American Sign Language Recognition Using Deep Learning Models

Gauri Revankar[1] and Smitha Kumar[2]

[1] Heriot Watt University Dubai, UAE
dr2007@hw.ac.uk.com
[2] Heriot Watt University Dubai, UAE
smitha.kumar@hw.ac.uk

Abstract. Sign language is a mode of communication that enables individuals with hearing or speech impairment, or both, to express themselves. Sign Language Recognition from videos has become a new challenge in this research field. This paper focuses on Isolated Sign Language Recognition, which involves recognizing and interpreting phrases or words expressed through gestures and hand movements through a short video. With the advancement of convolutional and recurrent neural network architectures in Computer Vision, this paper proposes efficient deep learning models to recognize American Sign Language (ASL). The models implemented in this paper are ResNet50, ResNet50 + BiLSTM, Xception, and Xception + BiLSTM. Overall, ResNet50 + BiLSTM performed the best, with training, validation, and test accuracies of 79.37%, 69.56%, and 52.17%, respectively. The models were trained and evaluated on a 10-gloss subset of the WLASL dataset. Furthermore, a comparative analysis was performed with the models proposed in other research papers implemented for the same purpose.

Keywords: Sign Language Recognition · Isolated Sign Language Recognition · Deep Learning · American Sign Language · Transfer Learning · Computer Vision.

1 Introduction

Language is a cornerstone of human civilization, playing an important role as a means of communication to convey thoughts, emotions, and feelings. It takes various forms, including written symbols, gestures, and vocalizations. Sign Language is a visual language primarily used by the deaf and mute community to communicate. It involves a combination of hand shapes, hand and arm movements, and facial expressions to convey a message. Sign Language Recognition (SLR) is considered a part of behavior recognition, which is a field of research that involves identifying and understanding human movements, and gestures through pattern recognition, computer vision, etc [1]. Automating SLR is a complex and challenging task, requiring the integration of feature extraction techniques and classification methods [2]. Hence this project seeks to explore various hybrid deep learning models and compare their performances in Sign Language Recognition on an American Sign Language (ASL) dataset. Automating SLR serves as a possible solution to reduce the communication gap between the hearing and vocally impaired community and the rest of the world. The motivation for this research stems from the pressing need to narrow this gap between those with speech impairment, hearing impairment, or both, and the rest of society.

2 Related Works

This section contains Literature Reviews on research work done in the field of Sign language Recognition using Deep Learning.

Dongxu Li, Cristian Rodriguez Opazo, Xin Yu, Hongdong Li [5] proposed a pose-based Temporal Graph Convolution Network (TGCN) to capture the spatial and temporal components of the human body movement. Their implementation focused on implementing and comparing 2 types of models – a holistic visual appearance-based approach and a 2D human pose-based approach. Their proposed model boosted the performance of the pose-based approach achieving a 62.63% top-10 accuracy on the complete WLASL dataset consisting of 2000 words. The proposed model stacks multiple residual graph convolutional blocks and uses the average pooling result across temporal dimensions. The temporal motion is tracked using the 2D pose and hand keypoints, extracted using OpenPose.

Tunga, A., Sai, V., Nuthalapati and Wachs, J. [3] proposed Pose Based Sign Language Recognition using GCN and BERT. This involved the capturing of spatial and temporal features separately and performing late fusion. The proposed architecture uses a GCN to capture the spatial interactions in the video and BERT to capture the temporal dependencies between frames. The model was trained on the WLASL dataset [4]. The model achieved an accuracy of 88.67%, improving the prediction accuracy by 5% compared to GRU and TGCN models proposed by [5].

Yufeng Jiang, Fengheng Li, and Zongxi Li [9] attempted to enhance Continuous SLR with a self-attention mechanism and Media Pipe Holistic. MediaPipe Holistic was used to convert raw video into sequential data. Self-attention layer was employed to extract the spatiotemporal features. LSTM and Bi-LSTM were chosen as base models to compare the evaluations of the proposed model. Accuracy was chosen as their evaluation metric. An accuracy of 40.4% was achieved with face mesh and 59% without face mesh by self-attention mechanism. The authors encourage the use of video augmentation techniques and transfer learning to address the problem of insufficient training samples. The models were trained on the first 10 glosses of the WLASL dataset.

