AI4LIFE project report
Sign Language Recognition and Intent Detection

Authors
19CS10035 / Jatoth Charansai
19EC10036 / Shivesh Chowdary
19CS30026 / Lavoori Vamshi
19CS10064 / Suguna Bhaskar
18CS30035 / Rangoju Sai Kumar

Instrucotr
Ankan Mullick
ankanm@kgpian.iitkgp.ac.in

Abstract

This project report outlines a Natural Language Processing (NLP) model capable of identifying intents from sign language. An API was created to incorporate the trained models and implement them. The primary objective of the API is to facilitate communication between individuals who are deaf and those who do not understand sign language. It achieves this goal by converting sign images to text messages and text messages to sign videos. The project also included the development of a Video Classification model.

1 Introduction

While voice agents like Alexa are capable of identifying the user's voice and categorizing their intention to execute the relevant task, individuals who cannot speak may encounter difficulties. To address this problem, we propose the development of an agent that can recognize intents from sign language. Additionally, communication barriers between deaf individuals and those who do not comprehend sign language are prevalent. Therefore, we aim to create a model that can recognize sign language to facilitate communication.
This project involved the development of two methods to address the problem of detecting intents from sign language. The first method consisted of two modules: one for sign language detection from images and the other for intent classification from text. The second method involved predicting intents directly from video. Additionally, the first model was integrated into an API that can convert sign images to text messages and text messages to sign videos, facilitating communication between deaf and hearing individuals. Another approach involved classifying intents directly from sign language without using separate modules. The three models were trained and tested, and results were obtained. The main objective of the project was to develop and evaluate these three models based on the aforementioned methods.
The report introduces a deep learning algorithm for video classification applied to a sports/activity classification dataset. The goal is to assign one or more labels to a given video based on its content. The dataset used in this project includes six categories of images, namely badminton, weightlifting, cricket, hockey, ice hockey, and basketball. This task has diverse applications, such as security surveillance and sports analysis.

2 Datasets

2.1 Sign Language MNIST

The Sign Language MNIST dataset is designed as a replacement for the classic MNIST image dataset. It includes 24 classes of hand gestures representing alphabetic letters A-Z, excluding J and Z. The dataset consists of 27,455 training cases and 7,172 test cases, each containing a single 28x28 grayscale image with pixel values between 0-255. The images were generated using an image pipeline that included cropping, grayscale conversion, resizing, and modification of filters, pixelation, brightness/contrast, and rotation.

2.2 SNIPS dataset

The SNIPS Natural Language Understanding benchmark is a dataset of 16,000+ queries in 7 user intents of varying complexity. These intents include SearchCreativeWork, GetWeather, BookRestaurant, PlayMusic, AddToPlaylist, RateBook, and SearchScreeningEvent. The training set has 13,084 utterances, while the validation and test sets contain 700 utterances each, with 100 queries per intent. The dataset was crowdsourced, ensuring a diversity of language use and query types. It serves as a benchmark for developing natural language understanding models for practical applications such as chatbots and voice assistants.

2.3 sports/activity classification dataset

The dataset contains the images of different classes such as cricket(763), wrestling, Tennis, Badminton, Soccer, Swimming, and Karate with total of 8227 train files and 2056 test files. Dataset is collected from Kaggle

3 Experimental Settings

3.1 Sign Language classification task

Desined a Convolutional Neural Network (CNN) model inspired from ResNet architecture to classify sign language represented in images. The dataset used in this code is "Sign Language MNIST" which contains 27,455 labeled images of hand gestures, representing 24 classes of ASL letters from A to Z, and including an additional class for space.
The data is well balance when the plots of the distribution of the classes to show the number of examples in each class was generated and observed. To represent space in the input text, a white image was used. As the letters J and Z are gestures and not present in the dataset, black images were used to represent them during training.
After preprocessing the data, the ResNet-8 CNN model is created and compiled with an optimizer, loss function, and evaluation metric. Then, the model is trained on the training set with a validation split of 0.25 and 25 epochs.
Overall, the goal of this code is to build a CNN model that can accurately classify the hand signs into their respective letters using the Sign Language MNIST dataset.
Now this ends the first module and also we can use this model for the communication purpose that we discussed in the introduction.

3.2 Intent detection model

3.2.1 Preprocessing

The pre-processing steps applied to the data included removing numbers and creating dictionaries to map the categorical labels to integers. The training data was then split into a smaller subset to train the model on. To convert the text data from the SNIPS dataset into a sequence of images, we combined the information in the MNIST and SNIPS datasets. Specifically, each letter in the SNIPS text was replaced by a sign randomly selected from the images corresponding to that letter present in the MNIST dataset. This resulting dataset was then saved for further training in method 1.

