Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.777632

SIGN LANGUAGE RECOGNITION AND DETECTION: A COMPREHENSIVE SURVEY

Simran Malik, Yukti Kholiwal, Dr. Jayashree J*
School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, India

ABSTRACT
Sign language recognition and detection (SLR) systems are designed in order to interpret and translate sign language gestures into textual or speech form, allowing for improved communication for hard-hearing individuals. The main objective of SLR systems is to provide a feasible and an efficient medium of communication between normal and deaf people by using hand gestures. Webcam or an in-built camera is used in these systems which detects and processes the sign language captured by the camera and the model deployed recognizes it. Recent advances in machine learning and deep learning techniques have greatly improved the accuracy of SLR systems. However, the technology still faces challenges such as environmental factors and limited training data. This paper provides an overview of the current state of SLR systems and analyzes the performance of existing ML technologies in sign language recognition tasks.
Keywords: CNN, deaf/dumb, GRU, LSTM, OpenCV, Mediapipe, Sign language Recognition

1. Introduction

People who have difficulty in hearing or speaking, communicate with others using an important medium known as sign language. Sign language makes them feel included in society also providing them the freedom to express themselves. Sign language allows individuals who are deaf to communicate with others in a way that is natural and efficient for them. It is a complete and independent language, with its own grammar, vocabulary, and syntax. It provides access to information for individuals who are hard of hearing. It allows them to fully participate in education, employment, and other aspects of daily life. Sign language can be used to support early language development for deaf children. It has been shown to have positive effects on cognitive development, including improved memory, problem-solving skills, and overall academic achievement. It allows them to form connections with others, participate in community activities, and advocate for their rights. There have been many research papers published on sign language recognition systems. These systems typically use computer vision techniques to process video input of sign language and convert it to text or speech output. Some key areas of research in sign language recognition include improving the accuracy of gesture recognition, developing sign language recognition systems that work in the real-time, and creating systems that can recognize multiple sign languages. Additionally, researchers have been focusing on creating more user-friendly sign language recognition systems, such as by incorporating machine learning or deep learning techniques. This paper aims to provide an annotated overview on the recent sign language recognition techniques along with their key points, highlight the advantages and disadvantages of various sign language interpreters, analyze and compare the performance of vastly used sign language detection algorithms on our own model application, and provide a comprehensive reference for possible future studies in order to improve the usability of the systems by making it more intuitive and easier to use for both the hearing-impaired individuals and the hearing community.

2. Literature Survey

The paper describes a proposed system for real-time conversion of audio to Indian Sign Language to assist hearing-impaired individuals in communicating with hearing individuals. The system includes components for audio-to-text conversion, tokenization of text, phrase structure trees being parsed into text, reordering the sentences based on grammar rules, lemmatization and part-of-speech tagging, and Indian Sign Language video output. It uses natural language technologies to process input text or audio to generate keywords. The text processing is done using NLTK library which includes tokenization, lemmatization, part-of-speech tagging, parse tree generation, and removal of stop words. [1]

Sign language is a natural and effective mode of communication used by people who have difficulty speaking which is about one percent of the population in India, which makes it important to create a framework to understand Indian Sign Language. The Bag of Visual Words model (BOVW) is used in this paper to recognize Indian sign language alphabets and digits in real-time video streams, and display the predicted alphabets as text and speech. The method involves segmentation based on skin color and background subtraction, extracting features from images using SURF (Speeded Up Robust Features) and generating histograms to map corresponding labels with their signs. The classification is done using machine learning models such as Convolutional Neural Networks (CNN) and Support Vector Machine (SVM). For the easy access of the application, an interactive Graphical User Interface (GUI) has been developed. The results of quantitative analysis results of SVM and CNN are compared. The SVM model achieved an accuracy of 99.14% on the test data, and the precision and recall values calculated for SVM's classification of alphabets and digits show an overall accuracy of 99%. When using CNN, the overall accuracy on the training set was 94% on the last epoch and greater than 99% on the testing set. The model was trained with a categorical cross-entropy loss function and a softmax activation function, resulting in a training loss of 0.1748 on the last epoch and a testing loss of 0.0184. The main objective of this work is to create a real-time recognition utility that can be used anywhere, achieved by creating a custom data set, solving the background dependency problem, and making the system rotation invariant. The model was successfully trained on all 36 ISL static alphabets and digits with an accuracy of 99%. In the future, the dataset could be expanded by adding more signs from different languages, creating a more effective framework for real-time applications. The method can also be extended to recognize simple words and expressions for both continuous and isolated recognition tasks. The key to true real-time applications is improving the response time.[2]

The implemented solution uses a single camera and has specific requirements for its use. The user’s bare hand should be held still within the frame and in camera range, without any obstructions. Additionally, the system is only intended to be used indoors as the selected camera does not perform well in sunlight. OpenCV library is used to produce dataset. The project uses a two-layer algorithm approach for predicting the hand gesture. A gaussian blur filter and threshold are applied by the first algorithm layer to the OpenCV frame to extract features and then passes the processed image to a CNN model for prediction. The second algorithm layer detects sets of similar symbols and uses classifiers specific to each set to classify between them. The project faced several challenges, including difficulty in finding a suitable dataset and selecting an appropriate filter for image processing. The team ultimately decided to create their own dataset and found that using a gaussian blur filter on images resulted in the best features for input into a CNN model in Keras. They also faced issues with the accuracy of the model, which was enhanced by expanding the size of the input image and thereby improving the dataset. [3]