Fikri Nugraha, Esmeralda C. Djamal [6] proposed a Two-stream CNN for sign language recognition and classification. The two-stream CNN is composed of a spatial and temporal stream. These two streams were combined using the Average Fusion function. This reduced the training time as well as helped overcome resource limitations. The study used 10 words for classification from the

American Sign Language Lexicon Video Dataset. The best results were given by using Xception SGD for spatial flow and Xception Adam for temporal flow configuration.

Luke T. Woods and Zeeshan A. Rana [7] proposed an encoder-only transformer and used pose keypoint data for SLR. The authors used the enhanced version of the WLASL dataset. The model was trained on 2D keypoint coordinates (x and y) which were extracted using OpenPose. Since the model did not have a decoder, it greatly reduced the complexity of the model. Experiments were performed with 1,2,3,4,6 and 9 attention heads but with the same number of encoder layers. The model achieved a 96.88% accuracy on 10 signs, 87.22% accuracy on 50 signs, 83.16% accuracy on 100 signs, and 70.52% accuracy on 300 signs when evaluated on top-1 accuracy.

3 Proposed Methodology

The figure below depicts the project implementation workflow:

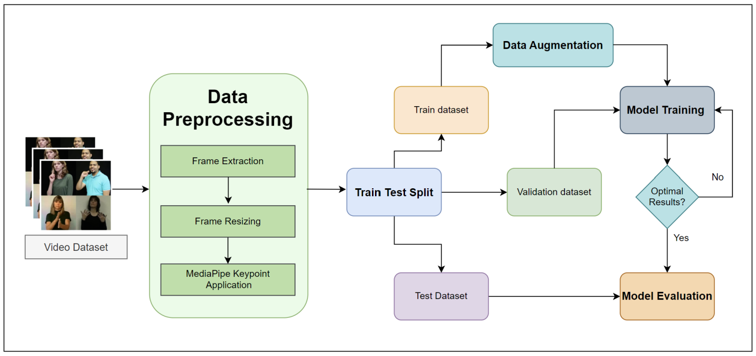


Fig. 1. Overview of the Project Implementation workflow

3.1 Datasets

The dataset that is used for training and testing purposes is the World Level American Sign Language (WLASL) video dataset [8] which is publicly available on Kaggle. It is a preprocessed dataset containing 12K videos of the WLASL dataset [5], which is a large-scale collection of annotated video samples covering a diverse range of American Sign Language (ASL) vocabulary.

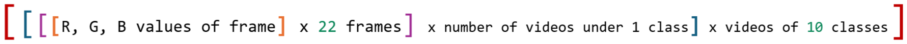
3.2 Data Preprocessing

The process of preparing the dataset can be broken down into 3 simple steps: 1. Frame Extraction 2. Frame Resizing 3. Application of MediaPipe Holistic keypoints

This implementation uses a subset of the WLASL dataset. The top 10 classes with the highest number of videos under them were selected. The following classes were selected – 'before', 'computer', 'cool', 'cousin', 'drink', 'go', 'help', 'take', 'thin', and 'who'. On average, there are around 11 videos per class.

Frame extraction is the process of extracting key frames or images from a video that summarize the video. Each of these images represents a single instance of the video in time. The implementation uses OpenCV for extracting frames from the videos in the dataset. 22 frames were extracted from each video. Since the videos were of unequal lengths, the frames were extracted equally across the length of the video. This ensures that the main movement of the action is captured, and avoids capturing the inactivity of the signer (if any), in the first few or last few frames.

The shape of the dataset is as follows:



[[[[R, G, B values of frame] x 22 frames] x number of videos under 1 class] x videos of 10 classes]

Fig. 2. Shape of the dataset

Pretrained models such as ResNet50 and Xception model, were trained on the ImageNet dataset that included images of size 224 x 224 pixels. Hence the video frames needed to be resized to meet the requirements of the model input structure. A center square was cropped out of the image to remove the extra background margins and only include the signer in the frame. This helped preserve the quality of the frames, which often deteriorates during resizing due to loss of pixels. The Open CV library was used to resize the frames. MediaPipe Holistic landmarks were then applied to the resized frames to track the body and hand movements of the signer. Only pose and hand landmarks were applied.