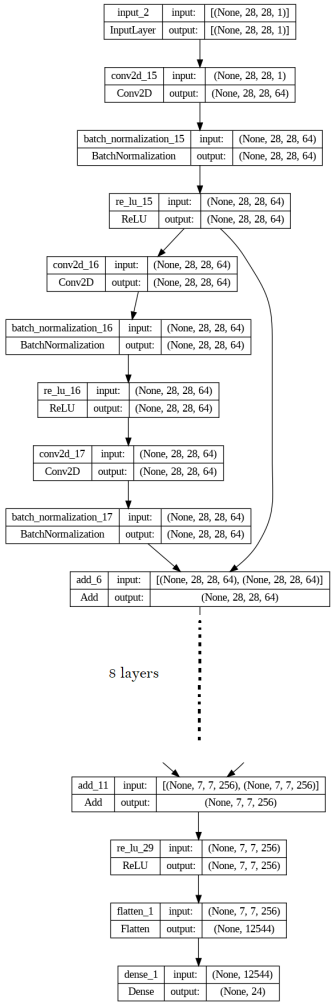

Figure 1: Sign language classification model architecture

3.2.2 Method 1 (CNN + LSTM)

Implementation of a deep learning model for Intent Detection directly from a video. The data set corresponding to this task was not available online hence we used the data that we created in the preprocessing step i.e combined data of Snips and MNIST. The model uses a combination of time distributed Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) layers. The input to the model is a sequence of images that represent sign language, and the output is the predicted label of the sign. .
The main motivation behind this idea was to create an embedding vector for sign language images, which could be used to predict the intent behind the sign language. The time-distributed CNN was used to take the image and output the corresponding embedding vector for the hand sign present in it. This

embedding vector was then passed into an LSTM to predict the intent behind the sign language. This approach allowed us to use the spatial information present in the sign language images to predict the intent accurately.
Due to limitations in Colab, the model is trained for 5 epochs at a time and saved after each epoch. The train, validation, and test datasets are loaded from pickle files stored in the Google Drive. The dataset was preprocessed to convert the images to grayscale and pad them to a fixed length.

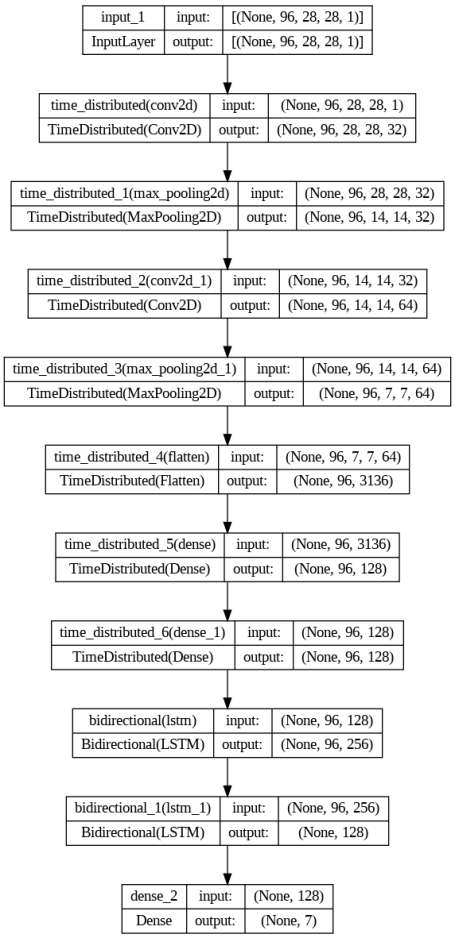

Figure 2: Intent detection model architecture

3.2.3 Method 2 (CNN and LSTM)

In this method initially the CNN that is trained previously in the section 3.1 is used to convert the sequence of images into text. This is the module 1 of the method 2.
Now since we have the ability to generate text from sequence of images all that's left is to train a model that can predict intent if the text is given. Now this is a text classification task. Using the snpis data set LSTM was trained on this task.

3.3 API

Built an API using Flask and HTML to deploy a sign language classification model. The API has two main features: converting given text to sign languages and capturing signs from a video and processing those images to extract the sign and convert those signs into sequence of characters using pretained Sign language classification model.
The first feature is to convert text to sign language. This involves processing the input text, generating the corresponding sign language for each character, and then displaying the resulting sign language sequence to the user as a video.
For the second feature, OpenCV was used to capture video and extract sign language images. then the images was preprocessed to extract only the sign and feed them into a pre-trained Sign language classification model to classify them. the model cannot predict space character. To address this, we have created a separate sign for space and implemented a rule-based approach where we fed that input preprocessed image into ResNet50 model to know if that sign is a space. However, the model cannot predict 'j' and 'z' characters too. To address this, we have implemented a rule-based approach for 'j' and 'z'.
Finally, the API was developed and deployed using Flask and HTML, allowing users to input text or capture signs from a video and receive corresponding sign language output.

