

National University of Computer and Emerging Sciences



Real-Time Translator for Sign Language

**FYP Team**

Ubaid Ur Rehman………….19L-2343 (Lead)

Eemaan Amir………………19L-1130

Muhammad Usama…………19L-0999

**Supervised by**

Dr. Aamir Wali

**FAST School of Computing**

**National University of Computer and Emerging Sciences**

**Lahore, Pakistan**

**December 2022**

Anti-Plagiarism Declaration

This is to declare that the above publication produced under the:

**Title: \_\_\_\_\_\_\_\_\_\_\_\_Real-Time Translator For Sign Language\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

is the sole contribution of the author(s) and no part hereof has been reproduced on **as it is** basis (cut and paste) which can be considered as **Plagiarism**. All referenced parts have been used to argue the idea and have been cited properly. I/We will be responsible and liable for any consequence if violation of this declaration is determined.

Date: \_\_\_\_07-12-2022\_\_\_\_\_

Student 1

Name: \_\_\_\_\_\_\_Eemaan Amir\_\_\_\_\_\_\_

Signature: A picture containing text

Description automatically generated

Student 2

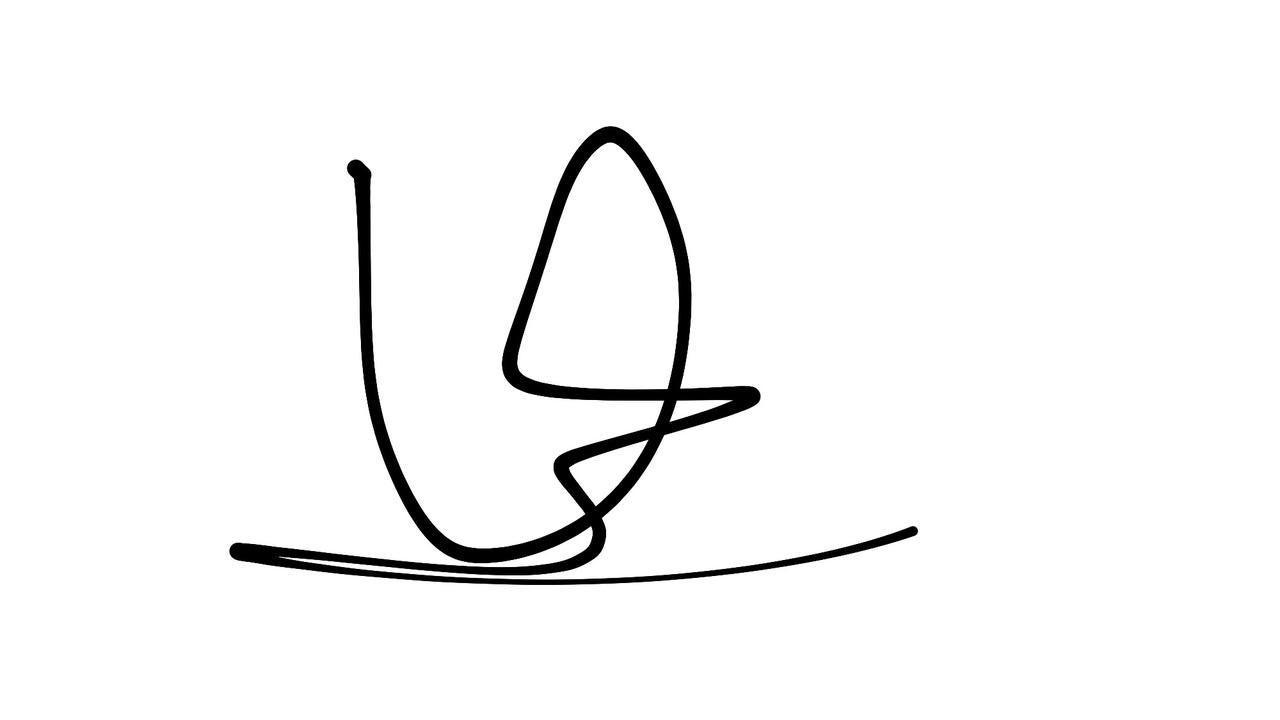
Name: \_\_\_\_\_Muhammad Usama\_\_\_\_\_\_

Signature: A picture containing diagram

Description automatically generated

Student 3

Name: \_\_\_\_\_Ubaid Ur Rehman\_\_\_\_\_\_\_

Signature: 

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Authors’ Declaration***

The work presented in this report is our own and has not been submitted/presented previously to any other institution or organization.

Abstract

Real-time communication poses significant challenges for individuals who are deaf or speech-impaired, affecting their academic, professional, and personal lives. This project addresses these challenges by developing a live communication interface to facilitate seamless interaction for these individuals. Our primary objective is to create a fast and smooth communication experience. While previous works in Sign Language Translation and Recognition have predominantly focused on accuracy, they have often overlooked the accessibility of real-world applications. To bridge this gap, we propose a real-time sign language translation system for American Sign Language (ASL) that is not only fast and accurate but also lightweight and compatible with various interfaces. Our solution leverages the MP holistic and Bi-LSTM models for sign language translation. To better address the communication needs of deaf and speech-impaired individuals, we embarked on an extensive endeavor, involving the creation of a purpose-built dataset for training and testing our model. The dataset underwent meticulous preprocessing to optimize its quality, including the removal of irrelevant points extracted from MP holistic, resulting in improved recognition speed. Subsequently, we developed a prototype to conduct live testing, simulating real-world scenarios and assessing the system's real-time output. The findings from our experiments validate the effectiveness and accuracy of our model, demonstrating its potential as a valuable tool in various applications. In conclusion, this project successfully implements a real-time sign language translation system using MP holistic and Bi-LSTM models. By providing a comprehensive solution, we aim to enhance communication for deaf and speech-impaired individuals, enabling them to keep pace with others and fostering inclusivity across domains.

**Executive Summary**

People who are deaf or have speech impairments frequently use sign language to communicate. It takes time to teach beginners to use sign language to understand and communicate. Sign Language Recognition (SLR) aims to decrease the communication barrier between sign language users and others. The project intends to provide an interface that will enable deaf and hearing users to establish a live video connection with real-time translation of sign language. We will strive to create a lightweight and fast model that can be used on any interface with ease. Our research aims to improve the way in which deaf and hearing-impaired people communicate with sign language users. It also aims to help people researching in the domain of sign language recognition and translation. Sign Language Translators (SLT) aims to replace traditional translation methods with vision-based ones employing Artificial Intelligence (AI), Machine Learning, and Deep Learning. Our main goal is to provide a lightweight, quick, and accurate real-time solution to SLR and SLT. We aim to bridge the gap between the hearing community and the deaf community. We provide a real-time system for sign language translation for the American Sign Language (ASL) The software is fast, accurate and lightweight enough to run on mobile devices. Sign language recognition is a problem that has been under research for quite a long time now. We hope to provide a solution that can be commonly used by the deaf community in real world communication scenarios like interviews, educational opportunities, video calls and other online communication as well as face to face communications. Skeleton-based SLR approach that distinguishes between the backdrop and the subject by using whole-body Keypoints and attributes from trained whole-body posture estimators. When compared to earlier project work, the application's integrity and adaptability are increased using image processing and convolution neural networks. A focus has been placed on understanding sign language translation and recognition, unlike other models. For end-level objectives of achieving accurate results, the transformer encoder with CTC loss trains to recognize and learn representation, but CNN models are typically overly reliant on input data preprocessing, and the Sign Pose SPOTER model has questionable computational efficiency for some datasets. We aim to establish a real-time fast and lightweight connection between a deaf and a hearing user using sign language. We will research and enhance a pre-made Skeleton Aware Sign Language Recognition Multi-model. To make the recognition process effective and quick, we seek to employ a skeleton-based model that can clearly discriminate between the backdrop and the subject and deliver high accuracy. Our goal is not sign language recognition, even though it is an essential first step in translating the signals into whole phrases in English. For this level, ISL and CSLR works that specifically address translation must be employed. The mentioned works provide strategies for gloss-free direct transcription of sign language into English. Using the MediaPipe Holistic and LSTM model, we offer a method for signing language recognition. Using machine learning techniques for detection and recognition, this strategy will also consider the "real time" aspect. Using the OpenCV library, we would retrieve camera frame data for keypoint detection and deliver the concatenated coordinates information. System requirements include all required diagrams and figures, as well as components for requirements and specifications. It goes into detail on functional requirements, design limitations, and other elements required to give a thorough and detailed description of the needs for the program. Following that, we have conducted several sample experiments and included all the specifications and configuration for our prototype. Detailed analysis and a summary of the findings are provided here. Finally, we have stated our achieved objectives so far and future plans for further enhancing our model’s performance.

Table of Contents

[Table of Contents i](#_Toc135945290)

[List of Tables iii](#_Toc135945291)

[List of Figures iv](#_Toc135945292)

[Chapter 1: Introduction 1](#_Toc135945293)

[1.1 Purpose of this Document 1](#_Toc135945294)

[1.2 Intended Audience 1](#_Toc135945295)

[Chapter 2: Project Vision 2](#_Toc135945296)

[2.1 Problem Domain Overview 2](#_Toc135945297)

[2.2 Problem Statement 2](#_Toc135945298)

[2.3 Problem Elaboration 2](#_Toc135945299)

[2.4 Goals and Objectives 2](#_Toc135945300)

[2.5 Project Scope 3](#_Toc135945301)

[2.6 Sustainable Development Goal (SDG) 3](#_Toc135945302)

[2.7 Constraints 3](#_Toc135945303)

[Chapter 3: Literature Review / Related Work 4](#_Toc135945304)

[The following section elaborates on the detailed literature review of previous research done in accordance with the chosen topic that is Sign Language Translation and the detailed knowledge of the techniques and models that have been used previously. 4](#_Toc135945305)

[3.1 Definitions, Acronyms, and Abbreviations 4](#_Toc135945306)

[3.1.1 Definitions 4](#_Toc135945307)

[3.1.2 Abbreviations 4](#_Toc135945308)

[3.2 Detailed Literature Review 5](#_Toc135945309)

[3.2.1 Skeleton Aware Multi-modal Sign Language Recognition 5](#_Toc135945310)

[3.2.2 OpenHands: Making Sign Language Recognition Accessible with Pose-based Pretrained Models across Languages 7](#_Toc135945311)

[3.2.3 Spatial-Temporal Graph Convolutional Networks for Sign Language Recognition 10](#_Toc135945312)

[3.2.4 Keypoint based Sign Language Translation without Glosses 12](#_Toc135945313)

[3.2.5 Part-based Graph Convolutional Network for Action Recognition 14](#_Toc135945314)

[3.2.6 Generative Multi-Stream Architecture for American Sign Language Recognition 17](#_Toc135945315)

[3.2.7 Open-Domain Sign Language Translation Learned from Online Video 18](#_Toc135945316)

[3.2.8 A Task-aware Instruction Network for Sign Language Translation Enhanced with Data Augmentation 21](#_Toc135945317)

[3.2.9 An Improved Sign Language Translation Model with Explainable Adaptations for Processing Long Sign Sentences 23](#_Toc135945318)

[3.2.10 Word-level Deep Sign Language Recognition from Video: A New Large-scale Dataset and Methods Comparison 25](#_Toc135945319)

[3.2.11 Sign Language Text to Speech Converter using Image Processing and CNN 27](#_Toc135945320)

[3.2.12 Sign Language Recognition Using Convolutional Neural Networks 29](#_Toc135945321)

[3.2.13 Sign Language Transformers: Joint End-to-end Sign Language Recognition and Translation 31](#_Toc135945322)

[3.2.14 Sign Pose-based Transformer for Word-level Sign Language Recognition Networks 34](#_Toc135945323)

[3.3 Literature Review Summary Table 37](#_Toc135945324)

[Chapter 4: Software Requirement Specifications 42](#_Toc135945325)

[4.1 Functional Requirements 42](#_Toc135945326)

[4.2 Hardware and Software Requirements 42](#_Toc135945327)

[4.2.1 Hardware 42](#_Toc135945328)

[4.2.2 Software 42](#_Toc135945329)

[4.3 Use Cases 43](#_Toc135945330)

