Gesture Vision

Shubh Himanshubhai Desai, Shubham Jitendrabhai Mendapara, Dhruv Patel, Deepkumar Prajeshbhai Prajapati

desai.shu@northeastern.edu

mendapara.s@northeastern.edu

patel.dhruv5@northeastern.edu

prajapati.de@northeastern.edu

***Abstract*— This project aims to integrate Computer Vision and Natural Language Processing to classify images of ASL hand signs to their respective characters and to generate meaningful sentences from the keywords signed. We employ Convolutional Neural Networks (CNNs) to classify ASL hand signs and we feed the output words to a RNN model to generate the next word prediction. Additionally, we use Fast Autocomplete to generate current word prediction and Text To Speech for converting the generated text to speech. The project attempts to bridge the communication gap between those who are hard of hearing or speaking with the rest of the world by eliminating the need to assign human translators thus making the process of communication faster and more efficient.**

***Keywords*— American Sign Language, Natural Language Processing, Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Computer Vision, OpenCV**

I. Introduction

Communication is a fundamental human right. Although most of us communicate by using spoken language, individuals that are hard of hearing or speaking express or understands through sign language. Sign language can consist of hand gestures, facial expressions, and body movements. American Sign Language (ASL) is a visual language primarily utilized in the United States and Canada [1]. It has its own system and rules, similar to other languages. In order to facilitate communication using ASL, one person must show gestures while the other person interprets those signs. When using ASL, speed and accuracy are of the utmost importance.

The first goal of this project was to enable accurate translation of ASL hand signs into words, thereby enhancing communication. Our dataset consists of ASL signs ranging from A to Z, sign to get ready for next input, sign for current, next word adaptation and sign for backspacing a character, which serves as the foundation of our project. We curated this ASL image dataset and augmented it with pre-existing data. Our next objective was to automatically generate sentences based on these signed words.

We propose a system that involves the utilization of CNNs to implement translation of ASL hand signs and then passes the outputs to the Fast Autocomplete model to give current word suggestion. Then the formed word is passed to RNN for generating next word prediction and when done whole sentence is passed to Text To Speech for speech conversion. The subsequent portions of this report present the related work, our problem statement and methods, experiments and results, discussion and conclusion, and finally future scope.

II. Related work

Our initial source for ASL classification was a GitHub repository titled ‘Sign-Language-To-Text-Conversion’ [2]. Analogous to our project, they worked with a CNN when translating signs into words. This repository was informative in understanding the potential mechanisms that can be employed in translating ASL signs to words. Additionally, many other approaches have been leveraged to convert sign language into words.

Another source we identified converts sign language to English and vice versa [3]. They trained a CNN in the form of the ResNet50 model, used rolling average prediction to recognize their words without jitters in their prediction, and obtained a validation accuracy of 95%. A third paper utilized Single Shot Multi Box Detection (SSD), Inception V3, and Support Vector Machine (SVM) for translating sign language [4]. They claim an accuracy rate of 99.9% for their hybrid model. An additional approach to sign language translation involved using a transformer neural network to analyze both hand movements and facial expressions [5].

Furthermore, we researched about Artificial Intelligence for Sign Language Translation [6] and have also done considerable research on generating sentences from keywords. In 2018, Ziang Xie wrote a paper that serves as a guide on text generation [7]. This paper first mentioned earlier methods of text generation such as rule-based systems and probabilistic models. Next, it reviews encoder-decoder models. Finally, Ziang explains how to find solutions when text generation models do not behave as expected. This guide gives an excellent explanation for how text generation models work and is a good resource.

III. Problem statement and methods

Despite the rapid growth of AI, there hasn't been much progress in Sign Language Communication. Surprisingly, there are fewer projects in this field now, even though the need for

human translators is expected to increase by 18% by 2027. This project aims to develop an AI tool that can generate complete sentences based on keywords signed in ASL from a live video feed. Keywords in this context refer to the current word user might be trying to form or next word user might be willing to form. We divide this project into five submodules: Classification, Current Word Prediction, Next Word Prediction, Text To Speech and Live Video System. The first segment deals with classification of ASL images into alphabets. The second module takes the current characters and provides the most likely word, third module takes this generated word and provides most likely next word, fourth module takes the generated sentence and converts it to speech. Finally, the Live video system attempts to use these modules in real time.

The first component of this System is a Sign Language Classifier that takes input an image of a hand signing an alphabet in ASL and predicts the alphabet. The first step for this component is to develop a suitable dataset. After our preliminary literature survey, we discovered that many analogous projects that attempt similar classifications often use images from one person's hands. This leads to overfitting. To combat this, we developed a new ASL dataset using data available online from various sources along with our own images. Thus, this way we developed a diverse dataset to combat overfitting.