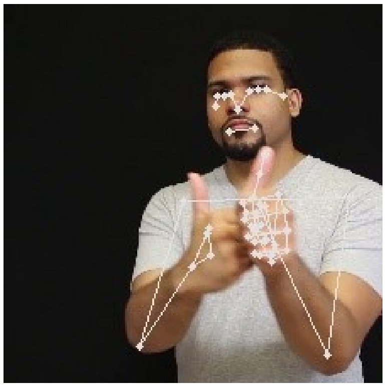


Fig. 3. Video frame after applying MediaPipe keypoints

The train test split module of the sklearn library was used to split the dataset into train, validation, and test datasets in an 70:15:15 ratio. The random state value to split the dataset was set to 42. Stratification was applied while splitting the datasets to ensure an equal number of videos from each class were used for model evaluation. To increase the number of training video samples in the dataset, video augmentation was used. The Image and Image Enhance modules of the PIL library were used to augment the videos. Frame mirroring and frame rotation video augmentation methods were used.

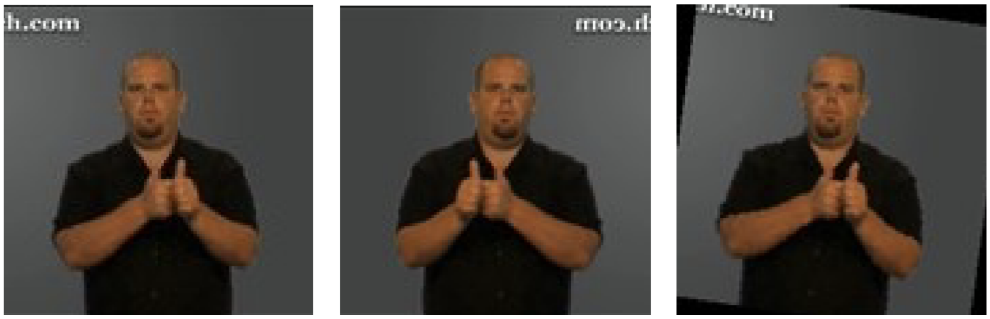


Fig. 4. (i) Original Frame (ii) Frame mirroring (iii) Frame rotation

3.3 Models

4 deep learning models are implemented in this paper namely - ResNet50, ResNet50 + BiLSTM, Xception, and Xception + BiLSTM. The concept of Transfer learning is used to implement ResNet50 and Xception models. The input shape of the pre-trained models is set to (224,224,3). The input passed to the ResNet50 and Xception models is of the shape (Number of videos, 22, 224, 224, 3). A Time Distribution layer is added to handle sequential data, hence enabling the processing of each video frame individually. To preserve the weights of the pre-trained models, all the weights of all the layers of the pre-trained models are frozen and the topmost layer of the model is excluded (since there are 10 output classes). Meanwhile the hybrid model structures - ResNet50 + BiLSTM and Xception + BiLSTM use ResNet50 and Xception models respectively as feature extractors of the videos in the dataset. Each video is passed frame by frame through the model for feature extraction and the extracted features are stored in a CSV file. After extracting the features, the CSV file is read and every 22 frames are grouped together to be stored as a video. The BiLSTM model architecture contains 3 BiLSTM layers with 64, 128, and 32 neurons respectively. These layers are followed by 4 dense layers with 64, 64, 64, and 32 neurons respectively. These layers have a ReLU activation function applied and l2 kernel regularizer with the value set to 0.001. The l2 regularizer helps in controlling the

complexity of the model and improves its generalization to new data. Finally, a dense layer with softmax function applied to it converges the model architecture to 10 output classes.