[4.3.1 Sign Language Recognition 43](#_Toc135945331)

[Chapter 5: Proposed Approach and Methodology 44](#_Toc135945332)

[5.1 Dataset 44](#_Toc135945333)

[5.2 Key Points Using MediaPipe Holistic 44](#_Toc135945334)

[5.2.1 MediaPipe Holistic 45](#_Toc135945335)

[5.2.2 Open CV 46](#_Toc135945336)

[5.3 Extract Keypoint Values 46](#_Toc135945337)

[5.4 Collect Keypoint Values 46](#_Toc135945338)

[5.4.1 Models 47](#_Toc135945339)

[5.5 Preprocessing 47](#_Toc135945340)

[5.6 LSTM Neural Network 47](#_Toc135945341)

[5.7 LSTM and MP Holistic 48](#_Toc135945342)

[5.8 Bi-LSTM Neural Network 49](#_Toc135945343)

[5.9 Categorical Cross-Entropy Loss 50](#_Toc135945344)

[5.10 Methodology Conclusion 51](#_Toc135945345)

[Chapter 6: Implementation and Test Cases 52](#_Toc135945346)

[6.1 Implementation 52](#_Toc135945347)

[6.1.1 Environment Setup 52](#_Toc135945348)

[6.1.2 Preprocessing 52](#_Toc135945349)

[6.1.3 Feature Extraction 53](#_Toc135945350)

[6.1.4 Sign Language Recognition Models 53](#_Toc135945351)

[6.2 Test Case Design and Description 55](#_Toc135945352)

[6.2.1 Test Case No.1 55](#_Toc135945353)

[6.3 Test Metrics 56](#_Toc135945354)

[6.3.1 Sample Test Case Matric.No.1 56](#_Toc135945355)

[6.4 Prototype 56](#_Toc135945356)

[Chapter 7: Experimental Results and Discussion 58](#_Toc135945357)

[7.1 Dataset Testing 58](#_Toc135945358)

[7.2 Live Testing 58](#_Toc135945359)

[7.2.1 Experimental Setup 58](#_Toc135945360)

[7.2.2 Experimental Results 59](#_Toc135945361)

[7.3 Results and Discussion 60](#_Toc135945362)

[Chapter 8: Conclusion 62](#_Toc135945363)

[8.1 Critical Analysis of The Research Items 62](#_Toc135945364)

[8.1.1 Strengths 62](#_Toc135945365)

[8.1.2 Weakness 62](#_Toc135945366)

[8.2 Relevance Of Review to Proposed Research Work 62](#_Toc135945367)

[8.3 Motivation Behind Methodology 63](#_Toc135945368)

[8.4 Results and Accuracy 63](#_Toc135945369)

[8.5 Challenges Faced 64](#_Toc135945370)

[8.6 Completed Work 64](#_Toc135945371)

[References 65](#_Toc135945372)

List of Tables

[Table 1](#_heading=h.2p2csry): 7 datasets used to test OpenHands Library 9

[Table 2:](#_heading=h.3mzq4wv) Results of Open Hands Library on 7 dataset 9

Table 3: Comparison of results of STGCN with other models 12

Table 4: Results of Keypoint based Sign Language Translation without Glosses 14

Table 5: Results of Part-based Graph Convolutional Network for Action Recognition 16

Table 6: Translation performance of baseline models and Open-Domain Sign Language Translation 20

Table 7: Comparing the translation performance of TIN-SLT 22

Table 8: Comparing the translation Word-level Deep Sign Language Recognition 27

Table 9: Validation Results of SLR using CNN 31

Table 10: Results Sign pose-based transformer on WLASL 36

Table 11:Top1 average recognition accuracy 36

Table 12: Work done on Sign Language Recognition 37

Table 13: Use case Details Table 43

Table 14: Test Case No.1 55

Table 15: Test Case Matric.No.1 56

Table 16: Testing Data Results 58

Table 17: Experimental Setup Summary 59

Table 18: Live Testing Results 59

List of Figures

[Figure 1:](#_heading=h.1mrcu09) STGCN mapping of whole body skeleton 11

[Figure 2:](#_heading=h.3l18frh) Geometric Features for motion recognition 15

[Figure 3:](#_heading=h.206ipza) Fusion and sequence modeling architecture (Open-Domain SLT) 20

[Figure 4:](#_heading=h.2afmg28) Architecture for Task-aware Instruction Module 22

[Figure 5:](#_heading=h.39kk8xu) Result of Sign Language translator tool 29

[Figure 6:](#_heading=h.48pi1tg) Result of background removal samples 30

[Figure 7:](#_heading=h.haapch) Architecture of deep learning model 30

[Figure 8:](#_heading=h.319y80a) A detailed overview of a single layered SLT 32

[Figure 9:](#_heading=h.1gf8i83) Standard Transformer is used with an Encoder and Decoder 35

Figure 10: MP Holistic Demo 45

Figure 11: BlazePose Detection Model 46

Figure 12: Flow Diagram of Our System 49

Figure 13: Bi-LSTM Neural Network 50

Figure 14: Visual Representation of Categorical Cross-Entropy Loss 50

Figure 15: Sequential LSTM Model Architecture 1 54

Figure 16: Sequential LSTM Model Architecture 2 55

# Introduction

Deaf or speech-impaired persons frequently use sign language for communication, which takes an ample amount of practice to become proficient in. Training beginners to understand and convey information through Sign language is time consuming and judging by the underlying statistics of people familiar with its semantics, taking this approach isn’t feasible. By detecting signals from given videos, Sign Language Recognition (SLR) aims to decrease the communication barrier between sign language users and others. It is a difficult but vital work since sign language necessitates the rapid and intricate motion of hands, body position, and even facial emotions. Skeleton-based sign identification has lately gained popularity due to the independence between the object and backdrop variance.

Our project aims to use a LSTM model for sign language recognition to help solve this major problem. The project intends to provide an interface that will enable deaf and hearing users to establish a live video connection with real-time translation of sign language. We will strive to create a lightweight and fast model that can be used on mobile devices with ease.

We have started our documentation with a basic abstract and later in Chapter 1, we gave a brief background on the Sign language problem and our proposed solution to it (plus purpose and targeted audience for our research work and application). Then, Chapter 2 covers the project vision, where we have elaborated our problem faced in SLR, our goals and objectives (results we are expecting on using the skeletal model) and common limitations faced on working on Sign language recognition. We proceed to literature review in Chapter 3 which contains summarized information about different approaches to SLR (Skeletal, Transformer, CNN) collected from 14 research articles plus a table to briefly mention each article. Lastly, in our Conclusion (Chapter 4), we ended our deliverable by stating the strength and weakness of the models we studied about in our literature review and stated different ways in which this research is related to our Project work.

## Purpose of this Document

Communication is a core and crucial component in our daily lives. In this digitalized era, convenient and faster means of conversation is essential. Unfortunately, there are several barriers in communication for individuals who are either speech or hearing impaired. Our goal through this project is to provide a faster and more effective mobile application interface for these individuals to translate their mode of communication that is sign language to written subtitles to communicate in real time with rest.

The basic purpose of this document is researching and collecting data based on sign language translation or recognition. For this project, we have reviewed 14 documents (along with a few available codes) which contain model workings and algorithms already implemented on this issue. We have filtered out the best possible techniques and accuracies and so we can take inspiration from them and use them together in an efficient way to provide the best possible solution.

## Intended Audience

The targeted audience for our project will be speech impaired individuals who are already experts in Sign Language to communicate with those who are not and individuals with hearing disability to communicate with Sign language users. Furthermore, our research work will also contribute to the research community and by making our progress public, with an aim to help people researching in the domain of sign language recognition and translation.

# Project Vision

The following chapter lists all the details regarding the project we have undertaken.

## Problem Domain Overview

By substituting all device-based procedures with vision-based ones employing Artificial Intelligence (AI), Machine Learning, and Deep Learning, the discipline of sign language translation seeks to eliminate all barriers to communicating with deaf individuals.

Our work will focus on accepting input from common cameras used in mobile devices, which are not as high-quality as most datasets. Furthermore, we intend to create a solution to segment and extract and recognize individual signs from RGB input photos. We wish to provide both isolated and continuous translations of American Sign Language. We intend to not only recognize signs on a gloss level but also translate them into suitable English language. Our main goal is to provide a lightweight, quick, and accurate real-time solution to SLR and SLT..

## Problem Statement

Deaf or speech-impaired persons frequently use sign language to communicate, but it takes a lot of practice to become proficient. Thus, a great gap has been created between the hearing community and the deaf community. We aim to bridge this gap by providing a real-time system for sign language translation for American Sign Language (ASL) that is fast, accurate, and lightweight enough to run on mobile devices.

## Problem Elaboration

Sign language recognition is a problem that has been under research for quite a long time now. There have been several substantial developments in the field that have significantly improved the accessibility and accuracy of sign language translation models. However, all the past methods provide methods that require extensive computing and time as well as proper hardware for high-quality input. Thus, most methods, despite having excellent accuracy, are not directly applicable in real-world scenarios. Our focus lies on this exact issue and on contributing to solving it in an efficient manner. We hope to provide a solution that can be commonly used by the deaf community in real-world communication scenarios like interviews, educational opportunities, video calls, and other online communication as well as face-to-face communications. The subproblems our solution will focus on are:

* Accepting input from common cameras that are available in mobile devices that are not as high quality as most training data available.
* Segmenting and extracting individual signs from input RGB images.
* Recognizing individual signs from video segments.
* Providing isolated as well as continuous translation of the said signs.
* Not only recognizing the signs on gloss level but also translating them into proper English sentences.
* Providing a real-time lightweight, fast and accurate solution to all of the above-mentioned subproblems.

## Goals and Objectives

We will research and enhance a pre-made Skeleton Aware Sign Language Recognition Multi-model to achieve our goals. The main goals and objectives of our project are as follows:

* Establish a real-time fast and lightweight connection between a deaf and a hearing user.
* Detects and segments sign language in the live video.
* Recognize and identify signs from segmented video.
* Translate the identified signs to “sign gloss’ i.e., a label.
* Convert sign gloss to translated English sentences.

## Project Scope

This project will use computer vision libraries to process sequences of images extracted from live video. Furthermore, artificial intelligence and machine learning will be used to recognize sign language from processed images. The project will also be utilizing a simple speech-to-text model.

## Sustainable Development Goal (SDG)

Our project’s main SDG is Goal 10 that is Reduced Inequalities. Our research work will try to bridge the gap between the hearing and the deaf community which will as a result ensure that everyone can avail similar and equal opportunities in every field of life. We hope to open the possibility of equal educational opportunities in accordance with the Goal 10 i.e., Quality Education.

## Constraints

The present trends in SLR include the following limitations.

The accuracy is hampered by the obstacle considering the several signers:

* Break off between the letter/ sign and speed up sign performing: The signer's rapid, successive, and repetitive signing makes feature extraction and segmentation difficult.
* Obstruction of hand-face and hand-hand overlapping.
* Long sleeves on a dress and colorful gloves have an impact on the ability to recognize signs.
* Interpersonal variation is high: Sign differs between signers and moments.

The obstacle in the video domain is:

It is not feasible to manage the video data inside the constrained GPU memory. Most CNN approaches only use movies with extratemporal information that are image-based. To complete the categorization and fine-tuning procedure on each frame separately, a straightforward scaling procedure may result in the loss of essential temporal information.

# Literature Review / Related Work

## The following section elaborates on the detailed literature review of previous research done in accordance with the chosen topic that is Sign Language Translation and the detailed knowledge of the techniques and models that have been used previously.

## Definitions, Acronyms, and Abbreviations

### Definitions

* Adam Optimizer: Adam is a replacement optimization algorithm for stochastic gradient descent for training deep learning models.
* Gloss: When a word is associated with a sign is called a GLOSS: In simplest terms, a GLOSS is a label.
* Pytorch Lightning: PyTorch Lightning is an open-source Python library that provides a high-level interface for PyTorch, a popular deep learning framework.
* Seq2Seq model: A Seq2Seq model is a model that takes a sequence of items (words, letters, time series, etc) and outputs another sequence of items. In the case of Neural Machine Translation, the input is a series of words, and the output is the translated series of words.
* SoftMax: Softmax is a mathematical function that converts a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector.