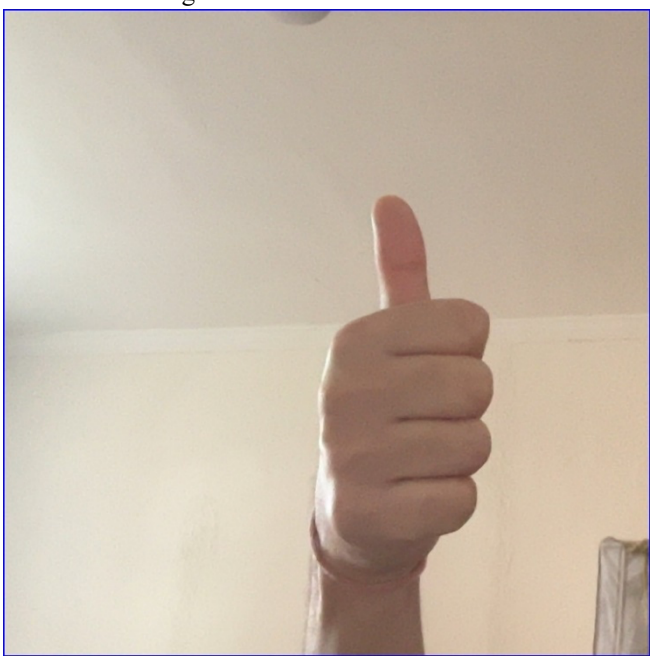

Fig. 1 An example image that we captured for our dataset.

The next step is the classification itself. We identified two methods of classification. The first would be to train a CNN model on these images. This may give us good accuracy however, we were concerned about the potential tradeoffs associated with this approach which include slower training time, slower prediction time (since we need this model to work in real-time) and the model's potential to learn noise. Another approach would be to detect "landmarks" from images of hands and feed these landmarks into our Machine Learning model. Landmarks here refer to x,y and z axis coordinates of points on the hands (fingertips, palms and knuckles).

We chose the latter approach and in our pre-processing, we convert images of hands into landmark arrays using the OpenCV library[8]. They can be 1-dimensioned flattened arrays of landmarks or 2-dimension landmark arrays with data points of type [x,y] depending on the model. The advantage of this method is that now our input data size is just [86][batch_size] or [42,2, batch_size] as opposed to [128,128,1][batch_size] in image based classification. Owing to a smaller input size, we can also attempt to use simpler ML models.

We experimented with RandomForest, 1D Convolutional Neural Networks and 2D Convolutional Networks. For our 2D CNN, we used 2 convolution layers each followed by a maxpool layer. We used the ReLU Activation for the convolutional layers and used Softmax activation for the last Dense Layer. We used the Adam Optimizer coupled with Cross Entropy Loss

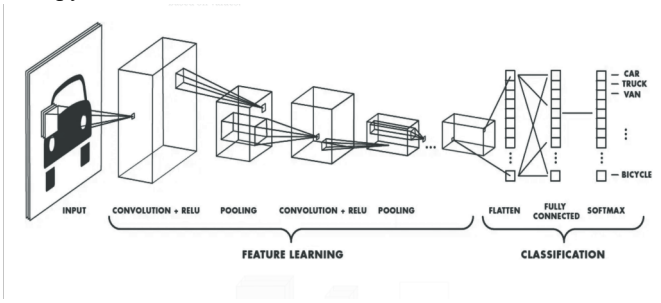

Fig. 2 Architecture of the CNN used. Here the input is a [42,2] 2D array consisting of the X and Y coordinates of the landmarks.

The signed letters are combined into words. Words are separated by breaks in signing. These keywords are sent off to the Current, Next Word Suggestion module for input.

The second component of the Translation System is to take the characters of the current forming word and give suggestion based on it. For this task we used fast autocomplete[9] to provide real-time word suggestions during ASL conversation. Fast Autocomplete is a Python library that offers efficient autocomplete functionality by leveraging trie data structures to efficiently provide autocomplete functionality. A trie, also known as a prefix tree, is a tree-like data structure where each node represents a single character of a word. The root node represents an empty string, and each subsequent node corresponds to a character in the word. The edges of the tree represent characters, and the path from the root to a node spells out a particular word.

