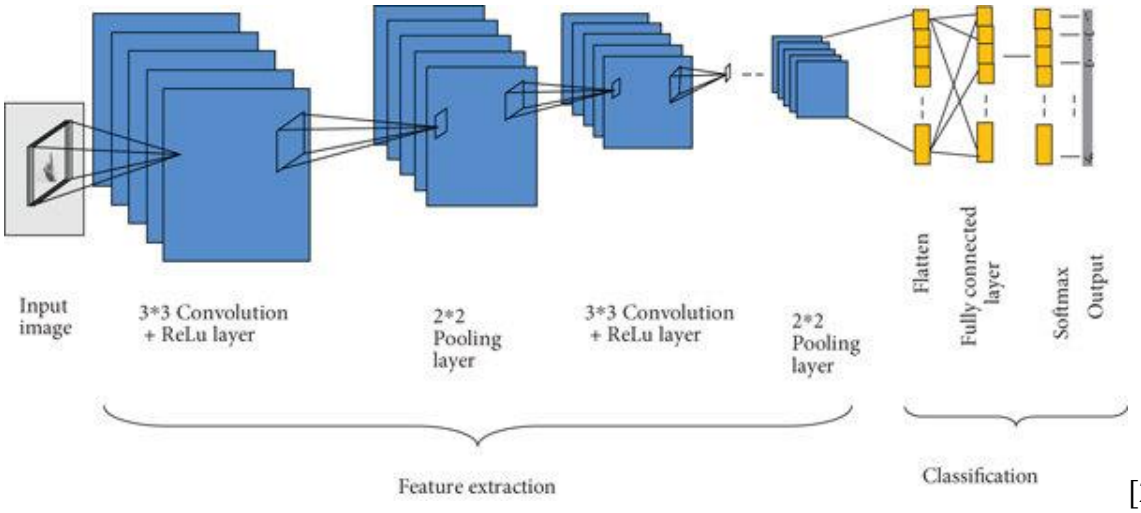
Chapter 5:

SYSTEM DESIGN AND ANALYSIS

5.1 SYSTEM ARCHITECTURE DESIGN

MODEL ARCHITECTURE

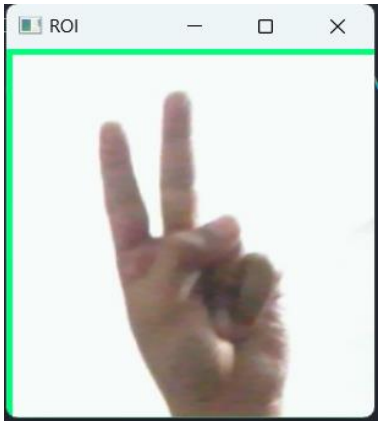
The project involves the design of a convolutional neural network (CNN) model using the Keras framework. The model comprises several layers, including convolutional layers, max-pooling layers, dropout layers, and dense layers[2]. Each layer is responsible for performing specific operations, such as feature extraction, dimensionality reduction, and classification.



[2]

DATA PROCESSING FUNCTIONS

The code includes functions for loading and preprocessing images from the dataset. These functions handle tasks such as resizing images, converting them to grayscale, and normalizing pixel values. Additionally, the code implements one-hot encoding for categorical labels to prepare the data for model training.

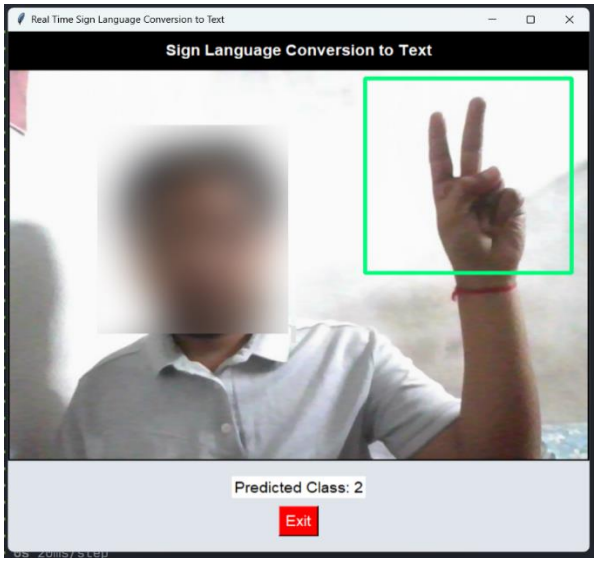


Raw image captured from camera.