Below is a diagram to describe the process of feature extraction using a pre-trained model and how the data is passed to the BiLSTM model to train:

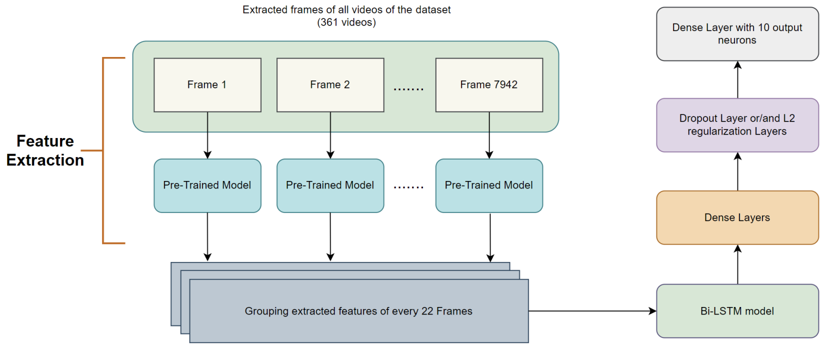

Extracted frames of all videos of the dataset
(361 videos)
Frame 1
Frame 2
....
Frame 7942
Feature Extraction
Pre-Trained Model
Pre-Trained Model
....
Pre-Trained Model
Grouping extracted features of every 22 Frames
Bi-LSTM model
Dense Layers
Dropout Layer or/and L2 regularization Layers
Dense Layer with 10 output neurons

Fig. 5. Multi-layer system architecture

3.4 Evaluation Metrics

For evaluating the models, Accuracy, Precision, Recall, and F1 score are the evaluation metrics that are used.

4 Model Evaluation and Testing

The 4 models were compiled as follows:

ResNet50

The model was trained on 100 epochs, with a batch size of 16 and a learning rate of 0.0001 with Adam optimizer.

ResNet50 + BiLSTM

The model was trained on 43 epochs (early stopping at 43 of 250 epochs), with a batch size of 8 and a learning rate of 0.0001 with Adam optimizer.

Xception

The model was trained on 10 epochs (the early stopping stopped the model training at 10 of 40 epochs), with a batch size of 8 and a learning rate of 0.001 with Adam optimizer.

Xception + BiLSTM

The model was trained on 40 epochs, with a batch size of 8 and a learning rate of 0.001 with Adam optimizer.

The tables below show the comparison of the performances of the models:

Table 1. Comparing Train, Validation and Test Accuracies

Sno.	Model	Train Accuracy	Val Accuracy	Test Accuracy
1.	ResNet50	73.97%	60.86%	52.17%
2.	ResNet50 + BiLSTM	79.37%	69.56%	52.17%
3.	Xception	82.54%	78.26%	52.17%
4.	Xception + BiLSTM	66.89%	69.56%	56.52%

Table 2. Comparing Precision and Recall

Sno.	Model	Val Precision	Test Precision	Val Recall	Test Recall
1.	ResNet50	59.78%	56.52%	60.86%	52.17%
2.	ResNet50 + BiLSTM	81.73%	60.28%	69.56%	52.17%
3.	Xception	73.91%	43.11%	78.26%	52.17%
4.	Xception + BiLSTM	65.79%	68.11%	69.56%	56.52%

Table 3. Comparing F1 Scores

Sno.	Model	F1-score Val	F1-score Test
1.	ResNet50	0.7004	0.5159
2.	ResNet50 + BiLSTM	0.8723	0.9145
3.	Xception	0.7536	0.4639
4.	Xception + BiLSTM	0.6514	0.5803