TABLE I
Key Features of Fast Autocomplete

Efficient Retrieval	Trie data structures facilitate quick retrieval of word suggestions based on partial inputs. Fast Autocomplete traverses the trie for matches.
Prefix Matching	Fast Autocomplete supports prefix matching, providing

	word suggestions that match the beginning of user input.
Memory Optimization	Storing words in a trie structure optimizes memory usage compared to traditional methods, efficiently representing large vocabularies.
Customization	Fast Autocomplete allows developers to customize parameters like cost thresholds, aligning with specific application requirements.

Third Component is for predicting the next word based on the previous word formed. For this task we used a Recurrent Neural Network (RNN)[10]. The RNN architecture consists of an embedding layer followed by a Long Short-Term Memory (LSTM) layer. The embedding layer maps each word in the input sequence to a dense vector representation, which is then fed into the LSTM layer to capture temporal dependencies and predict the next word in the sequence. The RNN model was trained on a dataset comprising ASL conversation transcripts, with each sequence representing a sequence of words in the conversation. The input sequences were tokenized and padded to ensure uniform length, and the target output was one-hot encoded for categorical prediction. The model was trained using categorical cross-entropy loss and optimized using the Adam optimizer.

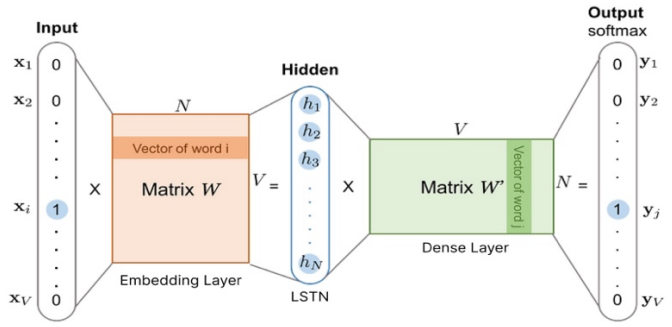

Fig. 3 Architecture of the RNN used.

The Fourth Component is used for converting the formed sentence into speech. For this task we employed gTTs library of python as it allows efficient conversion of text in different languages. It is a cloud-based service that converts text input into high-quality synthesized speech audio. By leveraging gTTs, we enable our ASL conversation application to provide auditory feedback to users, enhancing the accessibility and usability of the system. During runtime, as the user forms complete sentences using ASL signs, the text is synthesized into speech using gTTs. The synthesized speech audio is then played back to the user in real-time, allowing them to hear the words and sentences they have formed using ASL signs. This integration enhances the communication experience for users by providing both visual and auditory feedback.

IV. Experiments and Results

Our dataset comprises approximately a numerous images for each letter in the alphabet. We used OpenCV to get the landmarks of hands which are the X and Y coordinates of fingertips, knuckles and palms. For the RandomForest model, we flattened these landmarks consisting of X and Y coordinates into a 1D array and fed this as the input to the model. For the CNN, we used a 2D array of size (42,2) as input. CNNs gave us the best accuracy amongst all the other methods and we decided to use CNNs in our final model.

TABLE 2
Comparison Of Accuracies

Accuracy		
Model	**Input Size**	**Accuracy**
Random Forest	batch_size * 84	89%
Convolutional Neural Network	batch_size * [42,2]	98.3%

For the CNN model, we used a batch size of 32 and trained the model for 50 epochs. We employed a learning rate of 0.001 and utilized early stopping with a patience of 50 epochs to prevent overfitting. The CNN model was evaluated on a separate test set to assess its performance in classifying ASL alphabet signs. We tried using different learning rate but the best accuracy and least loss was obtained using the mentioned one. We used adam optimizer for it because of its adaptiveness, efficiency in memory usage, rapid convergence speed, and robustness to noisy gradients. The training set is further divided into training and validation dataset for more accurate investigation.

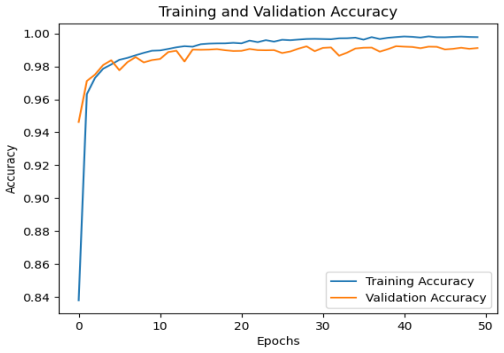

Fig. 4 Training and Validation accuracies for the CNN model

CNNs gave us the best accuracy amongst all the other methods and we decided to use CNNs in our final model. This can further be confirmed from the loss curve we obtained.