3.4 Video Classification

3.4.1 Data Preparation

80 percent of the data was picked for training and used the remaining 20 percent for validation. To increase the dataset size and improve model generalization, techniques like data augmentation techniques such as random cropping, flipping, and rotation to the training frames were applied. Validation data was not augmented.

3.4.2 Feature Extraction

To extract features from the video frames, the pretrained ResNet-18 CNN model was used. The fully connected layer from the model were removed and a new fully connected layer was added with six output neurons. Then all the weights of the layers of the model except for the last two convolutional layers and the new fully connected layer were freezed
.
each frame was passed through the modified

ResNet-18 CNN model to obtain the features for each frame. then the features were averaged across all frames to obtain the feature vector for the entire video.

3.4.3 Model Architecture

This deep learning algorithm for video classification used a rolling prediction averaging technique to improve the stability of the predictions. The algorithm worked as follows:
Loop over all frames in the video file, For each frame, pass the frame through the Resnet model, Obtain the predictions from the model, Maintain a list of the last K predictions, Compute the average of the last K predictions and choose the label with the largest corresponding probability, Label the frame and write the output frame to the disk, a deque was used to implement our rolling prediction averaging. The deque maintained the last K predictions, where K was set to 10 in our implementation.

4 Results

4.1 Sign Language Classifier

A sign language recognition model was created using the ResNet 8 architecture and was trained on a dataset of sign language. The model achieved a notable accuracy rate of 95% when tested, indicating its successful ability to recognize sign language.

The performance of the model was evaluated by utilizing the cross entropy loss function. The training process was then visualized through the creation of Figure 3 and 4, which demonstrate how the accuracy of the model progressed over the course of its training on the dataset. The plots indicate a consistent improvement in accuracy for each epoch, suggesting that the model was able to continuously learn and enhance its performance.

4.2 Intent classification model from

Due to memory constraints on Google Colab, the data for the intent classification model was split into two halves, each consisting of 5000 entries. The model was then saved and loaded every five epochs during the 60-epoch training process for both halves. As the training progressed, the validation accuracy for both halves became nearly constant around 40 epochs, while the test accuracy reached a saturation point.
The accuracy and loss on both train and validation

	precision	recall	f1-score	support
0	1.00	1.00	1.00	331
1	1.00	1.00	1.00	432
2	1.00	1.00	1.00	310
3	1.00	1.00	1.00	245
4	0.96	0.93	0.94	498
5	1.00	1.00	1.00	247
6	0.88	0.90	0.89	348
7	0.86	0.90	0.88	436
8	1.00	1.00	1.00	288
9	1.00	0.94	0.97	331
10	0.84	1.00	0.91	209
11	0.94	0.95	0.95	394
12	0.80	0.85	0.82	291
13	1.00	0.91	0.96	246
14	1.00	1.00	1.00	347
15	1.00	0.99	1.00	164
16	0.75	0.85	0.80	144
17	0.94	0.91	0.93	246
18	0.81	0.66	0.72	248
19	1.00	1.00	1.00	266
20	1.00	0.94	0.97	346
21	0.84	1.00	0.91	206
22	0.93	0.99	0.96	267
23	1.00	0.88	0.94	332
accuracy			0.94	7172
macro avg	0.94	0.94	0.94	7172
weighted avg	0.95	0.94	0.94	7172

Figure 3: Results for all 25 epochs and test data

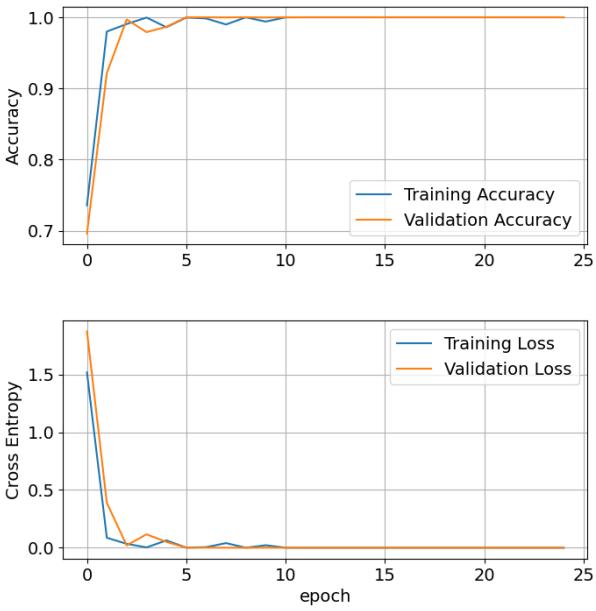

Figure 4: Accuracy, Cross entropy plot

data were plotted over time, revealing that the final train accuracy reached 95%, while the test accuracy peaked at 75%, as demonstrated in figure 5. Despite memory constraints, our approach of dividing the data and frequently saving and loading the model proved to be an effective way of training the intent classification model.