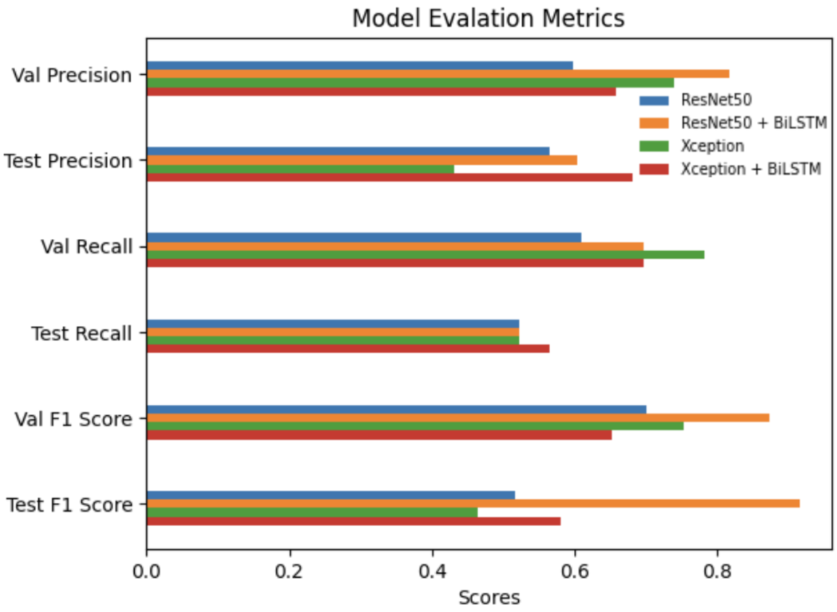


Fig. 6. Model Evaluation Metrics graph

Based on the training time and the results in the tables above, the following observations can be made:

1. The Xception models converged faster compared to the ResNet50 models.
2. In terms of accuracy, ResNet50 + BiLSTM performed better than the ResNet50 model and the Xception model performed better than Xception + BiLSTM.
3. Adding BiLSTM improved the model performance significantly, as seen in the case of ResNet50 + BiLSTM.
4. ResNet50 + BiLSTM performed the best in terms of F1 score, hence maintaining a better balance between precision and recall.
5. Based on the average accuracy across all datasets, Xception has the highest average accuracy (70.99%), followed by ResNet50 + BiLSTM (67.03%), Xception + BiLSTM (64.32%), and ResNet50 (62.33%).
6. Overall, ResNet50 + BiLSTM seems to perform the best across multiple metrics.
7. The results show signs of overfitting and the struggle of the models to generalize to new data.

4.1 Comparative Analysis

In this section, we compare the performances of our models with the models proposed by other researchers on a 10 classes/glosses subset of the WLASL dataset using model Accuracy as our evaluation criteria.

Table 4. Comparative Analysis based on Train, Validation and Test accuracies on 10 classes WLASL subset

Sno.	Model	Train Accuracy	Val Accuracy	Test Accuracy
1.	Encoder-Only Transformers [7]	100.00%	95.61%	93.46%
2.	Media Pipe + Self-attention [9]	80.00%	59.00%	-
3.	ResNet50 (Ours)	73.97%	60.86%	52.17%
4.	ResNet50 + BiLSTM (Ours)	79.37%	69.56%	52.17%
5.	Xception (Ours)	82.54%	78.26%	52.17%
6.	Xception + BiLSTM (Ours)	66.89%	69.56%	56.52%

The Encoder-only Transformer achieved this performance with 1 encoder layer with 1 attention head. [7] used Open Pose to extract pose and hand 2D keypoints from the video frames. These keypoints were later normalized before being passed to the model. The model had the architecture of a traditional transformer but it was modified to utilize only the encoder part. The model was run for 200 epochs. However, it is to be noted that [7] trained its model on the enhanced version of WLASL dataset, containing improved and enhanced classes/glosses. Hence the comparison of the performances is only indicative, which means that the authors acknowledge that using an improved version of the dataset may affect the validity of direct comparisons.