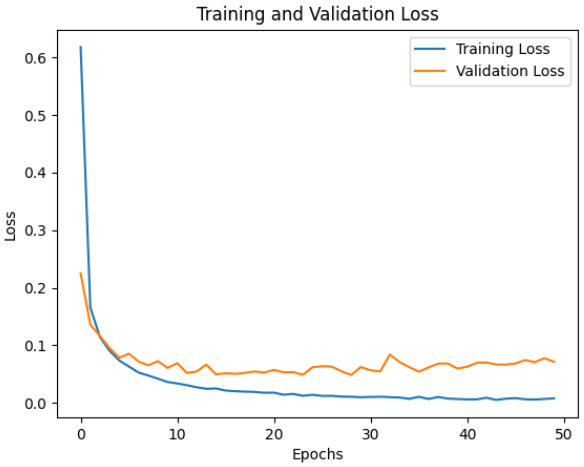

Fig. 5 Training and Validation loss for the CNN model

We can see the training loss converging to minima along with the validation loss curve which is a good sign of the model learning well. Additionally, we got test accuracy of 98% which confirms that the model learnt well.

We have 31 classification classes, 26 for each letter in the alphabet and 5 for other gestures. Thus, it becomes important to further analyse which classes perform well and which ones may benefit from some additional data. We can do it through confusion matrix.

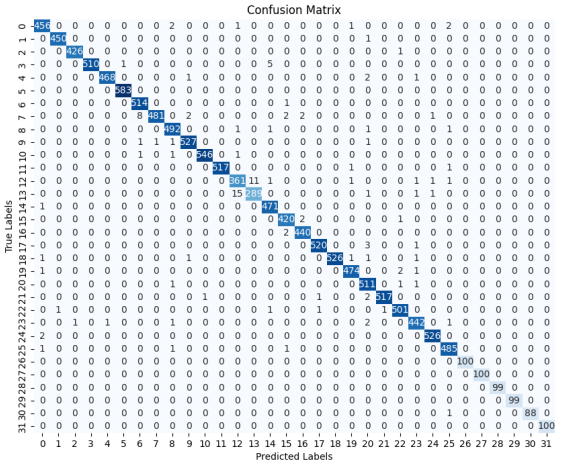

Fig. 6 Confusion Matrix for the CNN model

Here, most of the labels can be seen on the diagonal axis which suggests that the predictions of the label are done correctly by the model. We can also see the numbers on the off diagonal axis, higher numbers in this area suggests us to look at the prediction of those labels as it can be incorrect.

For the Fast Autocomplete system, parameters governing the search algorithm, such as the maximum edit distance and maximum number of suggestions, were adjusted to optimize the trade-off between suggestion accuracy and computational overhead. Furthermore, the pre-processing steps for the word datasets, including tokenization and indexing, were optimized to facilitate fast and efficient word suggestion generation.

In the RNN model, hyperparameters such as the number of LSTM units, embedding dimensions, and sequence lengths were fine-tuned to enhance the model's ability to capture the contextual dependencies and semantic relationships within the input text. Moreover, parameters related to the training process, such as the learning rate, batch size, and number of epochs, were optimized to accelerate convergence and mitigate the risk of overfitting. Accuracy result for this setup is:

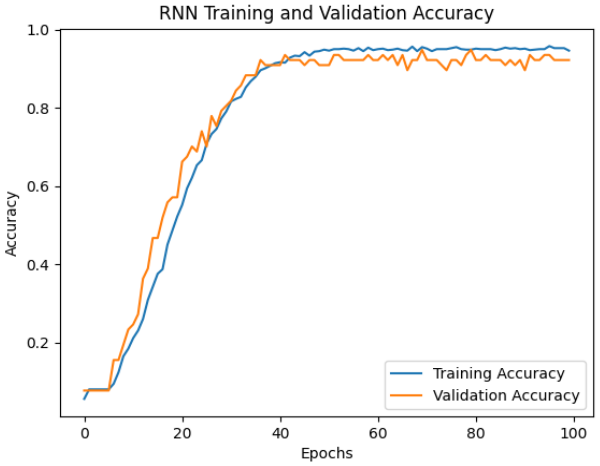

Fig. 7 Training and Validation accuracy for the RNN model

We can see here that the training and validation accuracy rises gradually simultaneously from 0 to 100 epochs hinting towards well learning of model through the dataset.