The paper discusses the use of sign languages as a means of communication for specially abled individuals, and the limitation that several people do not know the basics of sign languages. It suggests converting sign language gestures into spoken language using automated Sign Language Detection models. The system described in the paper is capable of recognizing sign language in real-time with an average success rate of 85.45%. It was trained on the Indian Sign Language alphabet dataset using TensorFlow object detection API. The data acquisition was done using a webcam, Python and OpenCV, which made it cheaper. The article states that the dataset used is small and limited and future work would be to increase the dataset size and learn more sign language symbols. Furthermore, the system may be constructed for many sign languages by altering the dataset and the current TensorFlow model employed. [4]

The suggested approach works with any built-in or webcam that can recognize signs by detecting them and processing them. The findings suggest that the suggested technology can provide precise results in conditions of regulated light and intensity. In addition, adding new motions is simple, and the model will be more exact if there are more photographs captured at various angles and frames. As a result, expanding the dataset makes it simple to scale up the model. However, the model has few drawbacks as well, such as environmental conditions that reduce the accuracy of the detection, such as low light intensity and unmanaged backdrop, which need to be further worked upon. [5]

This paper discusses the use of machine learning and deep learning techniques for sign language recognition. The paper reviews various SLR systems and the methods used in them, including feature extraction, classification, and post-processing. It also covers the challenges faced in SLR such as variability in sign language gestures, lighting conditions and background noise. The paper also highlights the recent advances in SLR using deep learning techniques and the potential of these techniques to improve the performance of SLR systems. The authors conclude by pointing out the need for more research in the field, particularly in developing large datasets, improving feature extraction and classification techniques, and addressing the issue of variability in sign language gestures. [6]

The survey paper discusses and compares all the state-of-the-art technologies utilized for sign language communication between deaf and speaking people. The stages of sign language application studies are sign language capturing, sign language recognition, translation and representation. The authors also present the merits and demerits of contemporary sign language technology and potential avenues for AI growth in the future to facilitate in bridging the communication gap for a more inclusive society. The two sign language recognition techniques discussed are “Continuous sign language recognition” and “Isolated sign language recognition”. Continuous sign language recognition results in high accuracy when used with bigger datasets of more than 1000 words, for ex Phoenix-2014 and Phoenix-2014-T. Continuous Sign Language Recognition is a complex process because it is necessary to be done without the information regarding starting and end of a gloss (unit of a sign language sentence like a phrase). Its accuracy is measured using WER (word error rate). Out of all the algorithms surveyed, CSLR methods that use 2D-CNN achieve high performance. The ISLR datasets discussed CSL-500, MS-ASL, WASL, AUTSL, LSA64, and IsoGD. These datasets have varying characteristics, such as the number of glosses, signers, and background settings, and some cover multiple sign languages. Even though IsoGD is a gesture recognition dataset, it is a good fit for ISLR due to its vastness and tricky conditions. Datasets with high resolution input perform better. The article explores various techniques for improving the accuracy and robustness of sign language recognition (SLR) methods. One such method involves utilizing multi-stream networks that extract various types of features from visual cues, such as optical flow and skeletal joints, to overcome potential confusion associated with using only one type of feature. Researchers have proposed different multi-stream SLR methods, such as using RGB and optical flow data, body and hand skeletal characteristics, hand and mouth areas, hand image regions, hand heatmaps, and 2D projections of hand skeletal joints. To effectively categorize sign language, these characteristics are processed using 3D-CNN and LSTM networks, with late fusion carried out at the score level. Some methods also incorporate textual sign representations that are extracted based on definitions of how the signs are gestured. These proposed methods have achieved high accuracy in various datasets. The essay argues that while ISLR approaches may identify and categorize isolated indications on pre-segmented films, they are not useful for real-world applications despite having excellent accuracy on large-scale datasets. Although CSLR approaches have high hardware and training requirements, they offer a lot of room for development in the future. [7]

Recognizing continuous sign language (CSLR) poses a challenge due to the absence of accurate temporal annotation of sign language data. Due to their restricted temporal receptive field, the majority of current approaches that depend on a hybrid model of "CNN + RNN" for feature extraction struggle to extract precise temporal data for each sign language phrase. To address this issue, the authors propose a multi-scale temporal network (MSTNet) which includes a Resnet and two fully connected layers for frame-wise feature extraction, a temporal feature extraction part for learning temporal features using a multi-scale temporal block (MST-block) and transformers, and a multi-level Connectionist Temporal Classification (CTC) loss for training. Their method demonstrated improved accuracy (21 WER) on two publicly available datasets: RWTH and CSL, indicating its effectiveness in CSLR feature extraction. MSTNet for CSLR improves the accuracy of temporal feature extraction using a multi-scale temporal receptive field approach. The proposed model also uses a multi-level CTC loss to more effectively train the CNN by learning and updating the shallow network parameters. This enhances recognition performance. In order to improve real-time performance while lowering redundancy and memory usage, the paper suggests a future research direction of maintaining a constant recognition rate after down sampling the temporal dimension of input sign language videos as it is currently a limitation. [8]

This paper presents an approach that uses the particle swarm optimization (PSO) algorithm to build CNN models. Finding optimal network architecture of CNNs is a complex problem due to the large search space of hyper-parameters. The authors propose two optimization approaches using PSO to determine the right criteria for CNNs, including the number of convolutional layers, filter size, number of filters, and batch size. The optimized architectures are tested on sign language databases and achieved a recognition rate of over 99%, which is competitive with other state-of-the-art approaches. The 2 presented approaches use the PSO algorithm to optimize CNN architectures for sign language detection. The optimization's focus is the number of convolutional layers, filter size, number of filters, and batch size. The proposed approaches achieved recognition rates of over 99% in three different sign language databases, which is competitive with state-of-the-art approaches. The authors suggest future work to enhance other CNN hyper-parameters, explore different evolutionary computational techniques, and implement real-time or video input images. The goal is to contribute to the creation of aided communication technologies and support machine interaction for the deaf community.[9]