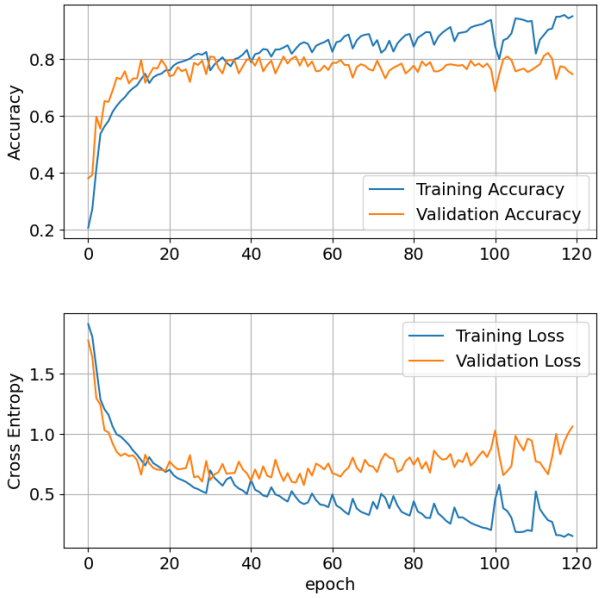

Figure 5: Accuracy, Cross entropy plot of intent classification model on SNIPS dataset (no. of epochs = 60)

4.3 API

4.3.1 Text to sign language video

In this API, input text is processed letter by letter, including spaces. As detailed in our experimental settings, each letter is mapped to a corresponding sequence of images according to a pre-maintained dictionary. For the gestures "j" and "z," a sequence of six images is utilized for each gesture, while white is used for space. Figure 7 illustrates the API after it has been deployed, including an input text box and a "send" button. Once the button is pressed, the text is converted to video and displayed, as depicted in the image.

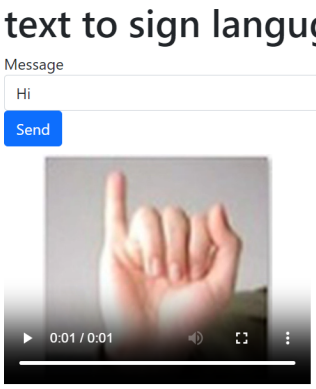

Figure 6: Text to sign

4.3.2 Sign Language to text

As in the figure 7. signs are been captured i.e., as an image which is then processed through a function which detects hand in the image to capture the hand itself. Detection of skin color regions is an image and then contours in the skin color mask, and the largest contour is found after then finally the bounding boxes were predicted using convex hull of the hand by using opencv library.
where the image is preprocessed so that the pre-trained Sign Language classification model can predict, these sequence of images are predicted. before the image is predicted using model it is compared to a predefined sign(which we have made soley for space) using tensorflow resnet50 model.

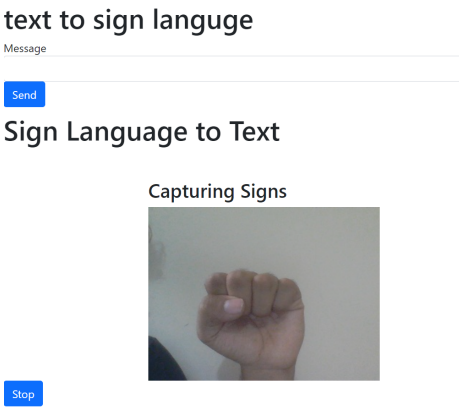

Figure 7: Capturing sings for text prediction

any image which is not space is then predicted, if the prediction probability is low then one of the two letter 'j' and 'z' is chosen randomly or else the predicted alphabet is chosen. Finally the predicted text is then shown in as a form of highlighted text which can be seen in figure 8, once the end button is clicked as shown in figure 7.

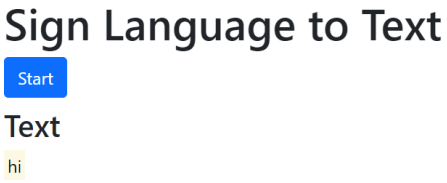

Figure 8: Text prediction after capturing signs

4.4 Video Classification

The sports/activity classification data set was analyzed using the above mentioned deep learning algorithm for video classification, resulting in a prediction accuracy of 93.4 percent.
In this report, we detailed our implementation of a deep learning algorithm for video classification on a sports/activity classification data set. This algorithm yielded a prediction accuracy of 93.4

percent and featured a rolling prediction averaging technique to enhance the prediction's stability.