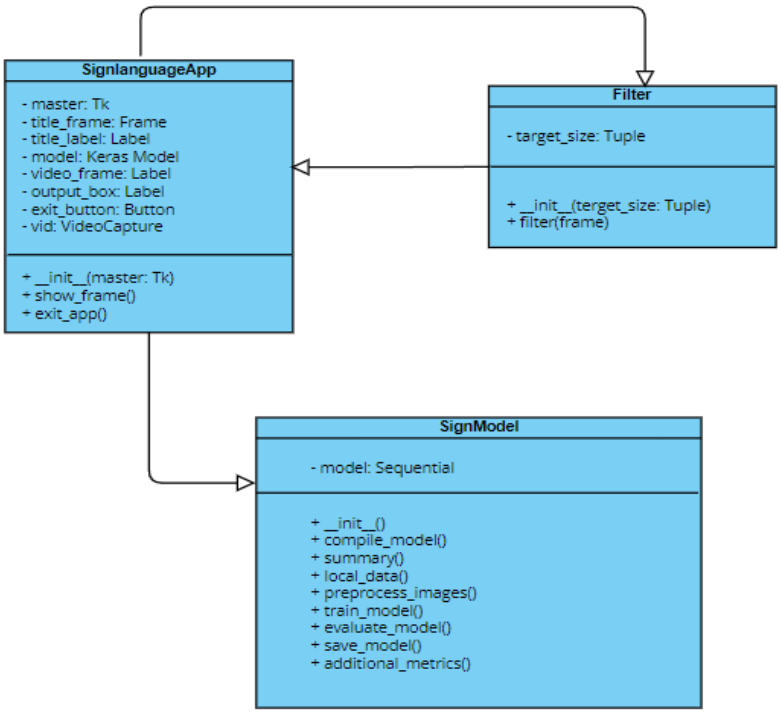
Processed image

USER INTERFACE

The application file contains components for creating a graphical user interface (GUI) using the Tkinter library[4]. These components include title labels, video frames for displaying webcam input, output boxes for showing predicted classes, and an exit button for closing the application.

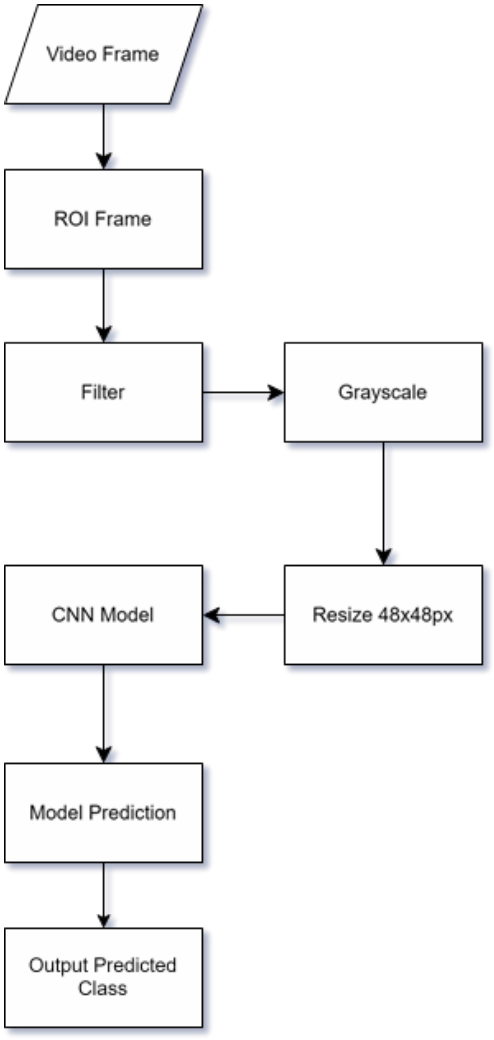


CLASS DIAGRAM



5.2 DATA FLOW DIAGRAM

During real-time prediction, the webcam captures video frames, which are then processed and filtered using the ‘filter ()’ function. The filtered frame is pre-processed and passed through the trained CNN model to predict the sign language gesture being performed. The predicted class is displayed in the output box of the GUI, providing real-time feedback to the user.