The research suggests a deep learning method which incorporates GRU and LSTM with attention mechanisms to translate sign language into natural language and vice versa. The ASLG-PC12 and Phoenix-2014T sign language corpora were used to test the suggested models, which performed better than earlier studies. For both text to gloss and gloss to text translation on the ASLG-PC12 corpus as well as for various translation directions on the Phoenix-2014T corpus, the best results were obtained using the GRU model with Bahdanau attention. The author suggests a deep learning method for encoder-decoder-based two-way translation of sign language to normal language with major focus on Bahndanau and Luong mechanisms. Two encoder-decoder network architectures using GRU and LSTM are used. Both the ASLG-PC12 and Phoenix-2014T corpora are used to test the suggested methodology. The results of the experiment show that the suggested strategy outperforms comparable work on the same corpus, and the top model translated from text to gloss with a ROUGE score of 94.37% and a BLEU-4 score of 83.98%. Future study should look at how text to pose estimation and vice versa may be translated. The authors suggest that future research could explore the use of pose estimation for sign language translation.[10]

The authors discuss sign language as a structured form of hand/arm gesture and its importance in building an inclusive society by bridging the evident communication gap between hearing and hearing impaired people. It highlights the usage of visual indicators, facial expressions, and body postures in sign language, and the role of pattern matching, computer vision, and natural language processing in sign language identification. The text also reviews current methodologies and algorithms used in sign language identification research and discusses the challenges and limitations of gesture identification research. The three methods of sign language identification analyzed are HMM, CNN, PNN. Based on the findings, it can be inferred that the CNN approach with security measures yields better Accuracy results of 80%, whereas the HMM and PNN approaches only yield 60% and 75% Accuracy respectively. The CNN method generates more precise outcomes with a rate of 82%, in contrast to the HMM and PNN techniques which generate 70% and 75% precision respectively. The CNN technique generates higher recall outcomes with a rate of 85%, whereas the HMM and PNN techniques yield only 75% and 80% recall respectively. In order to create a two-tier corpus for various SL dialects with the involvement of the deaf community, it has been determined that upcoming research in this field must incorporate the deaf community.[11]

Recognizing sign language from sequences of monocular images is difficult because it is not easy to infer 3D information from 2D data. We can not use traditional classification techniques because there is a huge variety of new signs which are getting produced constantly in production environments. To address these challenges, this paper proposes a new model which is called the Contrastive Transformer-based model. This model can enable better comparison of vector embedding and can learn representations from body key point sequences. This approach allows us to perform translation and classification on various tasks. The experiments demonstrated on this paper shows that models can generalize well and achieve competitive results for sign classes. Recent advances in machine learning have led to the use of deep learning models for sign language translation. This paper proposes a model to extract key-points of the body and hands for each frame. It also proposes a transformer to encode the sequential key-points of the input video. The Transformer passes the entire input sequence in parallel, whereas recurrent units pass these input sequences one step at a time. To enhance performance in long sequences, the transformer also utilizes a Self-Attention mechanism. The model is trained using triplet loss in a contrastive learning approach. This enables the model to semantically map sequences of key-points into vectors from an embedding vector space. One way to classify embedding vectors is by measuring the distance between them using methods such as nearest neighbor, cosine similarity, and euclidean distance. This method enables the model to compare new inputs to known signs and classify them without retraining. The paper proposes a new approach to classify isolated signals in sign language using a Transformer encoder trained in contrastive learning, which only requires a small number of examples. The paper's main contribution is this novel classification method. The proposed pipeline in this work involves several stages that go from the input video to the predicted output class. Initially, a pose and hands model is used to extract key-points from the RGB images, resulting in a representation of the person in the form of a skeleton of the person who is performing the sign. The skeletons obtained are then passed into the Transformer encoder to predict the sequence embedding. Finally, a classification model is used to predict the class of the input sign. The Transformer encoder model, trained using contrastive learning, is effective in generating valuable representations of sequences of body key points while preserving temporal information. The model was tested for classification using cosine similarity, KNN and prototypical networks, and they were near to 90% accurate. The results indicate that a large number of classes is beneficial to the model, and the mislabels were mainly due to similar signs being grouped together in the embedding space. Future work may involve constructing datasets with more classes, including different languages, and analyzing the Transformer's expressiveness and interpretability by examining its internal attention weights. [12]

3. System Model

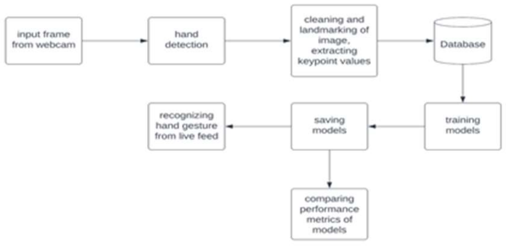

input frame from webcam
hand detection
cleaning and landmarking of image, extracting keypoint values
Database
training models
saving models
recognizing hand gesture from live feed
comparing performance metrics of models

Figure 1. System/ Architecture diagram for our project

4. Methodology Adapted

1. Data collection

To avoid any bias present in any of the existing sign language datasets, we decided to make our own dataset in different light conditions using the webcam. Using OpenCV, we collected the input frames for different ASL letters and saved it into a folder, by mapping the hand gesture to keyboard input.