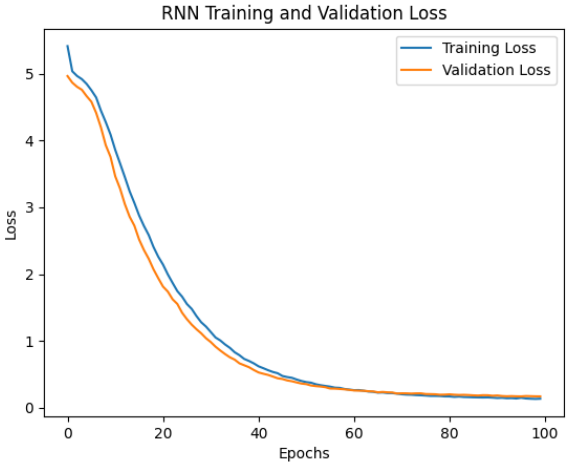

Fig. 8 Training and Validation loss for the RNN model

This loss backs the well learning of model as the training and validation loss decreases gradually over time suggesting that the model is handling the loss correctly. Correctness of both these figures can be confirmed through the test accuracy of 98%, telling us that the model is capturing new data well.

Below is the output we can get from these models:

Input: `go` Current Word Suggestion: `good`

Input: `good` Next Word Suggestion: `morning`

V. Discussion and Conclusion

In this project, we developed an AI system which uses CNNs for classification of ASL images into characters and Fast Autocomplete and RNNs for current and next word prediction with Text To Speech model. One key highlight of this project was the dataset. A lack of readily available diverse ASL hand signs dataset urged us to create our own dataset, combining smaller readily available datasets online. We attempted to detect hand landmarks from images of hands in our pre-processing and feed that into our CNN model. This significantly reduced training time without compromising accuracy. By leveraging computer vision through Convolutional Neural Networks (CNN), we've achieved precise gesture recognition, empowering users to express themselves seamlessly. The integration of Fast Autocomplete offers real-time word suggestions, enhancing the speed and accuracy of message formation. Moreover, Google Text-to-Speech ensures clear and natural audio feedback, facilitating effective communication. Approximately, 4.5 million people in the world suffers from hearing or speaking deficiency experiencing communication as the biggest barrier. Through this approach, we've not only demonstrated the potential of advanced technologies in accessibility but also highlighted areas for future improvement and expansion. With further refinement and exploration, our system holds promise in improving communication accessibility for individuals with diverse needs. The difference between our project and existing ideas is the combination of different models used as toolkit, which can effectively reduce the need of human translator, allowing them to feel connected. Lastly, our project is not just about technology; it is about people. It is about creating connections, fostering understanding, and building a more inclusive world for all. In essence, Gesture Vision represents a holistic solution aimed at improving communication accessibility for individuals with diverse needs, particularly those who rely on ASL as their primary means of expression. Beyond its immediate applications, our project underscores the potential of advanced technologies in fostering inclusivity and enhancing the quality of life for individuals with disabilities.

VI. Future Scope

In future, there are several aspects for future exploration in this project. One of which involves enhancing the accuracy and efficiency of the CNN model for ASL image classification by incorporating more advanced techniques and larger datasets. Additionally, refining the word prediction capabilities of the RNN model and improving the suggestions provided by the Fast Autocomplete feature could enhance the overall user experience. Exploring real-time translation capabilities and expanding the system to support additional sign languages could further broaden its impact and accessibility. Overall, continued innovation and development in these areas hold great promise for advancing communication tools for individuals with diverse needs.

VII. Individual Contributions

Mostly, each of the task is done by contribution of everyone in the group as we have made this project while staying connected physically or virtually.

Shubh Desai:

Researched CNN, other models; contribution to code.

Shubham Mendapara:

CNN and AI use in ASL classification; contributed to code.

Dhruv Patel:

RNN and its implementation; contribution to RNN code.

Deep Prajapati:

Completion of dataset collection, word autocompletion models, fine tuning RNN hyper-parameters.

Together:

Initially, we made outline of this project, combined different methods of project and then created ppt and report.

Link To Github Repository

Link and Instructions: Gesture Vision

References

[1] What is American Sign Language?

[2] Sign Language Translation

[3] Paper by V. D. Avina, M. Amiruzzaman, S. Amiruzzaman, L. B. Ngo, and M. A.

[4] Paper by R. H. Abiyev, M. Arslan, and J. B. Idoko.

[5] ASL using hand movements and facial expressions.

[6] Artificial Intelligence for Sign Language Translation – A Design Science Research Study

[7] Z. Xie, "Neural Text Generation: A Practical Guide," arXiv.org, 2017.

[8] OpenCV Modules

[9] Fast Autocomplete Link

[10] Paper on RNN Model for word prediction.