### Abbreviations

* 2D/3D: 2/3 -Dimensions
* ANN: Artificial neural network
* ASL: American Sign Language
* ASLU: American Sign Language University
* BERT: Bidirectional Encoder Representations from Transformers Language Model
* C3D: 3-D convolutional model
* CNN: Convolution Neural Network
* CPU: Central processing unit
* CSLR: Continuous Sign Language Recognition
* CTC: Connectionist Temporal Classification (Loss for difficult alignment)
* DCNN: Deep convolutional neural network
* DH-BiGRU: Dynamic hierarchical bidirectional GRU
* DTW: Dynamic Time Warping
* FF: Feed Forward
* FSDC: Frame stream density compression
* GAN: Generative Adversarial Network model
* GEM: Generalized Ensemble Model
* GPU: Graphics processing unit
* GRU: Gated Recurrent Unit
* HMM: Hidden Markov Models
* HOG: Histogram of oriented gradients
* HSV: Hue Saturation and Value
* I3D: Inflated 3D Networks
* ISL: Isolated Sign Language Recognition
* LCN: Local contrast normalization
* LSTM: Long short-term memory artificial neural network
* NLP: Natural Language Processing
* NMT: Neural Machine Translation
* PB-GCN: Part-based graph convolutional network
* rC3D: Spatio-Temporal Residual Block added to the 3-D convolutional model
* ReLU: Rectified linear activation function
* RGB: Red, Green, Blue Track
* RGBD: RGB and Depth track
* RNN: Recurrent Neural Networks
* ROI: Return on Investment
* SAM-SLR: Skeleton Aware Multi-Modal Sign Language Recognizer
* SL-GCN: Sign Language Graph Convolution Network
* SL: Sign Language
* SLRT: Sign Language Recognition Transformer
* SLTT: Sign language Translation Transformer
* SSTCN: Separable Spatial-Temporal Convolution Network
* STMC: Spatial-Temporal Multi-Cue Network
* SVM: Support Vector Machine
* T-Conv: Temporal convolution
* TC-DHBG-Net: Temporal convolution, and dynamic hierarchical bidirectional GRU hybrid network
* TIN-SLT: Task-aware Training Network for Sign Language Translation
* VGG: Visual Geometry Group
* WLASL: word-level ASL

## Detailed Literature Review

In this section we have included a summary of the research papers we have studied so far.

### Skeleton Aware Multi-modal Sign Language Recognition

[1] For sign language to be clearly and precisely expressed, both distinctive arm/hand motions and whole-body movements are necessary. Additionally, emotions can be expressed through facial expressions. Depending on how many times a similar gesture is repeated, it can potentially impose different meanings. Second, SLR can be more difficult since sign language may be performed in a variety of ways by various signers (for example, body shapes, left- or right-handers, and speed).

Jiang and Sun suggest a brand-new skeleton-based SLR technique that makes use of characteristics from trained whole-body posture estimators as well as whole-body Keypoints. In order to simulate the dynamics contained, they develop a new spatiotemporal skeleton graph for SLR and suggest a Sign Language Graph Convolution Network (SL-GCN). A novel Separable Spatial-Temporal Convolution Network (SSTCN) was suggested for the whole-body skeleton characteristics to properly utilize the data in whole-body Keypoints. A Skeleton Aware Multi-modal SLR framework (SAM-SLR) was created to combine the suggested skeleton-based approach with additional paradigms in both RGB-D and RGB scenarios, to significantly increase the identification rate. Following are all the methods introduced in the research paper:

* Using the whole-body Key Points and graph reduction, a unique skeleton graph was created for SLR. This approach uses a whole-body posture estimator that has been previously trained and requires no additional annotation work.
* To extract data from the whole-body skeleton graph, A SL-GCN is used. This, according to Jiang, is the first time the SLR challenge has been successfully completed utilizing whole-body skeleton graphs.
* The accuracy of whole-body Key Points can be greatly increased as compared to the conventional 3D convolution by using a unique SSTCN to further leverage whole-body skeleton data.
* A SAM-SLR system that learns from 6 methodologies and achieves cutting-edge performance on the AUTSL dataset for RGB-D and RGB-based SLR.

133 Keypoints are estimated from the recognized individual in videos using a pre trained whole-body posture estimation network. Then, a spatial-temporal graph can be created by joining the neighboring Keypoints in the spatial dimension in a manner similar to how the human body naturally connects, and by joining every Keypoint to every other Keypoint in the temporal dimension. The whole-body skeleton graph's numerous edges and nodes add a lot of distortion to the model. It was discovered that employing just a whole-body skeleton graph with all 133 nodes results in low accuracy. Thus, the approach reduces the 133 nodes in the whole-body skeleton graph to just 27 nodes through the process of graph reduction. The remaining graph consists of 7 nodes for the upper body and 10 nodes for each hand.

Multi-modal Data Preparation is done in the following way:

* Whole-body Pose Keypoints and Features
  + From the RGB films, 133-point whole-body Keypoints are estimated to create the 27-node skeleton graph.
  + For each frame's skeleton features, 33 joint features are selected. Each video will have a standardized sample size of 60 frames.
* RGB Frames and Optical Flow
  + For speedier parallel loading and processing, RGB videos' whole frames are retrieved and saved as pictures.
  + Using the Keypoints derived from whole-body posture estimation, optical flow frames and RGB frames are resized to 256x256.
* Depth HHA and Depth Flow
  + HHA characteristics, which stand for the horizontal disparity, height above the ground, and angle normal, encode depth info into an RGB-like 3-channel output. The identification rate increases, and the picture can be better understood by employing HHA rather than straight-up gray-scale depth movies.
  + To derive optical flow from the depth (called depth flow) modality, the exact same steps as for RGB are used. Compared to the RGB flow, the depth flow is clearer and captures additional information.

Jiang and Sun combine the four modalities using a straightforward ensemble approach. The output of the final completely connected layers in each methodology before the softmax layer is specifically saved. The size of those outputs is nc, where nc is the total number of classes. The final predicted score is then calculated by adding the weights assigned to each modality in accordance with their accuracy on the validation set, where q stands for the result of each modality and α1,2,3,4,5,6 are hyper-parameters that need to be changed based on accuracy of the ensemble on the validation set. Utilizing the argmax () function, maximum score indices are discovered as the final projected classes. In the experiments, they use α = [1, 0.9, 0.4, 0.4] for RGB track and α = [1.0, 0.9, 0.4, 0.4, 0.4, 0.1] for RGB-D track.

As a result, SAM-SLR achieved the highest performance in both RGB (98.42%) and RGB-D (98.53%) tracks in 2021.

### OpenHands: Making Sign Language Recognition Accessible with Pose-based Pretrained Models across Languages

[2] Selvaraj and Kumar introduce OpenHands, a library where word-level recognition for sign languages is achieved by using four basic principles from the NLP field for low-resource languages.

1. Firstly, to decrease training time and facilitate effective inference, they suggest utilizing pose extracted via pretrained models as the standard modality of data. To support this, they have released standardized pose datasets for six different sign languages: Greek, Argentinian, Indian, Chinese, and Turkish, American
2. Secondly, to provide baselines and deployable checkpoints, In all 6 languages, they create and publish checkpoints for 4 pose-based isolated sign language recognition models.
3. Thirdly, they suggest self-supervised pretraining on unlabeled data to alleviate the shortage of labeled data. The greatest pose-based pretraining dataset for Indian Sign Language has been assembled and released by OpenHands (Indian-SL).
4. Finally, by contrasting various pretraining approaches and establishing, for the first time, proving pre training is advantageous for understanding sign language by demonstrating:
   1. Enhanced fine-tuning performance, particularly in low-resource situations, and
   2. A greater cross-lingual transition to a select few other sign languages from Indian-SL.

In an effort to make sign language research widely accessible, OpenHands have made all models and datasets open-source.

Isolated Sign Language Recognition (ISL) refers to the recognition of individual signals, whereas Continuous Sign Language Recognition (CSLR) refers to the translation of sign language as whole sentences (ISLR). Various initiatives have been made to create datasets and models for ISLR and CLSR applications. However, these findings are frequently limited to a small number of sign languages (such as ASL) and are made available among numerous study communities with limited standardized reference points. The advancement of NLP research using text and speech is substantially ahead of that of sign languages. The larger NLP community has just become aware of this latency.

Selvaraj and Kumar train pose-based ISLR models on all datasets using same training settings to facilitate a uniform contrast of models. Sequence-based models and graph-based models are the two categories to which these models belong. RNN and Transformer based designs are taken into account for sequence-based models. A 4-layered bidirectional LSTM with a hidden layer dimension of 128 is utilized for the RNN model, and it accepts as input a vector of 54 points every frame, or the framewise posture representation of 27 Keypoints with 2 coordinates each. Additionally, a temporal attention layer is employed to rank the best slides for classification.

A BERT-based architecture is employed for the Transformer model, which consists of 5 Transformer-encoder layers with six attention heads, a hidden dimension size of 128, and a maximum sequence length of 256. Selvaraj takes ST-GCN and SL-GCN models into account for the graph-based models. The ST-GCN model is then used, which consists of 10 Spatio-temporal GCN layers, with the spatial dimension of the graph consisting of 27 Keypoints with a depth of 2, which corresponds to the two coordinates. They apply the same 10 SL-GCN blocks for the SL-GCN model as the ST-GCN model, along with the identical graph structure and hyperparameters.

4 models in total are trained - LSTM, BERT, ST-GCN, and SL-GCN- for every dataset out of the total 7. The data processing and training pipelines make use of PyTorch Lightning. Every model is trained using the Adam Optimizer.

**Table 1: 7 datasets used to test OpenHands Library**

*Following are the details of the 7 datasets supported by OpenHands Library*

*that were used in training*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Dataset** | **Language** | **Vocab** | **Signers** | **Videos** | **Hours** | **Data** |
| AUTSL | Turkish | 226 | 43 | 38,336 | 20.5 | RGB |
| CSL | Chinese | 100 | 5 | 500 | 108.84 | RGBD |
| WLASL | American | 2000 | 119 | 21,083 | 14 | RGB |
| GSL | Greek | 310 | 7 | 40785 | 6.44 | RGBD |
| LSA64 | Argentinian | 64 | 10 | 3,200 | 1.90 | RGB |
| DEVISIGN | Chinese | 4414 | 30 | 331050 | 21.87 | RGBD |
| INCLUDE | Indian | 263 | 7 | 4,287 | 3.57 | RGB |

**Table 2: Results of Open Hands Library on 7 datasets**

*Following are the detailed results of all 4 models available in the OpenHands Library*

*on each of the 7 datasets.*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Dataset** | **Language** | **State-of-the-art**  **(pose) model** | | **Model Available in OpenHands** | | | |
|  | | **Model (Params)** | **Accuracy** | **LSTM** | **Transformer** | **ST-GCN** | **SL-GCN** |
| INCLUDE | Indian | Pose-XGBoost | 63.10 | 83.0 | 90.4 | 91.2 | 93.5 |
| AUTSL | Turkish | Pose-SL-GCN | 95.02 | 77.4 | 81.0 | 90.4 | 91.9 |
| GSL | Greek | Pose-Attention | 83.42 | 86.6 | 89.5 | 93.5 | 95.4 |
| DEVISIGN\_L | Chinese | RGB-iRDML | 56.85 | 37.6 | 48.9 | 55.8 | 63.9 |
| CSL | Chinese | RGBD-13D | 95.68 | 75.1 | 88.8 | 94.2 | 94.8 |
| LSA64 | Argentinian | Pose-LSTM | 93.91 | 90.2 | 92.5 | 94.7 | 97.8 |
| WLASL2000 | American | Pose-TGCN | 23.65 | 20.6 | 23.2 | 21.4 | 30.6 |
|  | | **Average Accuracy** | | 69.38 | 73.47 | 77.43 | 80.69 |

Pretraining is done on randomly chosen continuous input streams of length 60–120 frames (approximating 2-4 secs with 30fps videos). Models are then refined using an additional classification head on the relevant ISLR dataset.