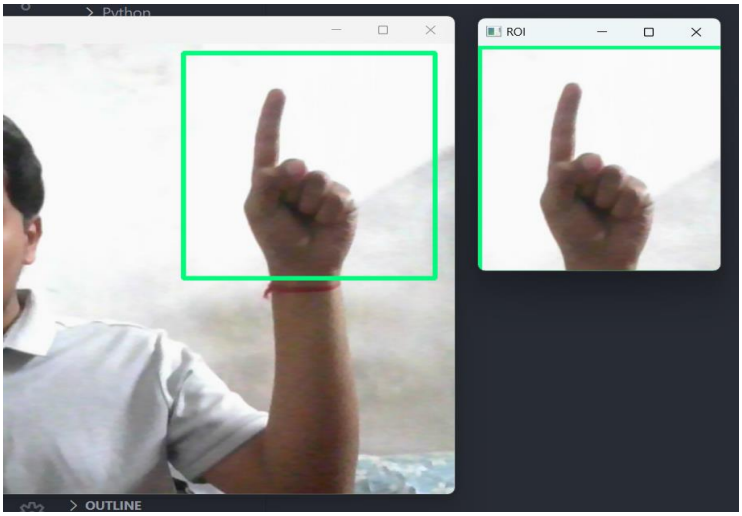


5.3 IMPLEMENTATION

DATASET GENERATION

For the project, we tried to find the already made datasets available on Kaggle but training the model on that dataset and testing it on completely different data i.e. our images, due to which model was predicting the wrong output every time. Hence, we decided to create our own dataset from scratch.

For the dataset generation we used OpenCV library. Firstly, we captured 400 images for each symbol (0-1) and a simple background image for ‘blank’. 320 images were used for training the data and remaining 80 for testing. We have added multiple background images so that for different backgrounds model will predict the blank symbol.



Data collection python script.

First, we capture the image by pressing the corresponding symbol button on the keyboard. Then the defined region of interest (ROI) frame is extracted, filtered and saved in the corresponding file for that symbol.

Created dataset.

CNN MODEL

Convolutional Layers

- The first convolutional layer (Conv2D) with 32 filters of size 3x3, applies ReLU activation, and expects input images of shape (48, 48, 1).
- It's followed by a max-pooling layer (MaxPooling2D) with a pool size of 2x2.
- The second convolutional layer with 64 filters of size 3x3, applies ReLU activation.
- Another max-pooling layer follows with a pool size of 2x2.
- The third convolutional layer with 128 filters of size 3x3, applies ReLU activation.
- Finally, another max-pooling layer with a pool size of 2x2.

Flatten Layer

- Flattens the output from the convolutional layers into a 1D array to feed into the dense layers[11].

Dense layers

- The first dense layer (Dense) with 128 neurons applies ReLU activation.
- A dropout layer (Dropout) with a dropout rate of 0.5 is added to prevent overfitting.
- The second dense layer with 64 neurons applies ReLU activation.

Output Layer

- The output layer (Dense) has 11 neurons, corresponding to the 11 classes in the dataset (0-9 digits and a blank symbol).
- It uses SoftMax activation to produce class probabilities.

REAL TIME PREDICTION

The process begins with the application capturing video frames from the webcam in real-time. This is achieved using the VideoCapture() function from the OpenCV library, which accesses the webcam feed and retrieves successive frames.