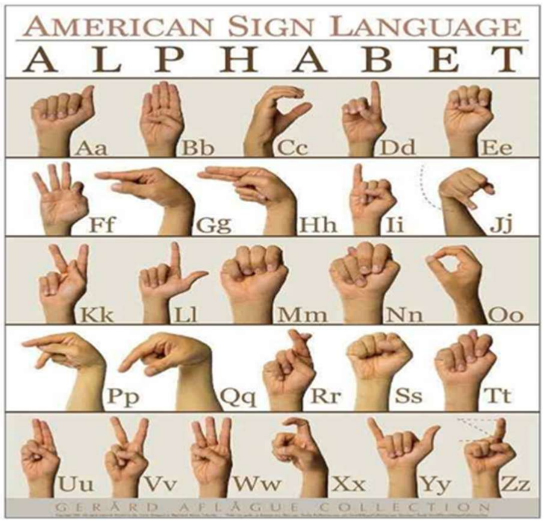


Figure 2. ASL Alphabet [13]

OpenCV

It is an open-source software that is frequently deployed in computer vision and machine learning. It was created to provide a standard architecture for computer vision applications and to expedite the incorporation of machine learning into finished goods. It largely focuses on video collection and analysis, as well as picture processing. Face and object detection are among its prominent features. It makes use of NumPy, a MATLAB-like syntax that is extremely effective. All OpenCV array formats may be converted into and out of NumPy arrays. Additionally, this makes it easier to communicate with other NumPy-using libraries like SciPy and Matplotlib.

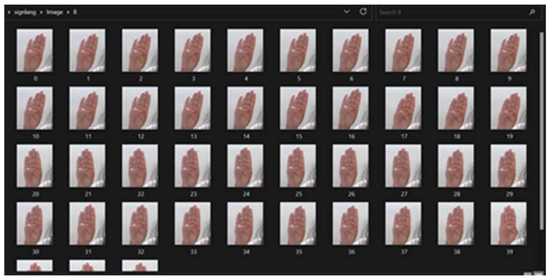


Figure 3.1 Sign Language Dataset for the alphabet 'B'

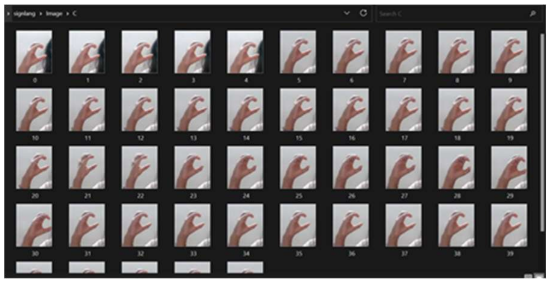


Figure 3.2 Sign Language Dataset for the alphabet 'C'

2. Using Mediapipe library to landmark the sign images, and extract key points.

Mediapipe is Google's open-source, cross platform library that offers pre-built ML solutions for computer vision tasks. It provides framework for developing ML pipelines for time-series input (audio, video, etc.). It also simplifies the integration of computer vision software into applications and demos running on separate hardware platforms. In this project, Mediapipe has been used for Hand landmark detection. Still images, decoded video frames and live video frames are acceptable forms of input. Landmarks of detected hands in image and world coordinates are given as output. Hand landmark model bundle is downloaded and stored., which has key point localization of 21 hand-knuckle coordinates.

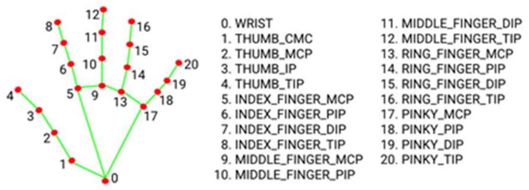
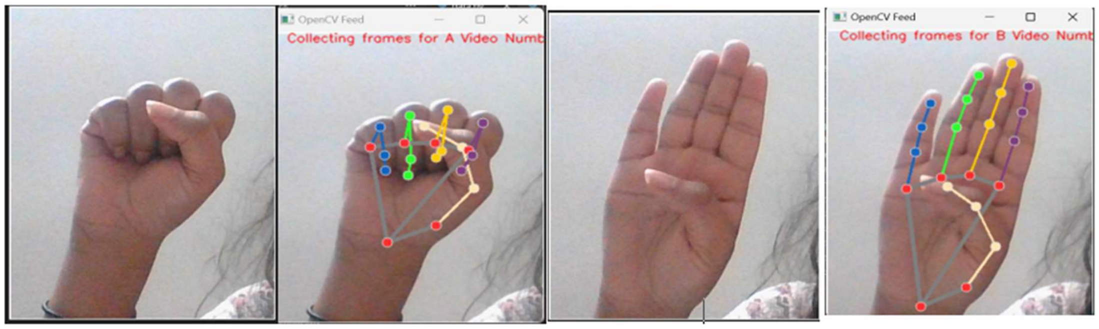


Figure 4. Hand landmark detection model [14]


OpenCV Feed
Collecting frames for A Video Numb
OpenCV Feed
Collecting frames for B Video Numb
Figure 5. Still image (left) converted to landmarked image (right)

3. Training machine learning models to identify sign language gestures by learning the extracted key points of various gestures.

Models deployed:

CNN:

Convolutional Neural Network (CNN) is deep learning architecture that learns directly from the input data. They yield high performance in categorizing time-series data, image classification and object detection tasks. It functions by distinguishing the objects of the input image and assigning weights to them. A CNN uses filters of different resolutions on each training picture, and the output from each filter is used as input for the next layer. The CNN consists of input and output layers, as well as hidden layers that analyze the data to identify its unique features. The main layers include convolution, activation (or ReLU), and pooling. Convolution applies a sequence of filters to the input images, highlighting different features of the images with each filter. The Rectified Linear Unit (ReLU) helps to speed up and make training more efficient by keeping only positive values and converting negative values to zero. This is also known as the activation layer because it only passes on active features to the next layer. Pooling is a process that reduces the number of parameters the network has to learn by conducting non-linear downsampling on the output. By repeating these procedures across multiple levels, each layer can learn to recognize different characteristics.