* Masking-based pretraining
* Contrastive-learning based pretraining
* Predictive coding based pretraining

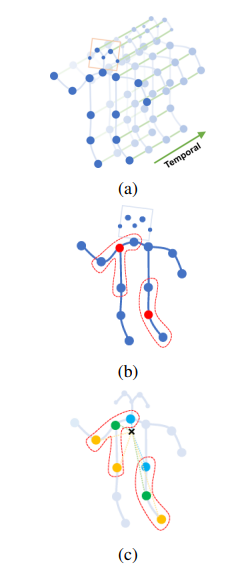
The OpenHands library was used to open-source every contribution to this publication. An Indian SL pretraining corpus with more than 1,100 hours of pose data is provided, along with pose-based datasets for each of the six SLs, four ISLR models trained on seven datasets, pretrained models on this corpus for all three pre training techniques, and models refined for four different SLs on top of the pretrained model. Additionally, it provides scripts for efficient deployment that make advantage of MediaPipe posture estimations and their refined ISLR models.

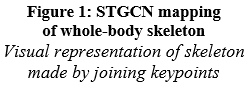
### Spatial-Temporal Graph Convolutional Networks for Sign Language Recognition

[3] One in ten people, or 360 million individuals globally, suffer from a severe hearing loss, according to the World Health Organization. The significance and range of sign language in many of these people's communication is further highlighted by this data. Despite this, very few hearing people can communicate using sign language.

Graph, spatial-temporal A novel method for recognizing human behavior based on spatial-temporal graphs is called convolutional networks (STGCN). By employing graph representations of the human skeleton, it concentrates on physical mobility and interactions between its components while neglecting the impact of the environment around them. Additionally, it manages motions in both the temporal and spatial dimensions, enabling us to capture the dynamic characteristics of activities throughout time. These qualities are applicable while coping with the challenges and peculiarities of sign language recognition. There are some new ways defined, such as:

* sign identification based on human movement, which considers various aspects of its dynamics and helps to address some of the fundamental problems in the area.
* the production of a brand-new human skeleton dataset for sign language, which will aid in the advancement of research in this field.

The ST-formulation GCN's is based on a set of skeletal graphs of the human body that were gathered from several people's activity frames. This structure is shown in Figure 1a, where each node corresponds to an articulation point. The body's natural connections determine the intra-body vertices. To depict their development over time, the inter-frame vertices link the same joints throughout succeeding frames.

First, based on the estimated skeletons of the people in the input videos, space-time graphs are built. Then, further ST-GCN convolution layers are added to the graphs that are shown, eventually producing feature maps with ever-higher degrees of detail. They are then submitted into a classifier, which chooses the appropriate course of action. It is crucial to first show the sampling and partitioning algorithms used by the ST-GCN in order to fully understand how it functions. It is simple to visualize a rigid grid (or rectangle) centered on a central point when dealing with convolutions over 2D images since it represents the sample region of the convolutional filter and defines the neighborhood. However, in the context of graphs, this definition must be broadened to include points that are immediately surrounding the center point and are joined by a vertex.

This definition is shown in Figure 1b for a single frame. The sample region for the convolutional filter for the red center points is shown by the dashed edges. Additionally, the algorithm does not look at sites that are physically close to the central points (such the points of the feet, knees, and waist) unless there are vertices connecting them to the red ones. The ST-GCN sampling strategy looks like this. The convolutional filter only looks at points that are closely linked to the core points, as shown in Figure 1b. In other words, neighbors with a distance D=1 are considered while defining the filter zone. In the ST-GCN proposal article, the authors determined this distance.

As shown in Figure 1c, the partitioning approach is based on the location of the joints and the characteristics of human body movement. The writers use eccentric or concentric movements to categorize various body sections., and the locations in the sample region are divided into three subsets:

* The root node (or center point, marked green in Figure 1c)
* The centripetal group (blue dots in the Figure 1c), which are the neighborhood nodes that are closest to the center of gravity of the skeleton
* The centripetal group (blue dots in the Figure 1c), which are the neighborhood nodes that are closest to the center of gravity of the skeleton

A new collection of human bones for sign language was provided based on the American Sign Language Lexicon Video Collection (ASLLVD). The ASLLVD is a sizable public dataset that includes annotations, start and end frame markers, and class labels for each sample along with video sequences of thousands of American Sign Language (ASL) signals. Following are some steps that how it works,

* obtaining the videos
* segment the videos
* estimating the skeletons
* filtering the key points.
* division of the dataset
* normalize and serialize

Between epochs 330 and 380, the model had an average sign detection accuracy of 61.04%. Consequently, the average top-5 accuracy was 86.36%. This performance fared better than conventional methods like MEI, MHI, and PCA, but it was not as successful as the HOF and BHOF processes in producing the desired outcomes. The comparison of these outcomes is shown in Table 3.

**Table 3: Comparison of results of STGCN with other models**

*The table shows comparison of STGCN model with all 5 models/methods used.*

|  |  |
| --- | --- |
|  | **Accuracy (%)** |
| MHI | 10.00 |
| MEI | 25.00 |
| PCA | 45.00 |
| **ST-GCN SL** | **61.04** |
| HOF | 70.00 |
| BHOF | 85.00 |

### Keypoint based Sign Language Translation without Glosses

[4] SLR research identifies the unique grammar of sign language, which is different from spoken language and has the drawback of being difficult for non-disabled individuals to understand. So let's talk about the problem of directly translating spoken words in a sign language video. To achieve this, we provide a unique keypoint normalization method based on the skeleton point of the signer and rigorous normalization of these points. It enhanced performance by employing a particular normalization strategy depending on body parts. To solve this, a stochastic frame selection method that enables both frame augmentation and sampling simultaneously is applied. There is a social environment where various people should be recognized and given equal chances as the value of an equal society has grown. Deaf individuals, however, struggle to communicate with non-disabled people not just in their daily lives, but also in situations where they need to be informed.

Instead of Sign English Translation (SLT), which instantly converts sign language video to spoken language, Continuous Sign Language Recognition (CSLR), which recognizes sequential glosses, is the focus. It is hard to provide meaningful interpretation alone using CSLR due to the differences between the syntax of sign language and the grammar of spoken language. In contrast to a translation based on sign language grammar, which is workable even without gloss, a new method for translating directly from spoken language to video must be used. We thus try SLT using Neural Machine Translation (NMT)-based methods. In this work, we use the GRU-based Seq2Seq model.

Sign language is a visual language that serves as the primary language of the deaf. A visual language, sign language uses several complementary channels to convey information. The signer's facial expressions, lip movements, upper body movements, and hand gestures are all important nonverbal cues. This paper proposes a technique for translating sign language based on the movement of the signer. To more accurately determine the signer's bodily movement, we employ keypoint-based SLT. The keypoint-based SLT of the signer requires frame processing. Depending on the signer's angle and placement in the frame, the keypoint can take on a wide range of values. To get around and generalize the differences, the keypoint vector should be normalized. Furthermore, whereas conventional keypoint-based sign language translation models fix the length of the sign language image using the frame sampling technique, the NMT model requires a fixed-length keypoint vector to perform SLT. In this process, if the video is too short, there is a higher chance that crucial frames may be missed, losing information from the video. When the time is set to be very long, memory use increases even when a large number of frames are still integrated. We offer a straightforward yet efficient solution to these problems. First, a normalization approach based on the separation between each keypoint is applied. We propose a stronger normalizing technique that modifies normalization according to the body part. The term "Customized Normalization" was used. Second, we describe a probability-based frame selection method for adjusting the sign language video's runtime. As a result, it is feasible to raise the priority of the key frame while lowering the importance of the less significant portion of the sign language video. Additionally, a technique that uses both the augmentation and the sampling method concurrently and is based on the duration of the video is presented, and it corresponds to the length of a dynamic video frame. "Stochastic Augmentation and Skip Sampling (SASS)" was the name we gave it.

To determine the feature value of the video, the signer is retrieved. By focusing less on the signer and more on the video's backdrop, this method maximizes the effect on the signer while still collecting enough points from them. The framework proposed by Fang et al. was used. Alphapose can extract essential points more quickly than the Cao, Zhe, et al. OpenPose model by recognizing signers top-down and then doing so from cropped images. We used a pretrained model on a Halpe dataset. The lower body was removed by eliminating 13 key points, leaving us with 123 key points out of a total of 136. Based on the better results obtained in the Ko et al. study when the keypoint of the face was eliminated, 55 key points were used, deleting 68 detailed facial keypoints.

This was resolved utilizing a broad strategy that is applicable to different datasets while fixing the input value's length. Additionally, we emphasize how crucial the hand-moving frame is to the sign language movie and provide a strategy for keeping it. We propose an augmentation strategy that 5 accentuates the central frame of the image except for this since the frames at the start and end of the movie contain less hand movement for most signers. Additionally, a sampling technique that preserves the key frame is used when the duration of the video is quite long. In this study, we present an augmentation and sampling approach that takes into account the differences in frames for each movie by combining these two methods. "Stochastic Augmentation and Skip Sampling (SASS)" is how it is known.

To show the effectiveness of the technique, experiments were done on two separate datasets: RWTH-PHOENIX-Weather 2014 T. KETI A Korean sign language movie comprising 105 phrases and 419 words is available in the KETI collection (HD). Ten people signed the petition, and the video was taken from two different angles. The KETI dataset, however, lacks a test set, thus we randomly partitioned it into three equal halves with an 8:1:1 ratio. For train, development, and test, the video splits are 6043, 800, and 801, respectively. These films were divided into 30 fps frames. The KETI dataset's absence of gloss made it easy to convert to spoken language.

**Table 4: Results of Keypoint based Sign Language Translation without Glosses**

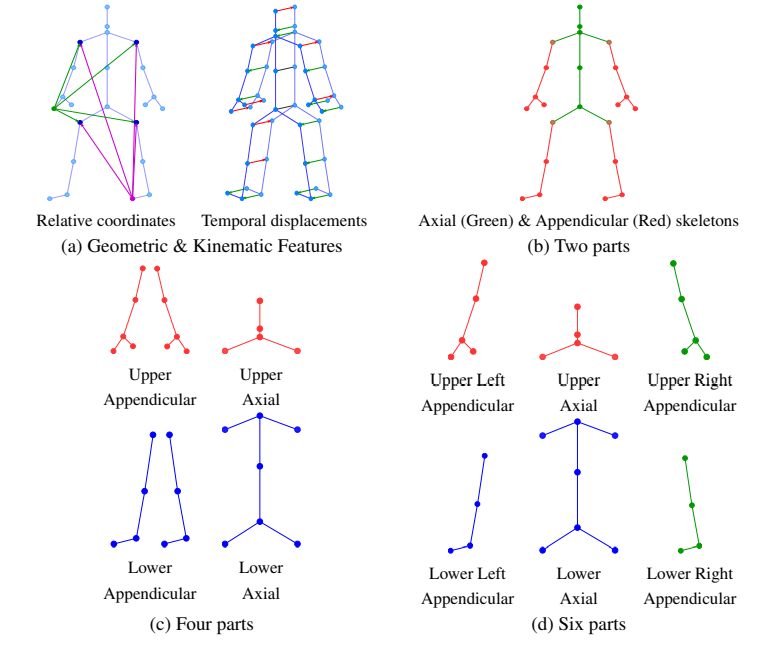
*First table shows (TOP) Comparison of SLT Performance by Sampling Method*

*(Bottom) Comparison of ST/T Performance by Augmentation Method*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **KETI** | | | **RWTH-PHOENIX-Weather 2014 T** | | |
| **BLEU-4** | **ROUGE-L** | **METEOR** | **BLEU-4** | **ROUGE-L** | **METEOR** |
|  |
| **Skip + Random** | 84.24 | 84.75 | 84.80 | 11.5 | 27.63 | 28.68 |  |
| **Stochastic + Random** | 83.92 | 84.65 | 84.57 | 10.16 | 26.77 | 28.38 |  |
| **Stochastic + Stochastic** | 83.5 | 83.93 | 83.95 | 10.83 | 27.04 | 28.7 |  |
| **SASS**  ***(this paper’s)*** | 84.39 | 84.85 | 85.07 | 13.31 | 24.72 | 25.85 |  |

### Part-based Graph Convolutional Network for Action Recognition

[5] Recognizing human behaviors in videos is required for comprehension. RGB, depth, and skeleton video modalities all give distinct sorts of information for comprehending human motions. Skeletal models give 3D joint-positions, which have a higher degree of detail than RGB or depth. The recent publication of numerous multi-modal datasets has attracted a lot of interest in activity detection from the Skeletal model. To learn high-level characteristics from any graph structure, graph convolutions have been utilized. Graph convolutions, a technique used in modern activity recognition from the Skeletal model, portray the complete skeleton as a single graph. Nevertheless, it makes sense to consider the human skeleton to be a combination of several biological parts. Understanding the importance of each body component and their interactions across space and time may be possible with a representation based on body parts. We describe a model for identifying actions in S-videos that uses a unique part-based graph convolution approach. The model outperforms a model's complete skeleton as a single graph in terms of recognition. At each vertex, current models for skeletal activity recognition use 3D coordinates as features. Action recognition may benefit from the use of geometric features like relative joint coordinates and motion traits like temporal displacements.