Each captured frame undergoes preprocessing to enhance its suitability for sign language recognition. The filter() function is applied to the frame, which performs operations such as converting the frame to grayscale, resizing it to the required dimensions, and applying any necessary filters to improve image quality and reduce noise[3].

The prepared frame data is fed into the CNN model for prediction. The model processes the input data through its layers, performing feature extraction, pattern recognition, and classification tasks. The output layer of the model generates a prediction for the sign language gesture depicted in the frame.

The predicted class or gesture label generated by the model is extracted and displayed to the user in real-time. This is typically achieved by updating a designated area or output box within the graphical user interface (GUI) of the application. The user can observe the predicted gesture label as it changes dynamically with each processed frame.

By following this flow of data, the application effectively transforms live webcam input into meaningful predictions of sign language gestures, providing users with immediate feedback and facilitating communication through sign language interpretation[6].

Chapter 6:

TRAINING AND TESTING

6.1 TRAINING

The training data flow begins with loading images from the training directory using the load_images() function. We have already processed the whole dataset already while creating it. Images were read, resized, and converted to grayscale before being pre-processed and added to the training set. Labels are encoded using one-hot encoding to match the model's output format. The pre-processed training data is then fed into the CNN model for training.

The fit() function is called to train the model using the training data (train_images and train_labels).
Training is performed for 20 epochs with a batch size of 32.
The validation data (test_images and test_labels) are specified to monitor model performance during training.

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 46, 46, 32)	320
max_pooling2d (MaxPooling2D)	(None, 23, 23, 32)	0
conv2d_1 (Conv2D)	(None, 21, 21, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 10, 10, 64)	0
conv2d_2 (Conv2D)	(None, 8, 8, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 128)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 128)	262,272
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8,256
dense_2 (Dense)	(None, 11)	715

Total params: 363,915 (1.39 MB)
Trainable params: 363,915 (1.39 MB)
Non-trainable params: 0 (0.00 B)

Model summary.

```
Epoch 5/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 3s 23ms/step - accuracy: 0.8296 - loss: 0.5127 - val_accuracy: 0.9227 - val_loss: 0.2708
Epoch 6/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 2s 22ms/step - accuracy: 0.8584 - loss: 0.3882 - val_accuracy: 0.9409 - val_loss: 0.1914
Epoch 7/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 3s 24ms/step - accuracy: 0.8907 - loss: 0.3191 - val_accuracy: 0.9773 - val_loss: 0.1279
Epoch 8/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 2s 21ms/step - accuracy: 0.9251 - loss: 0.2325 - val_accuracy: 0.9830 - val_loss: 0.0753
Epoch 9/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 3s 23ms/step - accuracy: 0.9444 - loss: 0.1641 - val_accuracy: 0.9864 - val_loss: 0.0639
Epoch 10/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 2s 21ms/step - accuracy: 0.9371 - loss: 0.1678 - val_accuracy: 0.9909 - val_loss: 0.0528
Epoch 11/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 2s 21ms/step - accuracy: 0.9593 - loss: 0.1336 - val_accuracy: 0.9852 - val_loss: 0.0670
Epoch 12/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 2s 22ms/step - accuracy: 0.9572 - loss: 0.1413 - val_accuracy: 0.9909 - val_loss: 0.0394
Epoch 13/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 2s 21ms/step - accuracy: 0.9672 - loss: 0.1001 - val_accuracy: 0.9943 - val_loss: 0.0425
Epoch 14/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 2s 22ms/step - accuracy: 0.9741 - loss: 0.0835 - val_accuracy: 0.9932 - val_loss: 0.0337
Epoch 15/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 2s 21ms/step - accuracy: 0.9638 - loss: 0.1020 - val_accuracy: 0.9943 - val_loss: 0.0238
Epoch 16/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 2s 21ms/step - accuracy: 0.9669 - loss: 0.0965 - val_accuracy: 0.9920 - val_loss: 0.0444
Epoch 17/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 2s 22ms/step - accuracy: 0.9723 - loss: 0.0733 - val_accuracy: 0.9955 - val_loss: 0.0385
Epoch 18/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 2s 20ms/step - accuracy: 0.9669 - loss: 0.0879 - val_accuracy: 0.9943 - val_loss: 0.0257
Epoch 19/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 2s 21ms/step - accuracy: 0.9753 - loss: 0.0765 - val_accuracy: 0.9955 - val_loss: 0.0317
Epoch 20/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 2s 22ms/step - accuracy: 0.9763 - loss: 0.0670 - val_accuracy: 0.9966 - val_loss: 0.0219
```