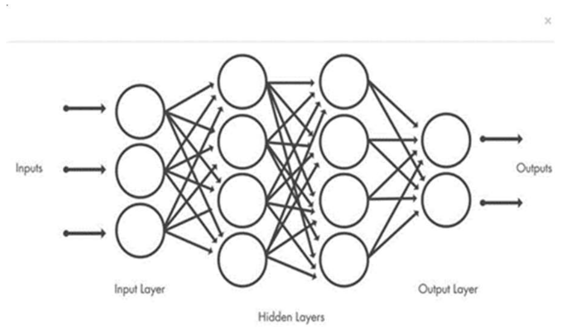


Figure 6. CNN organized in layers of interconnected nodes [15]

Shared Biases and Weights

A CNN has shared weights and bias values, which are the same for all hidden neurons in a particular layer, unlike a conventional neural network.

The same feature, such an edge or a blob, is being detected by all hidden neurons in various parts of the image, according to this. This increases the network's tolerance for object translation in an image.

Classification Layers

The design of a CNN changes to categorization after learning information across several layers.

Why CNN?

- Lowers the high dimensionality of input images without information loss with its built-in convolution layer functionality.

- Eliminates the need for human supervision for identification of key features.
- Minimal computation in comparison to other networks, with weight sharing as an added advantage.

CNN is a go-to technology for image classification and object detection applications.

LSTM

Long Short-Term Memory (LSTM) network is a variant of recurrent neural network (RNN). Due to its ability to understand long-term connections between data time steps, LSTM is frequently used to learn, analyze, and categorize sequential data. The problem of long-term dependence in recurrent neural networks (RNNs) was specifically tackled by the creation of LSTM networks, which solved the vanishing gradient problem. Unlike traditional feedforward neural networks, LSTMs have feedback connections that enable them to process entire sequences of data (like time series) as a whole, without treating each data point individually. This means that they can utilize past information from the sequence to handle new data points. LSTMs are therefore very adept at processing data sequences.

Three factors affect an LSTM's output at a specific moment in time:

- The input information for the present time step
- The prior concealed state, or the output at the previous instant
- The network's present long-term memory is referred to as the cell state

LSTM has three gates which are an output gate, an input gate, and a forget gate. These gates regulate the data entry in the sequence, its storage and its departure from the network.

The forget gate comes first. The prior concealed state and the current data point in the sequence determines which parts of LSTM should be forgotten (have less weight).

The second stage includes input gate and new memory network. The aim of this stage is to determine the additional data that needs to be included in the network's long-term memory (cell state), considering the previous hidden state and the newly received input information.

The third stage involves the output gate. This gate selects the new concealed state. Three factors involved in this determination are the recently updated cell state, the prior hidden state, and the fresh input data.

These steps are repeated several times. The model iterates several times to make a prediction. This is prediction is also a hidden state. Linear layer step is the final step that converts hidden state to the desired output.

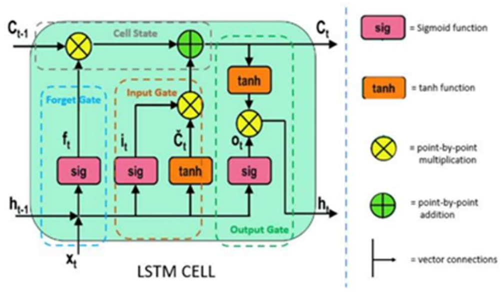


Figure 7. Diagram of LSTM Cell [16]

Why LSTM?

● It excels at managing long-term dependencies as a result of its long-term memory retention characteristic.

● the vanishing gradient issue is significantly less likely to affect LSTMs. This is due to the fact that they employ an LSTM cell, a different type of activation function that aids in information retention across lengthy periods.

● It is capable of processing the complete sequence of data, without expanding the size of the network, due to its feedback connections.

GRU

Gated recurrent Unit or GRU is part of the RNN family. The Vanishing gradient issue is addressed by this algorithm, like LSTM. It is in fact, a simpler variant of LSTM: it uses only 2 gates which are update gate and reset gate.

The update gate is the long term memory of the network. It decides what and how much past information should be propagated to the future.

The reset gate is responsible for determining what past information should be forgotten- hence is accountable for the short term memory of the network.

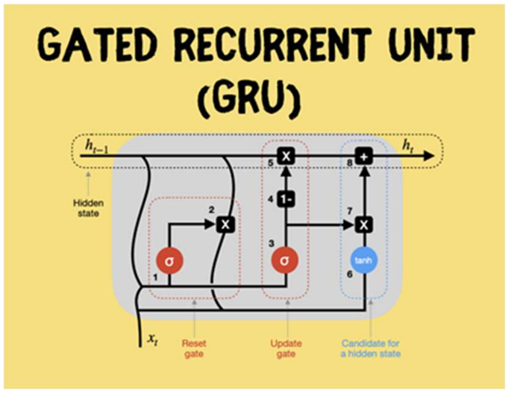


Figure 8. GRU Model [17]

Why GRU?

● It requires less memory

● Fast to train

● Overcomes Vanishing gradient problem

● Simpler architecture than LSTM, offers quick execution

These three models are trained using the pipelined images produced by media pipe, after performing the train test split.

4. Performing prediction from live feed

To compare the accuracy of the models in real time, sign language gesture is detected and captured using OpenCV, and landmarked using mediapipe. The models compare the extracted key point arguments from life feed and compare it with their past learnings and make a prediction of the sign, which is printed on the output screen, along with the accuracy.