****Optical flow aids in the detection of activity from RGB movies, while the Manhattan line map aids in the generation of a 3D design using just one picture. Skeletal action recognition has already made use of geometric and kinematic properties. In our part-based graph convolution model, for strongly influence, we use a motion feature that encodes temporal displacements at each vertex and a geometry feature that encodes relative joint coordinates.

**Figure 2: Geometric Features for motion recognition**

*Above figures (a) to (d) show visual representation of keypoints & features*

*extracted from videos for motion recognition*

Some solutions were:

* Creation of a universal segment graph convolutional network (PB-GCN) that can be applied to distinguish actions in S-videos and learnt for any graph with well-known attributes.
* To improve recognition performance, geometric and kinematic information are used instead of 3D joint positions at each vertex.
* Surpassing the current state of the art on the difficult benchmark datasets NTURGB+D and HDM05. Figure 1 displays an overview of our representation and signals.

Graph convolutions over parts are intended to record high-level characteristics of components and comprehend their interrelationships. Different components are recognized in a Deformable Part-based Model, and relationships between them are learnt by deforming the connections amongst them. Graph convolutions on a subgraph discover its attributes, and an aggregate across subgraphs learns their relationships.

Spatio-temporal graphs are used to depict S-videos. Matching joints in each segment are temporally connected to account for the time dimension. The Spatio-temporal graph for the torso across five frames. By expanding concepts from section 3.2, we offer an overview of convolution formulation for our Spatio-temporal graph by adapting the select-assemble-normalize (PATCHY-SAN) presented by Niepert et al. We refer the reader to for a more in-depth understanding. Spatial convolution on each partition, aggregate the convolved partitions using Fagg, then perform temporal convolution on the graph created by aggregating the partitions. In practice, we spatially convolve each partition individually for each frame, aggregate them at each frame, then conduct temporal convolution on the aggregated graph's temporal dimension.

SGD was used as the optimizer and train for 80 (NTURGB+D) / 120 epochs (HDM05). The starting learning rate is set to 0.1, and all tests are done on a cluster of four Nvidia GTX 1080Ti GPUs. The batch size has been set to 64. A validation set is used to complete the learning rate decay schedule (configured to decay by 0.1 at epochs 20, 50, and 70 for NTURGB+D, and at epoch 80 for HDM05). In accordance with the graph-based strategy, no augmentation is conducted for any of the tests. We execute ablation investigations on the large-scale NTURGB+D dataset and then compare the state-of-the-art on both HDM05 and NTURGB+D using our model's optimal configuration.

**Table 5: Results of Part-based Graph Convolutional Network for Action Recognition**

*Top table shows performance with number of parts  
Bottom table shows performance with various signals for best & worst*

*number of parts*

|  |  |  |
| --- | --- | --- |
| **#Parts** | **Accuracy** | |
| **CS** | **CV** |
| One | 79.4 | 87.9 |
| Two | 80.2 | 88.4 |
| Four | **82.8** | **90.3** |
| Six | 81.4 | 89.1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Signals** | **Accuracy** | | | |
| **#Parts=1** | | **#Parts=4** | |
| **CS** | **CV** | **CS** | **CV** |
| **Jloc** | 79.4 | 87.9 | 82.8 | 90.3 |
| **DR** | 83.6 | 87.7 | 84.6 | 88.4 |
| **DT** | 84.3 | 91.6 | 85.4 | 92.6 |
| **DR|DT** | **85.6** | **91.8** | **87.5** | **93.2** |

### Generative Multi-Stream Architecture for American Sign Language Recognition

[6] In earlier studies, researchers have offered supplemental data from other sources at the expense of extra hardware, which is supplied in streams to get over this restriction and improve performance. With the goal of enhancing feature richness without running the risk of impracticability, Gurrapu and Olson suggest a generative multi-stream architecture that does not require extra hardware. Additionally, they add the compact Spatio-Temporal Residual Block to the traditional C3D (3-D convolutional model). On the FASL-RGB dataset, the performance of our rC3D model is comparatively like that of the top C3D residual variation architecture ie. the pseudo-3D models.

The main contributions of this paper can be summarized as follows:

* For ASL machine translation, it suggests a three-stream paradigm that makes use of RGB, Generative Depth, and Motion.
* To dynamically generate depth data in real-time, it develops and constructs a generative adversarial network.
* It provides a residual form of the deep 3-dimensional convolutional network model (C3D).

ASL signals for the following terms are featured in 1600 films in the Frederick American Sign Language (FASL) dataset: alarm, call, lock, movie, no, off, on, rain, reminder, set, sports, today, tomorrow, weather, yes, and nothing. This dataset was created using clips that were 3–10 seconds long and were recorded using an Intel RealSense Depth Camera. This dataset contains RGB-mapped depth data. Data augmentation was necessary to provide architecture with the requisite reliability and invariance because of the magnitude of the FASL dataset. The photos were rotated by using a random rotation filter between [-30°, 30°]. In order to mimic imperfect real-world situations when the camera lens is dirty or has a low resolution, a Gaussian-distributed additive noise filter was also used. Additionally, data augmentation helps to lessen overfitting and makes it possible to create a general ASL recognition model.

Standard machine learning models were built to provide the baseline accuracy for evaluation in order to verify our multi-stream rC3D architecture. In an attempt to decrease data dimensionality for these shallow models, the dataset was pre-processed using principal component analysis with the greatest number of components to preserve over a 98% variance ratio.

The Generative Multi-Stream Architecture used is as follows:

* 3D Model
  + The rC3D is based on the C3D architecture but uses our residual block, also known as rC3D blocks, in place of the convolutional layers. The bottleneck feature in the rC3D residual connection refers to the compression and decompression of the number of filters. The bottleneck was created to encourage the generalization of the learned filters.
  + As opposed to the pseudo-3D block, the residual connections utilize filters that operate concurrently on the spatial and temporal dimensions to combine the two facets of data rather than using a pipeline.
* Generative Depth Model
  + Many cameras lack an integrated depth sensor. In order to do away with the requirement for a camera with a depth sensor, the generative model was designed to dynamically produce the depth information in real time. FastAI used its GAN Learner and Switcher to assist in the model's construction and training.
* Motion
  + The dense optical flow picture shows the amplitude and intensity of the displacement vector fields. The Gunner Farnebacks method and OpenCV are used to construct the pictures based on changes in the spatial and temporal dimensions in adjacent frames. To create the 3D model, the motion picture is sent as a single frame.

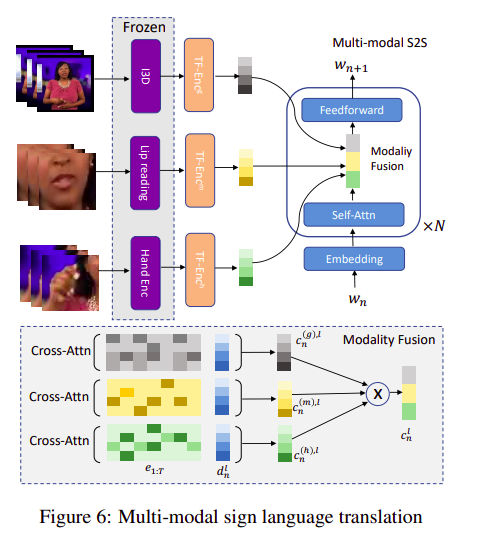
Gurrapu and Olson's techniques outperformed previous models by 0.45% in validation accuracy and 5.53% in variance, achieving 95.62% validation accuracy with a variance of 1.42% from training. They used feature-rich RGB data to explain how to use multiple streams, and also highlighted how using motion and generative skills may improve the effectiveness of ASL sign identification.

### Open-Domain Sign Language Translation Learned from Online Video

[7] The applicability to real-world situations is limited by the fact that most previous work on sign language translation—that is, the translation of sign language videos into sentences in a written language—has mainly concentrated on (1) data collected in a controlled environment or (2) data in a specific domain. Shi and Brentari introduce OpenASL in this research, a sizable ASL-English dataset gathered from internet video portals (e.g., YouTube). They suggest a number of strategies, such as the combining of mouthing and handshape characteristics and sign search as a pretext job for pre-training, to address the difficulties of sign language translation in realistic contexts and without glosses. The suggested methods significantly increase the quality of the translation compared to baseline models based on earlier work in a consistent and significant manner.

The largest publicly accessible ASL translation collection to date, OpenASL has 288 hours of ASL movies in various categories (news, VLOGs, etc.) from over 200 signers. Its video content is mostly gathered from YouTube and other internet video portals. ASL news makes up a sizable percentage of the data. All videos with English subtitles are downloaded by the authors. The National Association of the Deaf's online short-video uploads comprises the remainder of the dataset (NAD). One aspect of the data is the usage of the English translation included in the video's linked subtitles, which eliminates the need for manual annotation.

Method:

* Global image encoding
  + Here, the visual framework is based on the I3D model (Carreira and Zisserman, 2017), a popular 3D CNN utilized for action identification.
* Coarticulated sign pre-training
  + The I3D network is pretrained on pertinent tasks that offer the convolutional layers a quite direct line of supervision than sign language translation. Particularly, WL-ASL, a sizable, isolated sign recognition dataset for ASL, is used to pretrain I3D.
  + To pre-train the visual backbone for translation, it is suggested to look for coarticulated signals in the signing video. A lexical sign identifier and a fingerspelling identifier are both used in the search.
* Local ROI encoding
  + Here, the emphasis is on picking up details for the handshape and mouthing of two local visual modalities. Two large-scale fingerspelling datasets are used to train a fingerspelling identifier, which is subsequently used to retrieve features for the hand ROI, including the handshape feature.
  + To identify the mouthing characteristic from the signer's lips, the model uses an external English lip-reading framework.
* Fusion and sequence modeling
  + See the picture below, where TF-Enc(g), TF-Enc(m), and TF-Enc(h) stand for the transformer encoders for the sequences of global, mouthing, and hand features, respectively, and Cross-Attn(g), Cross-Attn(m), and Cross-Attn(h) are cross-attention layers for the same features.

**Figure 3: Fusion and sequence modeling architecture (****Open-Domain SLT)***Architecture diagram for fusion and sequence modeling consisting of transformer*

*layers & cross attention layers for global, mouthing and hand features*

**Table 6: Translation performance of baseline models and Open-Domain Sign Language Translation**

*Results of Open-Domain SLT evaluated using ROUGE and BLEU (1-4)*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **DEV** | | | | | | **TEST** | | | | |
| **Model** | ROUGE | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 | ROUGE | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 |
| **Conv-GRU** | 16.82 | 16.21 | 9.15 | 5.04 | 3.83 | 17.78 | 15.65 | 7.55 | 4.83 | 3.52 |
| **13D-transformer** | 20.91 | 18.62 | 11.17 | 8.24 | 6.71 | 19.83 | 17.84 | 9.81 | 6.76 | 5.19 |
| **Ours** | 25.31 | 24.35 | 14.94 | 10.72 | 8.39 | 24.83 | 23.87 | 14.08 | 9.90 | 7.54 |

### A Task-aware Instruction Network for Sign Language Translation Enhanced with Data Augmentation

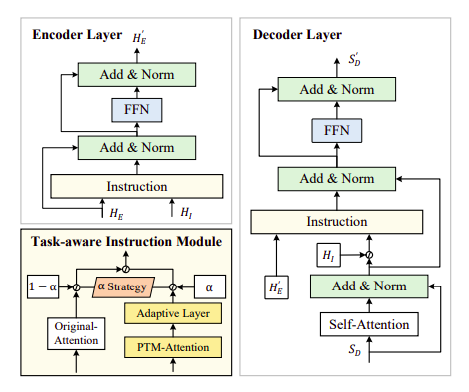
[8] The majority of extant studies place more emphasis on the recognition process than sign language translation. In this study, Cao and Li introduce the instruction module and the learning-based feature merge approach into a Transformer network to create a task-aware training network, called TIN-SLT, for sign language translation. The linguistic skills of the pre-trained model may then be fully exploited and used to improve translation results. Furthermore, they provide a multi-level data augmentation approach to alter the dispersion of the training set's data by investigating the representation space of sign language glosses and the target verbal language.