6.2 TESTING

After training, the model's performance is evaluated on the test set using the evaluate() function. Loss and accuracy metrics are computed and printed to assess the model's performance. The model's predictions are generated for the test set using the predict() function. Classification metrics such as confusion matrix and classification report are computed using scikit-learn functions (confusion_matrix and classification_report) to further evaluate the model's performance.

We achieved the accuracy of 99.65% on our test data.

```
Loss on test set: 0.021949686110019684
Accuracy on test set: 0.9965909123420715
```

```
Confusion Matrix:
[[79  0  0  0  0  1  0  0  0  0  0]
 [ 0 80  0  0  0  0  0  0  0  0  0]
 [ 0  0 80  0  0  0  0  0  0  0  0]
 [ 0  0  0 80  0  0  0  0  0  0  0]
 [ 0  0  0  0 80  0  0  0  0  0  0]
 [ 0  0  0  0  1 79  0  0  0  0  0]
 [ 0  0  0  0  0  0 80  0  0  0  0]
 [ 0  0  0  0  0  0  0 80  0  0  0]
 [ 0  0  0  0  0  0  0  1 79  0  0]
 [ 0  0  0  0  0  0  0  0  0 80  0]
 [ 0  0  0  0  0  0  0  0  0  0 80]]
```

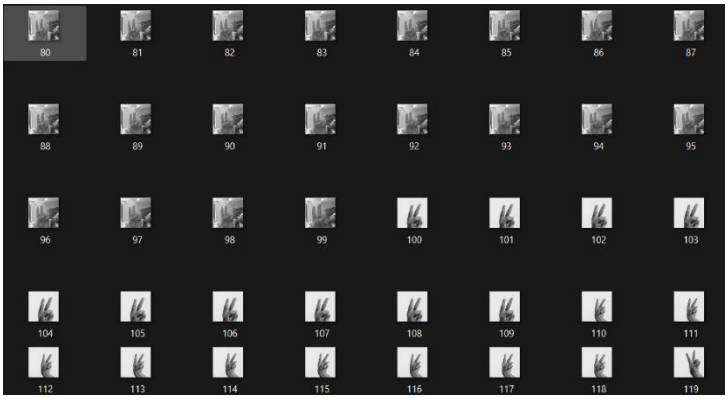
Confusion matrix

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

           0       1.00      0.99      0.99        80
           1       1.00      1.00      1.00        80
           2       1.00      1.00      1.00        80
           3       1.00      1.00      1.00        80
           4       0.99      1.00      0.99        80
           5       0.99      0.99      0.99        80
           6       1.00      1.00      1.00        80
           7       0.99      1.00      0.99        80
           8       1.00      0.99      0.99        80
           9       1.00      1.00      1.00        80
          10       1.00      1.00      1.00        80