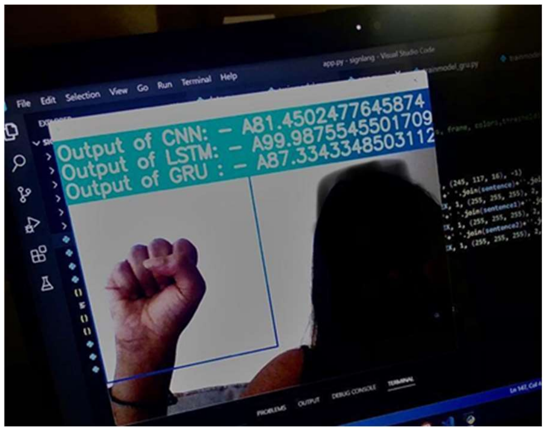

Output of CNN: – A81.4502477645874
Output of LSTM: – A99.98755455501709
Output of GRU : – A87.33433348503112

Figure 9.1 Sign language detection and accuracy out of CNN, LSTM and GRU models for Letter ‘A’

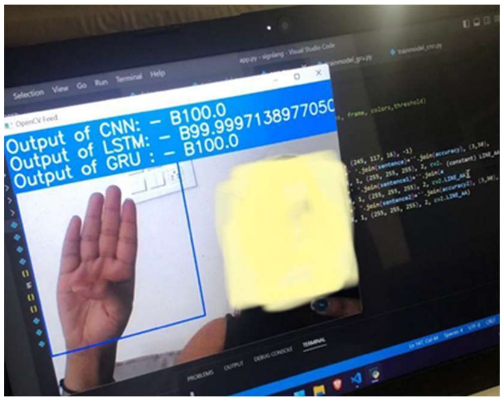
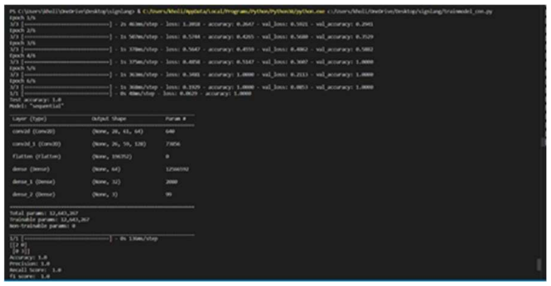


Figure 9.2 Sign language detection and accuracy out of CNN, LSTM and GRU models for Letter ‘B’

5. Results and Discussion

The three models are compared based on their performance metrics: Precision, Recall, Accuracy and F1 score.



```
PS C:\Users\Dholl\OneDrive\Desktop\signlang> & C:/Users/Dholl/AppData/Local/Programs/Python/Python38/python.exe c:/Users/Dholl/OneDrive/Desktop/signlang/trainmodel_cnn.py
Epoch 1/6
3/3 [==============================] - 2s 463ms/step - loss: 1.2058 - accuracy: 0.2647 - val_loss: 0.5921 - val_accuracy: 0.2941
Epoch 2/6
3/3 [==============================] - 1s 507ms/step - loss: 0.5784 - accuracy: 0.4265 - val_loss: 0.5680 - val_accuracy: 0.3529
Epoch 3/6
3/3 [==============================] - 1s 370ms/step - loss: 0.5647 - accuracy: 0.4559 - val_loss: 0.4862 - val_accuracy: 0.5882
Epoch 4/6
3/3 [==============================] - 1s 375ms/step - loss: 0.4858 - accuracy: 0.5147 - val_loss: 0.3687 - val_accuracy: 1.0000
Epoch 5/6
3/3 [==============================] - 1s 363ms/step - loss: 0.3481 - accuracy: 1.0000 - val_loss: 0.2113 - val_accuracy: 1.0000
Epoch 6/6
3/3 [==============================] - 1s 368ms/step - loss: 0.1929 - accuracy: 1.0000 - val_loss: 0.0853 - val_accuracy: 1.0000
1/1 [==============================] - 0s 40ms/step - loss: 0.0629 - accuracy: 1.0000
Test accuracy: 1.0
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 61, 64)        640

 conv2d_1 (Conv2D)           (None, 26, 59, 128)       73856

 flatten (Flatten)           (None, 196352)            0

 dense (Dense)               (None, 64)                12566592

 dense_1 (Dense)             (None, 32)                2080

 dense_2 (Dense)             (None, 3)                 99

=================================================================
Total params: 12,643,267
Trainable params: 12,643,267
Non-trainable params: 0
_________________________________________________________________
1/1 [==============================] - 0s 136ms/step
[[2 0]
 [0 1]]
Accuracy: 1.0
Precision: 1.0
Recall Score:  1.0
F1 score:  1.0
```

Figure 10. Performance Metrics for CNN model

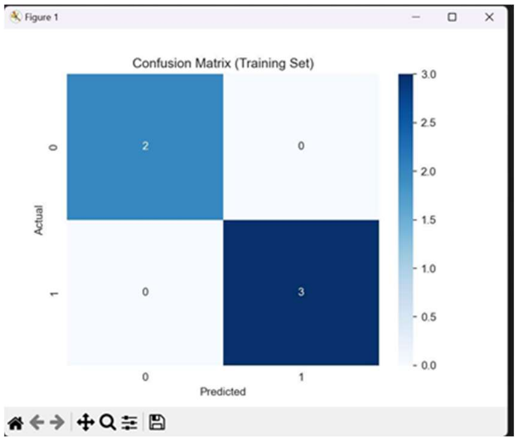
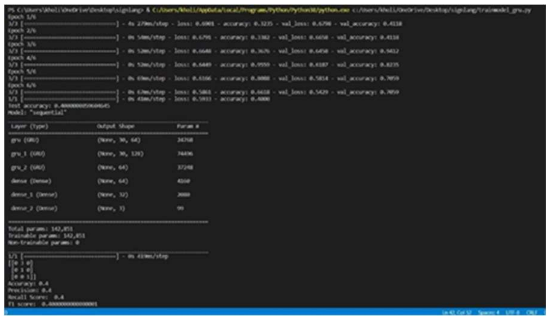