Like prior techniques, we likewise use a two-step pipeline by first taking a sign language video with the format V = {V1,..., VT} with T frames. The following are the steps:

1. Recognizing V into a sequence G = {g1, . . . , gL} with L independent glosses and then (ii) translating G into a complete spoken sentence S = {w1, . . . , wM} with M words, but we pay more attention to solve step
2. We employ the STMC network, which combines a spatial multi-cue module with a temporal multi-cue module, in empirical studies.
3. Given that the representation space of glosses is substantially less than that of text, after getting the series G of sign glosses, we build a multi-level data augmentation approach to increase the gloss representation space.
4. The core of the design is a task-aware instruction network with a Transformer as the network's foundation and layers of encoders and decoders with the goal of learning the conditional probabilities p(S|G).
5. In order to produce the anticipated sentence S, a linear transform and softmax layer are used after the outputs of the final decoder are sent through a non-linear point-wise feed-forward layer.

Task-aware Instruction Module consists of

1. Encoder
2. Decoder
3. Learning-based feature fusion

**Figure 4: Architecture for Task-aware Instruction Module**

*Task-aware Instruction Module Architecture consisting of encoder and decoder layers*

The PHOENIX-2014-T dataset and the ASLG-PC12 dataset, two well-known benchmark datasets of various languages and sizes, were used for the experiments.

**Table 7: Comparing the translation performance of TIN-SLT**

*Comparison against state-of-the-art techniques on*

*PHOENIX-2014-T and ASLG-PC12 datasets*

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Dev Set** | | | | | | **Test Set** | | | | | |
|  | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 | ROUGE-L | METEOR | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 | ROUGE-L | METEOR |
| **PHOENIX-2014-T Dataset** Evaluation | | | | | | | | | | | | |
| Raw Data | 13.01 | 6.23 | 3.03 | 1.71 | 24.23 | 13.69 | 11.88 | 5.05 | 2.41 | 1.36 | 22.81 | 12.12 |
| Seq2seq | 44.40 | 31.93 | 24.61 | 20.16 | 46.02 | - | 44.13 | 31.47 | 23.89 | 19.26 | 45.45 | - |
| Transformer (Camgoz) | 50.69 | 38.16 | 30.53 | 25.35 | - | - | 48.90 | 36.88 | 29.45 | 24.54 | - | - |
| Transformer Ens (Yin) | 48.85 | 36.62 | 29.23 | 24.38 | 49.01 | 46.96 | 48.40 | 36.90 | 29.70 | 24.90 | 48.51 | 46.24 |
| Transformer (Yin) | 49.05 | 36.20 | 28.53 | 23.52 | 47.36 | 46.09 | 47.69 | 35.52 | 28.17 | 23.32 | 46.58 | 44.85 |
| DataAug (Moryossef) | - | - | - | - | - | - | - | - | - | 23.35 | - | - |
| TIN-SLT(Ours) | 52.35 | 39.03 | 30.83 | 25.38 | 48.82 | 48.4 | 52.77 | 40.08 | 32.09 | 26.55 | 49.43 | 49.36 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ASLG-PC12 Dataset Evaluation** | | | | | | | | | | | | |
| Raw Data | 54.60 | 39.67 | 28.92 | 21.16 | 76.11 | 61.25 | 54.19 | 39.26 | 28.44 | 20.63 | 75.59 | 61.65 |
| Seq2seq | - | - | - | - | - | - | 86.70 | 79.50 | 73.20 | 65.90 | - | - |
| Preprocessed data | 69.25 | 56.83 | 46.94 | 38.94 | 83.80 | 78.75 | 68.82 | 56.36 | 46.53 | 38.37 | 83.28 | 79.06 |
| Transformer Ens (Yin) | 92.98 | 89.09 | 83.55 | 85.63 | 82.41 | 95.93 | 92.98 | 89.09 | 85.63 | 82.41 | 95.87 | 96.46 |
| Transformer (Yin) | 92.67 | 88.72 | 85.22 | 81.93 | 96.18 | 95.95 | 92.88 | 89.22 | 95.95 | 85.28 | 96.22 | 96.60 |
| TIN-SLT(Ours) | 92.75 | 88.91 | 88.51 | 82.33 | 95.17 | 95.21 | 93.35 | 90.03 | 87.07 | 84.29 | 95.39 | 95.92 |

### An Improved Sign Language Translation Model with Explainable Adaptations for Processing Long Sign Sentences

[9] Two neural SLTmodels use effective modules for tokenization. In order to abbreviate lengthy sign phrases without losing information, we first introduce the frame stream density compression (FSDC) approach for identifying and reducing redundant similar frames. Then, an improved design is used to gradually incorporate a temporal convolution (T-Conv) unit and a dynamic hierarchical bidirectional GRU into an NMT module's traditional encoder (DH-BiGRU). The improved component considers temporal tokenization data while extracting more detailed data with efficient resource use.

However, most of the current SLT works only improve the CNN or NMT module separately, resulting in poor connection between the two modules which causes two serious problems:

1. **Poor interpretability:** Instead of taking into consideration SLT's uniqueness, the majority of enhancements depend on widely used methods. SLT's characteristics show that it is a distinct NMT task since the input form is different from conventional spoken language. Therefore, learning from the input form may help us find some fascinating SLT occurrences and enhance interpretability. A string of words is frequently the starting point for a spoken sentence. Although there are semantic connections between words, they are communicated in different ways. A video stream is frequently used as the input while using a sign word. In reality, the video has to be split up into photos with continuous frames. Each video frame may be intuitively matched to a fundamental sign language word component. Unlike spoken language, each sign statement's video frames are continuous, and the order of the signs is coherent. In other words, it's against the rules to arrange any frames in reverse. We found that there are a lot of similar frames nearby that express the same meanings again, leading to redundant information and lengthy phrases. The application of this visual phenomenon to tailor sign language algorithms hasn't been studied, though.
2. **Poor performance for long sentences**: longer phrases have long-distance dependencies, use a lot of resources, and get worse grades. It demonstrates the need for better CNN and NMT modules. However, the work of the inventive NMT module is given less emphasis and the visual CNN module receives more attention. A very essential feature is also the improvement from the standpoint of model interpretation.

Superfluous frames in the temporal vicinity are decreased by using the frame-level frame stream density compression (FSDC) approach, which may compare pixels at the picture level unsupervised. By analyzing the similarity of incoming image frames in the region, it may be seen intuitively as preserving high-density information. Greater amounts of information may be sent inside the constrained window length due to lowered convolution information generating tokenization with a smaller size. Reduced input length for the NMT module is another consequence of fewer input frames. Overall, this optimizer for sign language that may be understood visually condenses long sentences. The NMT module's old encoder is then replaced with a new one that has an improved design to improve the connection between long-sentence video frames. The FairSeq study has resulted in the presentation of a hybrid model. A dynamic hierarchical bidirectional GRU and a temporal convolution (T-Conv) unit are integrated into the model in turn (DH-BiGRU). Before embedding the semantic information in the ensuing deep hierarchical RNNs, it first convolves the input in the temporal domain. The tokenization layer was thought to be the vector representation layer of the dimensionality-reduced frames. In order to improve the correlation between frames in the time domain, 3DCNN/C3D was included in the CNN module. However, it uses more resources and might not always work well if there aren't enough sign language resources available. According to the observation, the NMT module may approach both the function of 3DCNN/C3D and the speed of 2DCNN if it convolves the tokenized sign phrases in the time domain using 2DCNN. Overall, this reduces long phrases in the temporal domain while also hierarchically strengthening the RNN structure. The NMT structure may support both lengthy and brief sentences in this case. Following are some algorithm used for SLT,

* Longer sentences can be accommodated by the SLT model's tokenization-related units with reduced resource consumption and better interpretability.
* Unsupervised FSDC is used to reduce the density of the input frames without losing important data. This method may be used for a variety of video jobs.
* Improved encoder-related units, temporal convolution, and a dynamic hierarchical bidirectional GRU hybrid network (TC-DHBG-Net) are all included in the NMT module for SLT. This network compresses the tokenization layer's useful information in the time domain so that hierarchical GRUs can find semantic information.

The baseline is an attention-based framework that 2DCNN and Seq2Seq successively combine. The parameters of the spatial 2DCNN, an AlexNet, were trained on Imagenet beforehand. Encoders and decoders for Seq2Seq are nonhierarchical GRUs. To appropriately compare results to the standard, all experiments are conducted using the same dataset and GPU environment. All model parameters are consistent by default, except for the deviations mentioned in the article.

The RWTH-PHOENIX-Weather 2014T dataset is the most popular continuous SLT dataset. It was acquired by enlarging the RWTH-PHOENIX-Weather 2014 Corpus German Sign Language Recognition (SLR) dataset. This dataset is of higher quality and contains more data as compared to other SLT datasets. It includes 113,717 words overall, 4,839 vocabulary words, 8,257 video segments, and 947,756 frames. Each video matches a sentence that has been translated. Even though the dataset includes a sign language gloss corpus, our model was trained without using the glosses to determine the meaning and order of the signs.

Based on the findings of the baseline analysis and our prior experience, we established a few crucial hyperparameters. For both the encoder and decoder, we use GRU as the recursive module, with 1,000 hidden units in each recurrent layer.

The Bahdanau mechanism, one of the several attention processes used by the current basic systems, is the finest. It is important to note that the transformer does not perform well in the SLT dataset due to its small size, while doing well in many NMT tasks.

Our model can reduce the quantity of training data by 9.3% and outperform the baseline on the sign-to-text translation task by up to 1.5+ BLEU-4 score when compared to the current state-of-the-art baseline. Additionally, we carried out many comparison and ablation tests as well as a qualitative analysis of translation performance.

### Word-level Deep Sign Language Recognition from Video: A New Large-scale Dataset and Methods Comparison

[10] Serving as a fundamental building block for understanding sign language sentences, the word-level sign recognition task itself is also very challenging:

* The combination of body movements, hand gestures, and head positions largely determines the meaning of signals, and tiny differences may lead to different interpretations. The hands' posture is the only distinction between the signs for "dance" and "read."
* There are hundreds of different signs that are often used and have a large vocabulary. In contrast, there are just a few hundred categories for tasks like gesture recognition and action recognition. The scalability of recognition algorithms is significantly constrained by this.
* The same sign may be shared by both nouns and verbs with the same lemma. Small-scale datasets that are now accessible do not adequately capture these distinctions.

To learn a useful ASL recognition model, the training data must have a sizable number of classes and training instances. We first compile a large-scale vocabulary of ASL word-level signs and their associated annotations because existing word-level datasets do not offer this. Furthermore, only monocular RGB-based videos are downloaded from the Internet in order to employ the least amount of technology possible for sign identification. As a consequence, the trained sign recognition models may be used in a range of settings without the need for specialized tools like depth cameras or coloured gloves. Additionally, individuals frequently sign in frontal viewpoints when they engage with one another. We only collect videos with signers in near-frontal viewpoints in order to produce a high-quality large-scale dataset.