    accuracy                           1.00       880
   macro avg       1.00      1.00      1.00       880
weighted avg       1.00      1.00      1.00       880
```

Classification report

6.3 CHALLENGES FACED

There were many challenges faced during the project development.
The very first issue was the dataset. Already available dataset was of no use to us, so we had to create our own.



Second issue was to select an appropriate filter for which we could apply on dataset so that proper features of the image could be extracted. We tried many filters but most of them were making data noisy and was confusing the image. Finally, we selected the grayscale filter, and we also resized the image to 48x48px.

Rejected filters.

Used filter.

The last issue was the model accuracy, at start when we trained the model, the accuracy was very low. Then we increased the dataset size and variation in the dataset, which improved the model accuracy.

Chapter 7:

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

The project's main goal is to find a solution for the deaf and dumb population. This technology will save effort and improve accuracy and time efficiency by automating the laborious work of identifying sign language, which is difficult for the average person to grasp. We are attempting to construct this system using a variety of image-processing concepts, libraries, and fundamental picture features. This research described a vision-based system that can translate hand movements from sign language into text. Real-time testing of the suggested system allowed for the demonstration of the obtained CNN models' hand gesture recognition capabilities.

7.2 FUTURE SCOPE

In the future, we can increase the accuracy and efficiency of the model by adding more images to the dataset. Images having different backgrounds can also be added so that in any different background the model can be used and the correct gesture will be predicted.

Currently, our model can predict only numbers (0 to 9) and blank space. Further, we can extend it by adding the English alphabet letters (A to Z) also.

We can also attempt to diversify this machine learning model by training it on datasets for Indian and British sign languages, in addition to the American sign language dataset.

REFERENCES

[1] Hsien-I Lin; Ming-Hsiang Hsu; Wei-Kai Chen, "Human hand gesture recognition using a convolution neural network," *2014 IEEE International Conference on Automation Science and Engineering (CASE)*, New Taipei, Taiwan, 2014, pp. 1038-1043, doi: 10.1109/CoASE.2014.6899454.

[2] Pathan, R.K., Biswas, M., Yasmin, S. *et al.* Sign language recognition using the fusion of image and hand landmarks through multi-headed convolutional neural network. *Sci Rep* 13, 16975 (2023).

[3] Reddygari Sandhya Rani, R Rumana, R. Prema, 2021, A Review Paper on Sign Language Recognition for The Deaf and Dumb, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 10, Issue 10 (October 2021)

[4] Mehreen Hurroo, Mohammad Elham, 2020, "Sign Language Recognition System using Convolutional Neural Network and Computer Vision", INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 09, Issue 12 (December 2020)

[5] Obaid, Falah & Babadi, Amin & Yoosofan, Ahmad. (2020). Hand Gesture Recognition in Video Sequences Using Deep Convolutional and Recurrent Neural Networks. Applied Computer Systems. 25. 57-61. 10.2478/acss-2020-0007.

[6] S. Thakar, S. Shah, B. Shah and A. V. Nimkar, "Sign Language to Text Conversion in Real Time using Transfer Learning," 2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT), Bangalore, India, 2022, pp. 1-5, doi: 10.1109/GCAT55367.2022.9971953.

[7] Avina, Vijayendra & Amiruzzaman, Md & Amiruzzaman, Stefanie & Ngo, Linh & Dewan, M.. (2023). An AI-Based Framework for Translating American Sign Language to English and Vice Versa. Information. 14. 569. 10.3390/info14100569.

[8] Adnan Md Ashpak Qureshi1, Athrava Manoj Kargirwar2 , Ishaan Iqbal Sheikh3 , Kartik Sadashiv Tummawar4 , Mohit Sharad Mandhalkar5 , Prof. Anand D. G. Donald6**.** Sign Language to Text Language Conversion. International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal Volume 3, Issue 3, June 2023

[9] https://ijarsct.co.in/Paper3657.pdf

[10] https://apps.apple.com/us/app/ai-sign-sign-language/id1663187476

[11] https://en.wikipedia.org/wiki/American_Sign_Language

[12] Yamashita, R., Nishio, M., Do, R.K.G. *et al.* Convolutional neural networks: an overview and application in radiology. *Insights Imaging* **9**, 611–629 (2018).

PROJECT GUIDE AND TEAM MEMBERS

PROJECT GUIDE

Dr. Kapil Gupta
Assistant Professor
Project Guide
Computer Engineering

TEAM MEMBERS

	Roll no.	Name of Team Member	Email
1.	15	Naina Makhijani	nainamakhijani07@gmail.com
2.	19	Saloni Sable	salonisable586@gmaail.com
3.	26	Yamini Mirashe	yaminimirashe20@gmail.com
4.	57	Shreeyash Gaiki	shreeyashgaiki1339@gmail.com