Figure 11. Confusion Matrix for CNN model



```
PS C:\Users\kholi\OneDrive\Desktop\signlang> & C:/Users/kholi/AppData/Local/Programs/Python/Python38/python.exe c:/Users/kholi/OneDrive/Desktop/signlang/trainmodel_gru.py
Epoch 1/6
3/3 [==============================] - 4s 279ms/step - loss: 0.6981 - accuracy: 0.3235 - val_loss: 0.6798 - val_accuracy: 0.4118
Epoch 2/6
3/3 [==============================] - 0s 54ms/step - loss: 0.6791 - accuracy: 0.3382 - val_loss: 0.6658 - val_accuracy: 0.4118
Epoch 3/6
3/3 [==============================] - 0s 52ms/step - loss: 0.6640 - accuracy: 0.3676 - val_loss: 0.6450 - val_accuracy: 0.5412
Epoch 4/6
3/3 [==============================] - 0s 52ms/step - loss: 0.6449 - accuracy: 0.5559 - val_loss: 0.6187 - val_accuracy: 0.8235
Epoch 5/6
3/3 [==============================] - 0s 69ms/step - loss: 0.6166 - accuracy: 0.8088 - val_loss: 0.5814 - val_accuracy: 0.7059
Epoch 6/6
3/3 [==============================] - 0s 67ms/step - loss: 0.5861 - accuracy: 0.6618 - val_loss: 0.5429 - val_accuracy: 0.7059
1/1 [==============================] - 0s 41ms/step - loss: 0.5933 - accuracy: 0.4000
Test accuracy: 0.4000000059604645
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru (GRU)                   (None, 30, 64)            24768

 gru_1 (GRU)                 (None, 30, 128)           74496

 gru_2 (GRU)                 (None, 64)                37248

 dense (Dense)               (None, 64)                4160

 dense_1 (Dense)             (None, 32)                2080

 dense_2 (Dense)             (None, 3)                 99

=================================================================
Total params: 142,851
Trainable params: 142,851
Non-trainable params: 0
_________________________________________________________________
1/1 [==============================] - 0s 419ms/step
[[0 3 0]
 [0 1 0]
 [0 0 1]]
Accuracy: 0.4
Precision: 0.4
Recall Score:  0.4
f1 score:  0.4000000000000001
```

Figure 12. Performance Metrics for GRU model

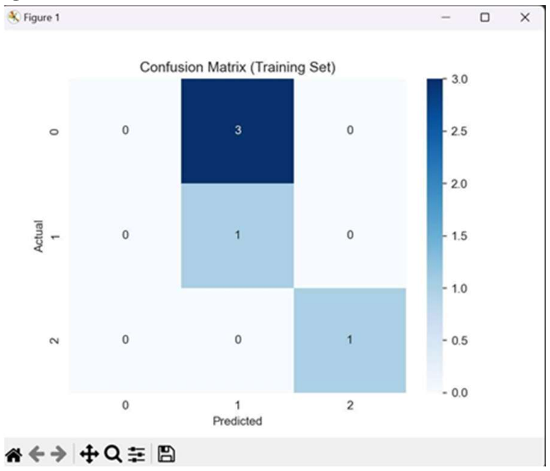
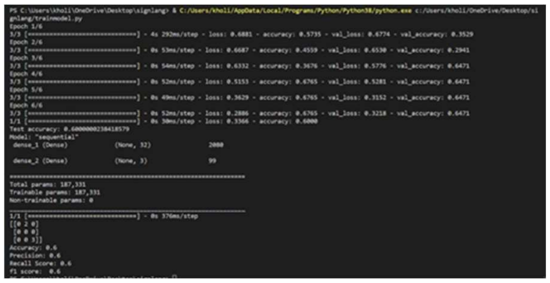


Figure 13. Confusion Matrix for GRU model

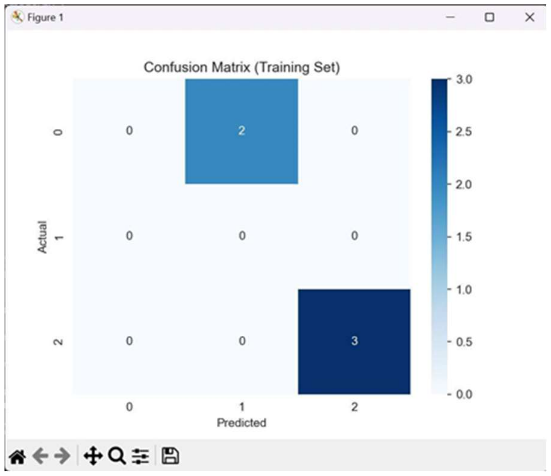


```
PS C:\Users\kholi\OneDrive\Desktop\signlang> & C:/Users/kholi/AppData/Local/Programs/Python/Python38/python.exe c:/Users/kholi/OneDrive/Desktop/si
gnlang/trainmodel.py
Epoch 1/6
3/3 [==============================] - 4s 292ms/step - loss: 0.6881 - accuracy: 0.5735 - val_loss: 0.6774 - val_accuracy: 0.3529
Epoch 2/6
3/3 [==============================] - 0s 53ms/step - loss: 0.6687 - accuracy: 0.4559 - val_loss: 0.6530 - val_accuracy: 0.2941
Epoch 3/6
3/3 [==============================] - 0s 54ms/step - loss: 0.6332 - accuracy: 0.3676 - val_loss: 0.5776 - val_accuracy: 0.6471
Epoch 4/6
3/3 [==============================] - 0s 52ms/step - loss: 0.5153 - accuracy: 0.6765 - val_loss: 0.5281 - val_accuracy: 0.6471
Epoch 5/6
3/3 [==============================] - 0s 49ms/step - loss: 0.3629 - accuracy: 0.6765 - val_loss: 0.3152 - val_accuracy: 0.6471
Epoch 6/6
3/3 [==============================] - 0s 52ms/step - loss: 0.2886 - accuracy: 0.6765 - val_loss: 0.3218 - val_accuracy: 0.6471
1/1 [==============================] - 0s 30ms/step - loss: 0.3366 - accuracy: 0.6000
Test accuracy: 0.6000000238418579
Model: "sequential"
 dense_1 (Dense)             (None, 32)                2080

 dense_2 (Dense)             (None, 3)                 99

=================================================================
Total params: 187,331
Trainable params: 187,331
Non-trainable params: 0
_________________________________________________________________
1/1 [==============================] - 0s 376ms/step
[[0 2 0]
 [0 0 0]
 [0 0 3]]
Accuracy: 0.6
Precision: 0.6
Recall Score: 0.6
f1 score:  0.6
```