Many deep learning algorithms were tried for word-level sign detection based on 2D human posture and holistic visual appearance based on WLASL. By retraining the VGG backbone and GRU as a representation for convolutional recurrent networks, we provide a starting point for appearance-based methods. A 3D convolution network baseline that performs better than the VGG-GRU baseline is also shown. It is based on fine-tuned I3D. In order to use human postures as input features for pose-based approaches, we first extract them from real films. By using GRU to simulate the temporal movements of the postures, we provide a starting point. GRU may not properly take use of the spatial relationship between body key points since it specifically only captures temporal information in posture trajectories. A novel pose-based model temporal graph convolutional network (TGCN) that simultaneously captures temporal and spatial correlations in posture trajectories is presented in this study as a result of this. Our results reveal that both the pose-based technique and the appearance-based approach achieve equivalent classification performance on 2,000 words, reaching up to 62.63% accuracy.

The three main components of these sign recognition methods are categorization, temporal-dependency modeling, and feature extraction. Previous attempts represented static hand postures using hand-crafted features, including SIFT-based, HOG-based, and frequency domain features. Then, using Hidden Markov Models, the temporal correlations in video sequences are modeled (HMM). Additionally used to account for differences in sequence lengths and frame rates is dynamic time warping (DTW). Support Vector Machine (SVM) and other classification techniques are used to determine the signs that correspond to the matched words.

The WLASL database, a large-scale word-level ASL database, was created to overcome the issues with sign recognition. Only RGB-only videos are included in our dataset, making it possible to swiftly and affordably apply the algorithms we train on to actual issues. In order to make it simpler to evaluate future efforts, we also provide a set of benchmarks based on cutting-edge sign recognition methods.

A substantial signer-independent ASL dataset was produced using two main Internet sources. First off, there are several websites that instruct sign language, such ASLU and ASL-LEX, and they include an ASL signal lookup facility. The mappings between glosses and signals on such websites are accurate as the movies there were examined by experts before uploading. Another useful tool for learning ASL is YouTube instructional videos. We only utilize movies whose subtitles are true to the gloss of the sign. We access 68,129 films with 20,863 ASL glosses from a total of 20 websites. A signer makes one sign (maybe several repeats) in each video, which features a nearly frontal view and diverse backgrounds.

Videos were retrieved with lengths ranging from 0.36 to 8.12 seconds after acquiring all the annotations for each video; the overall average video length is 2.41 seconds. The videos' median intra-class standard deviation is 0.85 seconds. In terms of a gloss's sample number, we rank the glosses in descending order. We do tests on datasets with various vocabulary sizes to gain a better understanding of the challenges of the word-level sign identification problem and the scalability of sign recognition systems.

The resolution of each original video frame was increased such that the bounding box of the subject had a diagonal measurement of 256 pixels. For training VGG-GRU and I3D, a 224224 patch from an input frame was randomly chosen, and a horizontal flipping with a probability of 0.5 was applied. It should be noticed that the same cropping and flipping processes are applied to all of the video frames rather than being done frame-by-frame.

**Table 8: Comparing the translation Word-level Deep Sign Language Recognition**

*First table shows Top-1, top-5, top-10 accuracy (%) achieved by*

*each model (by row) on the four WLASL subsets.*

*Second table shows Top-10 accuracy (%) of 13D (and Pose-TGCN when trained*

*(row/) and tested (column) on different WLASL. Subsets*

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Method** | **WLASL100** | | | **WLASL300** | | | **WLASL1000** | | | **WLASL2000** | | |
| **top-1** | **top-5** | **top-10** | **top-1** | **top-5** | **top-10** | **top-1** | **top-5** | **top-10** | **top-1** | **top-5** | **top-10** |
| **Pose-GRU** | 46.51 | 76.74 | 85.66 | 33.68 | 64.37 | 76.05 | 30.01 | 58.42 | 70.15 | 22.54 | 49.81 | 61.38 |
| **Pose-TGCN** | 55.43 | 78.68 | 87.60 | 38.32 | 67.51 | 79.64 | 34.86 | 61.73 | 71.91 | 23.65 | 51.75 | 62.24 |
| **VGG-GRU** | 25.97 | 55.04 | 63.95 | 19.31 | 46.56 | 61.08 | 14.66 | 37.31 | 49.36 | 8.44 | 23.58 | 32.58 |
| **I3D** | **65.89** | **84.11** | **89.92** | **56.14** | **79.94** | **86.98** | **47.33** | **76.44** | **84.33** | **32.48** | **57.31** | **66.31** |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **WLASL100** | | **WLASL200** | | **WLASL1000** | | **WLASL2000** | |
| **I3D** | **TGCN** | **I3D** | **TGCN** | **I3D** | **TGCN** | **I3D** | **TGCN** |
| **WLASL100** | 89.92 | 87.60 | - | - | - | - | - | - |
| **WLASL200** | 88.37 | 81.40 | 86.98 | 79.64 | - | - | - | - |
| **WLASL1000** | 85.27 | 77.52 | 86.22 | 74.25 | 84.33 | 71.91 | - | - |
| **WLASL2000** | 72.09 | 67.83 | 71.11 | 65.42 | 67.32 | 64.55 | 66.31 | 62.24 |

### Sign Language Text to Speech Converter using Image Processing and CNN

[11] Sign language is regarded as the primary means of communication for those who are deaf or hard of hearing. This disability has created many challenges for the affected population in the field of problem-solving and employment and has arisen the requirement for a translating tool for the conversion of ASL (American sign language) to text or speech. The objective is to create a gesture recognition application that will convert sign language to text and later to speech.

This concept accepts hand signs representing English alphabets with the addition of a space gesture for word separation and an ‘Ok’ gesture for sentence termination. A subset of SL (sign language) is finger sign which is used to spell out words in written or spoken English. The model involves the segmentation of finger sign gestures from frames of images used as input. The following points denote the proposed architecture and methodology:

* Generating and storing multiple ‘hand’ gestures signs by using Image processing methods converting RGB to grayscale
* Using the output of the first process and with the help of threshold, converting int to binary image to make it system readable.
* Smoothing the image with Gaussian and Median blur techniques and identifying the edges of the hand gesture by using the methodology of contour.
* The output will be stored in a database and with the help of TensorFlow and Keras, the data pass through the CNN algorithm for training and testing of our model

The captured image is first preprocessed using OpenCV libraries and converted from RGB to HSV (hue saturation and value). Then the process of smoothening and contouring segments the image into smaller chunks which reduces recognition complexity and makes it easier for the model to trace out patterns from this image dataset. The model is trained on the basics of convolution neural network, which has proven to be quite efficient in the field of image recognition and can detect patterns through pixelated configurations. The following softmax-based loss function is used in the CNN model:

(1)

(2)

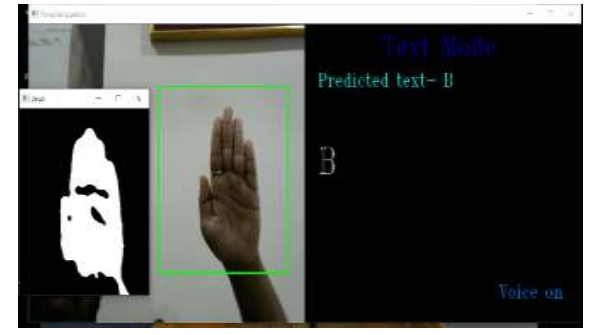
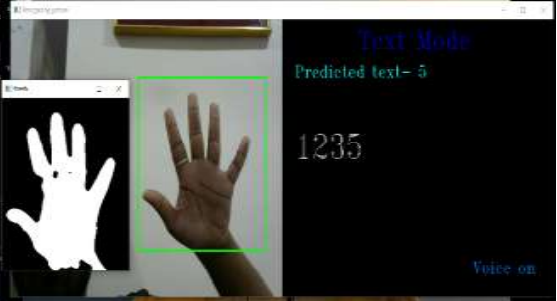
N = total number of training examples

C = total number of classes

The function takes the feature as vector z and decreases its value to a vector of [0,1] valued real numbers. The dataset used for the model was 1200 images of each English alphabet and 0-9 numeric digits with the facility of the pipeline for generating and adding additional datasets

The CNN layer in total has 4 different layers which capture image features, non-linear interactions, and features plus hand-gestures. The architecture in total has 3 groups of 2 Convolutional layers with a max pool and dropout layer next to it and then 2 fully connected layers. The intermediate results then conclude to the final output by passing to an additional dropout and final output layer.

* Convolution layer scans the source image by applying a filter of a certain dimension to extract features for classification. For every specific region, a dot product is calculated with pixel values and weights values are defined. The filter meets the image 3 times for every layer separately. After the first convolution, features are samples for the process to be repeated to identify sub-features of an image which then ultimately identifies spot features for classification.
* The pooling layer then reduces the complexity of computation by decreasing the overall spatial size of a problem
* The fully connected layer then takes the compressed results of the pooling layer and classifies the image into labels. Each value represents the probability of a particular feature that belongs to a specific label. With the process of backpropagation, the weights are adjusted and configured according to the pattern of the dataset for the training of the model to classify unseen images.

The following are the results for the Sign Language translator tool:

**Figure 5: Result of Sign Language translator tool**

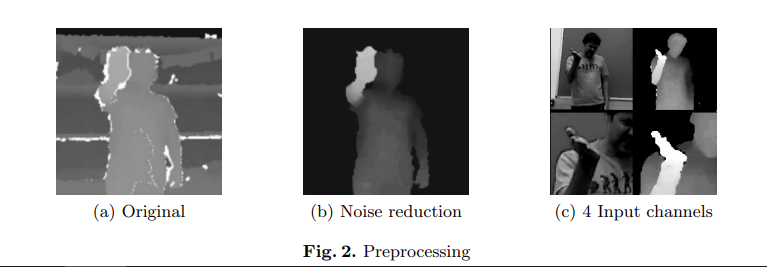
*Example results of the translation tool successfully identifying the*

*alphabet B (left) and the number5 (Right)*

By training and testing the dataset on the model, an accuracy of 85% is achieved. The project has demonstrated the use of convolution layers and Neural networks as a combination for the classification of American sign language.

### Sign Language Recognition Using Convolutional Neural Networks

[12] The communication gap between the deaf and hearing majority is a major barrier and in dire need to be dealt with. The purpose of this project is to create a recognition system using the Microsoft Kinect, convolutional neural networks (CNNs) and GPU acceleration. The two main steps towards establishing a SLR is feature extraction from frame sequences (resulting in a vectorial representation of features called as descriptors) and the classification of these actions to translate them to proper words. This project does feature extraction through Convolution Neural Network (CNN) and classification through Artificial neural network (ANN).

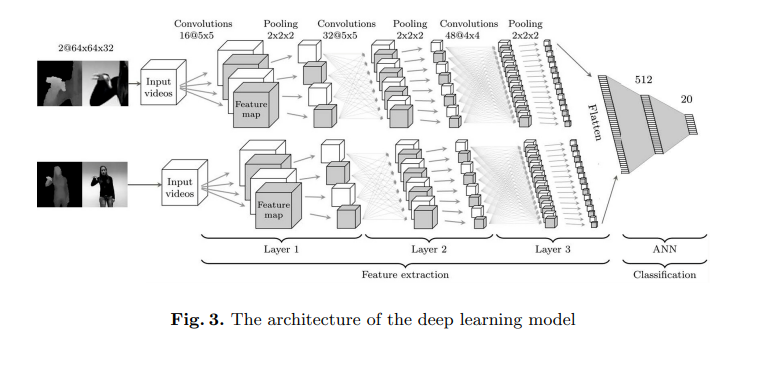
The starting step is the preprocessing of the videos recorded by Microsoft Kinect is cropping the uppermost hand and body using the provided joint information. This produces four video samples of dimension 64x64x32 which passes through thresholding, background removal using the user index, and median filtering for noise reduction. The outcome is as follow:

**Figure 6: Result of background removal samples**

*Outcome of thresholding, background removal using the user index,*

*and median filtering for noise reduction*

After preprocessing the data, the result is sent as an input to CNN models which are proven to be efficient deep learning models to detect images. CNN applies discrete convolutions to the picture and uses filter values as trainable weights. Multiple filters are used for each channel, and coupled with the neuron activation functions, they produce feature maps. Then for the pooling method, the max-pooling mechanism retains the largest value in a feature map's local neighborhood. This specific architecture uses 3-D convolution for video input data where the filter travels in x, y and z dimensions in a video frame in a pooling layer.