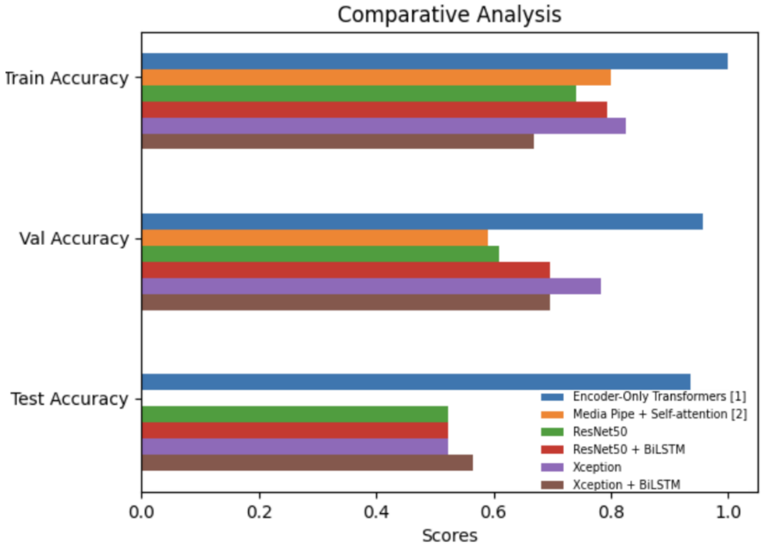


Fig. 7. Comparative Analysis graph

[9] proposed the use of Self-Attention mechanism for SLR. Media Pipe was used to extract keypoints from the videos. The self-attention layer was used to extract spatiotemporal features from the data. A sequence of these keypoints served as an input to the model. Experiments were carried out with and without the face mesh as well as with different sequence lengths. The model was trained for 2000 epochs.

Although our models performed better than the Self Attention model approach proposed by [9] in terms of validation accuracy, they still are overfitting and do not generalize very well to new data. The Encoder-only Transformer does surpass our models by a large margin in terms of the accuracies recorded.

5 Conclusion and Future Work

This paper aimed to explore and implement deep learning models and compare their performances for Sign Language Recognition. 4 deep learning models, namely - ResNet50 model, ResNet50 + BiLSTM hybrid model, Xception model, and Xception + BiLSTM hybrid model were implemented in this paper, and their performances were evaluated on a 10 gloss subset of the WLASL dataset. Several evaluation metrics were used, namely - accuracy, precision, recall, and f1- score. Accuracy and loss graphs were also plotted, in addition to confusion matrices. On evaluating the models, it was observed that the Xception models converged faster compared to the ResNet50 models. Overall, ResNet50 + BiLSTM performed the best, with training, validation, and test accuracies of 79.37%, 69.56%, and 52.17%, respectively.

A possible limitation of the project may be the small number of videos per class in the dataset. Although it is impressive that the dataset covers vocabulary up to 2000 words, the number of videos available under each gloss or class averages to 11. Hence, various video augmentation techniques were adopted to increase the number of videos under each class and, in turn, increase the size of the dataset. Another limitation of the chosen dataset is that many a time, 2 different glosses or classes contained the same sign videos. This can have a serious impact on the performances of the models. One potential limitation concerning the trained models may be the indications of overfitting. They also struggle to generalize well to new data. This, again, could be due to the lack of data to train the models. As a part of future work, we aim to use the enhanced version of the WLASL dataset (WLASL-alt dataset) as suggested by [7]. It contains the corrected and improved versions of the glosses present in the original WLASL dataset proposed by [5].

References

1. Jia Lu, Nguyen Minh, and Yan Wei Qi. Sign language recognition from digital videos using deep learning methods. In Minh Nguyen, Wei Qi Yan, and Harvey Ho, editors, Geometry and Vision, pages 108–118, Cham, 2021. Springer International Publishing.
2. Sign language recognition - an overview — sciencedirect topics. https://www.sciencedirect.com/topics/computer-science/sign-language-recognition.
3. Tunga Anirudh, Nuthalapati Sai Vidyaranya, and Wachs Juan. Pose-based sign language recognition using GCN and BERT. arXiv, December 2020.
4. Dongxu. WLASL: WACV 2020 "word-level deep sign language recognition from video: A new large-scale dataset and methods comparison".
5. Li Dongxu, Opazo Cristian Rodriguez, Yu Xin, and Li Hongdong. Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison. In 2020 IEEE Winter Conference on Applications of Computer Vision (WACV), pages 1448– 1458, 2020.
6. Fikri Nugraha and Esmeralda C. Djamal. Video recognition of american sign language using two-stream convolution neural networks. In 2019 International Conference on Electrical Engineering and Informatics (ICEEI), pages 400–405, 2019.
7. Luke T. Woods and Zeeshan A. Rana. Modelling sign language with encoder-only trans- formers and human pose estimation keypoint data. Mathematics, 11(9), 2023.
8. Risang Baskoro. WLASL (world level american sign language) video, September 2021.
9. Jiang Yufeng, Li Fengheng, Li Zongxi, Liu Ziwei, and Wang Zijian. Enhancing continuous sign language recognition with self-attention and mediapipe holistic. In 2023 8th Inter- national Conference on Instrumentation, Control, and Automation (ICA), pages 97–102, 2023.