Figure 14. Performance Metrics for LSTM model



Actual \ Predicted	0	1	2
0	0	2	0
1	0	0	0
2	0	0	3

Figure 15. Confusion Matrix for LSTM model

Precision

Precision tells the ratio of true positive predictions to the total positive predictions. It reflects the quality of the predictions made.

Accuracy

It is referred to as the ratio of the number of correct predictions to the total number of predictions made.

Recall

It is the percentage of data examples that the model categorizes correctly from all the samples of that category.

F1 Score

This score is used for evaluating the performance of a binary classification model. It is the harmonic mean of precision and recall.

Confusion Matrix

The confusion matrix is a matrix which evaluates the efficiency of a classification model. Count values describe the number of inaccurate and accurate predictions for every category.

6. Conclusion

From the above performance metrics, we infer that CNN model has proven to be most efficient on our application and dataset to detect and recognize sign language. The survey describes the existing techniques to recognize and detect sign language. Each model or approach has its own pros and cons. Our aim was to study and compare the performance metrics of existing models on our application. Our future aim will be to apply and compare more models and algorithms to achieve higher accuracies. Anyone who wants to do extensive research on sign language and overcome the shortcomings of existing models, can use this paper to get in depth knowledge about various models and approaches. Effort from researchers in this area will build an inclusive society in future, by bridging the communication gap.

7. References

[1] Sharma, P., Tulsian, D., Verma, C., Sharma, P., & Nancy, N. (2022). Translating Speech to Indian Sign Language Using Natural Language Processing. Future Internet, 14(9), 253. https://doi.org/10.3390/fi14090253

[2] Katoch, Shagun & Singh, Varsha & Tiwary, Uma Shanker. (2022). Indian Sign Language recognition system using SURF with SVM and CNN. Array. 14. 100141. 10.1016/j.array.2022.100141.

[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] Srivastava, S., Gangwar, A., Mishra, R., & Singh, S. (2022). Sign Language Recognition System using TensorFlow Object Detection API. ArXiv, abs/2201.01486.

[5] Pathak, Aman & Kumar|priyam|priyanshu, Avinash & Chugh, Gupta|gunjan & Ijmtst, Editor. (2022). Real Time Sign Language Detection. International Journal for Modern Trends in Science and Technology. 8. 32-37. 10.46501/IJMTST0801006.

[6] Das, S., Biswas, S. kr., Chakraborty, M., & Purkayastha, B. (2021). A Review on Sign Language

[7] Papastratis, I., Chatzikonstantinou, C., Konstantinidis, D., Dimitropoulos, K., & Daras, P. (2021). Artificial Intelligence Technologies for Sign Language. Sensors (Basel, Switzerland), 21(17), 5843. https://doi.org/10.3390/s21175843

[8] Zhu, Q., Li, J., Yuan, F., & Gan, Q. (2022). Multi-scale temporal network for continuous sign language recognition. ArXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2204.03864

[9] Fregoso, J., González, C., & Martinez, G. (06 2021). Optimization of Convolutional Neural Networks Architectures Using PSO for Sign Language Recognition. Axioms, 10, 139. doi:10.3390/axioms10030139

[10] Amin, M., Hefny, H., & Mohammed, A. (2021). Sign Language Gloss Translation using Deep Learning Models. International Journal of Advanced Computer Science and Applications, 12(11). https://doi.org/10.14569/ijacsa.2021.0121178

[11] M. Prema, & P.M. Gomathi. (2022, January). Analysis On Different Methods For Sign Language Identification. Ictact Journal.

[12] Ferreira, S., Costa, E., Dahia, M., & Rocha, J. (2022). A Transformer-Based Contrastive Learning Approach for Few-Shot Sign Language Recognition. ArXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2204.02803

[13] American Sign Language (ASL) Alphabet (ABC) Poster. (n.d.). Gerard Aflague Collection. https://www.gerardaflaguecollection.com/products/american-sign-language-asl-alphabet-abc-poster.html

[14] Gesture recognition task guide. (n.d.-b). Google Developers. https://developers.google.com/mediapipe/solutions/vision/gesture_recognizer

[15] What Is Deep Learning? | How It Works, Techniques & Applications. (n.d.). MATLAB & Simulink. https://www.mathworks.com/discovery/deep-learning.html

[16] Singhal, G. (2020, September 9). Introduction to LSTM Units in RNN | Pluralsight. https://www.pluralsight.com/guides/introduction-to-lstm-units-in-rnn

[17] Dobilas, S. (2022b, February 26). GRU Recurrent Neural Networks — A Smart Way to Predict Sequences in Python. Medium. https://towardsdatascience.com/gru-recurrent-neural-networks-a-smart-way-to-predict-sequences-in-python-80864e4fe9f6