The proposed architecture consists of two CNNs (for hand features and the other for the upper body) where each CNN itself is 3 layers deep. After crossing through the series of convolution and pooling, the resulting data entered the ANN phase for classification. The ANN uses one hidden layer and concatenates the result of the CNN layer to classify the input. For this architecture, local contrast normalization (LCN) and the activation function of ReLU is used.

**Figure 7: Architecture of deep learning model**

*Detailed architecture of the entire deep learning model consisting of*

*2 CNNs (3 layered) entering an ANN (1 hidden layer)*

Dropout and data augmentation are the key methods for reducing overfitting during training where the CPU does real-time data augmentation. The image data goes through 10% zooming, 3-degree rotations, -5-pixel spatial translations in both axes, and (-4) temporal translations. We also use the gradient descent method with a momentum coefficient of 0.9, the learning rate of the model at 0.003 with a 5% decrease after each (epoch) and a batch size of 20. The weights are initialized randomly in CNN with a normal distribution and the value of sigma (σ) is kept at 0.04, 0.02 for the ANN.

The dataset used for this model is Track 3 (Gesture Spotting) of the CLAP14 (ChaLearn Looking at People 2014) which comprises 20 different Italian gestures. For our testing purpose, we used 6600 gestures in the CLAP14 development set: 4600 for the training set and 2000 for the validation set. An additional 3543 samples were also used for testing purposes. For predicting the start and finish frames of each gesture in video recordings, we used temporal spacing.

**Table 9: Validation Results of SLR using CNN**

*Validation results with an accuracy of 91.70% (8.30% error rate) and*

*test data accuracy of 95.68% with a 4.13% false positive rate*

|  |  |  |
| --- | --- | --- |
|  | **Error rate (%)** | **Improvement (%)** |
| **Tanh units** | 18.90 |  |
| **ReLU** | 14.40 | 23.8 |
| **+dropout** | 11.90 | 17.4 |
| **+LCN (first 2 layers)** | 10.30 | 13.4 |
| **+data augmentation** | 8.30 | 19.4 |

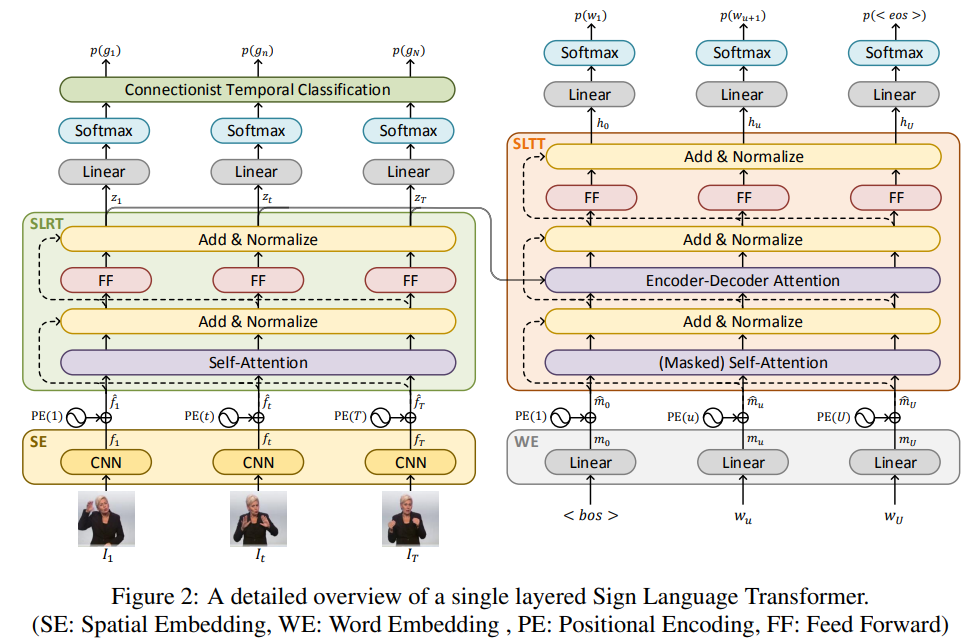
### Sign Language Transformers: Joint End-to-end Sign Language Recognition and Translation

[13] A mid-level sign gloss representation (successfully recognizing the individual signs) has been demonstrated in earlier work on sign language translation to significantly increase translation performance. For this project, a unique architecture is presented based on transformers that simultaneously learn “Continuous Sign Language Recognition and Translation” while being end-to-end trainable. The goal of a Sign Language Translator is usually to extract text sentences from video clips of someone performing continuous signs. The focus of this work has been placed upon recognizing sign-glosses since spoken language grammar is different from sign language. Therefore, a Spatio-temporal machine translation task is to produce spoken language words from sign language videos. The following sub-task are currently unsolved:

* Detecting sign sentences (sentence segmentation) from a continuous sign language. Speech-based recognition or translator identifies pauses or ‘silent regions’ to be sentenced segmentation indicators.
* After sign segmentation, identifying the subject of the sentence is a challenging task for the machine model which can be assisted by the recognition of sign gloss or other linguistic features.
* Generation of spoken language sentences after the information is extracted from sign language stream. To learn the correspondence between sign glosses and their spoken language translations, computational linguists have used text-to-text statistical machine translation models.

Research finding favors mid-level sign gloss (sign-gloss-text) representation in improving sign language translator performance significantly. The model first recognizes sign glosses from a video stream input using the CSLR method (tokenizing layer) and passes it to NMT for text to text conversion for generating spoken language sentences. This methodology creates less number of frames for the model to process as compared to sign to text and provides an additional guidance for understanding sign sentences.

The paper proposes a novel Sign language Transformer approach where results accuracy is solely based on correct sign to text recognition. The main objective is to learn sign language recognition from spatial representation of sign language in a comprehensive way. To predict sign gloss sequences, we present the Sign Language Recognition Transformer (SLRT), an encoder transformer model trained with a CTC loss. SLRT learns Spatio-temporal representations from spatial embeddings taken from video input. This representation in the next step becomes the input of the Sign language Translation Transformer (SLTT), which decodes only one particular word at a time to generate proper English sentences.

**Figure 8: A detailed overview of a single layered SLT**

*A detailed overview of a single layered Sign Language Transformer.*

*(SE: Spatial Embedding, WE: Word Embedding, PE: Positional Encoding, FF: Feed Forward)*

The following is an overview of the model workings to process a Sign language video and generate textual results:

* First, the **Sign Language Transformers** transform sign videos to sign gloss and language sentences using the concept of conditional probabilities p(G|V) and p(S|V) of generating a sign gloss sequence. Modeling these probabilities requires CSLR to get sign gloss and an NMT model to learn sign gloss-text translation. Another group works simultaneously by focusing on sign language translation with no intermediate base used (SLRT decoder).
* We embed video frames of sign language through Spatial embedding approach and use CNN models to pass on the images as input. The division of Spatial embedding is as follows:

mu = WordEmbedding (wu)

ft = SpatialEmbedding(It)

Here, Ft is the non-linear frame level spatial representation derived from a CNN, and Mu is the embedded representation of the spoken language word Wu.

* The result of positional encoding spatial embedding is then used to train transformer encoder models. The first input of SLRT is a self-attention layer to configure contextual relation between video frames. This process precedes a non-linear point wise feed forward layer where the normalization process takes place. We then use a linear projection layer followed by a softmax activation to extract frame level gloss probabilities, p(gt|V), from spatio-temporal representations.
* Lastly, our goal is to sign up for spoken language generation, which is done by SLTT. We begin by adding the sentence initiator, <bos> to the target spoken language sentence S. The positionally encoded-word embeddings are then extracted. Embedding is then passed on to the masked attention layer and merged with the SLRT attention layer to learn the mapping between input and output. Like SLRT, the SLTT model passes output through normalization and then the feed-forward layer. We calculate the cross-entropy loss for each word using conditional probabilities for evaluating the performance of our model

Evaluation of the paper's proposed model was done through the **“****PHOENIX 14T dataset”** dataset which contains sign language clips, gloss annotations, and translations. The corpus contains 1066 distinct signs from unrestricted continuous sign language from 9 different signers, translation of which is provided in German spoken Language. According to the results of our initial trials, pre-trained features on sign data overlook the generic Image-text with no intermediate approach. Furthermore, we have demonstrated that learning translation and recognition together enhanced performance on both tasks. More significantly, by directly translating spoken language words from video representations, we have outperformed the text-to-text translation findings, which were designated as a virtual upper bound.

The loss function applied to depict efficiency and performance analysis:

ℒT =

ℒ = R ℒR + T ℒT

### Sign Pose-based Transformer for Word-level Sign Language Recognition Networks

[14] Sign Language (SL) is composed of both manual and non-manual components. The former component consists of arms and hand gestures while the latter deals with detailed motions like facial expressions. SLR is classified into two levels: isolated SLR (also known as word-level in the literature), which classifies recordings of individual signals into glosses, and continuous SLR, which identifies whole utterances. For this specific work, our focus will be on isolated SLR, and we will base our evaluation on the Transformer model.

The basic methodology is divided into two methods/models which are:

1. SPOTTER (Sign Pose based Transform ER): detecting signing phase
2. Transformer: classify pose sequence into a sign gloss

The following is the description of the working components of our model (Sign language recognition):

* Preprocessing:

A Standard Pose Estimation algorithm from Vision API gathers gesture poses from video frames. 54 body landmarks, including five heads and 21 for each hand, are extracted as body joints excluding the head (ears, eyes, nose, etc.). Because all the markers are two dimensions, we get a 108-dimensional position vector for each frame. An exception of no person standing in a frame has marked a zero which is dealt with later

* Augmentation:

The following spatial augmentations are applied to the skeletal data during training to avoid overfitting and increase the model's generalizability (parameters are selected randomly from a uniform distribution).

1. **In-plane rotation:**

Each frame's joint coordinates are all randomly rotated up to 13 degrees in the following way:

f rotate (x,y) = ( (x-0.5) cos 𝜽 - (y - 0.5) sin 𝜽 + 0.5,

(y - 0.5) cos 𝜽 + (x - 0.5) sin 𝜽 + 0.5),

1. **Squeeze:**

The horizontal sides of each frame are compressed. The joint coordinates are then recalculated with respect to the new plane using the following formula:

f squeeze(x) =

1. **Perspective transformation:**

To mimic recording the sign video with a tiny tilt, the joint coordinates are projected onto a new plane with a spatially defined center of projection. The new plane is then defined by shrinking both the vertical edge (height) at each of its neighboring corners and the width at the desired side.

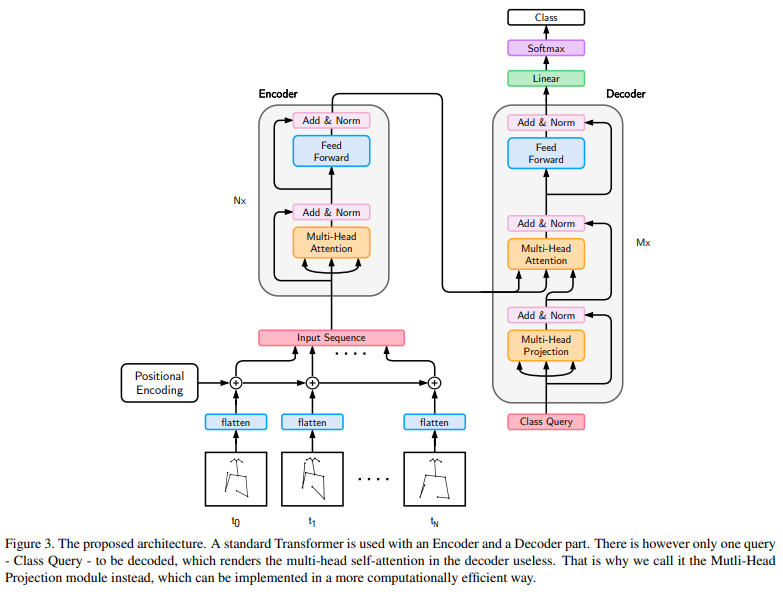
1. **Sequential joint rotation:**

Both arms' joint coordinates are repeatedly transferred, and the upcoming landmark is slightly rotated relative to the present one. This stimulates very small variations in each execution of a sign that have no impact on the semantic meaning.

* Normalization:

To project the body landmarks onto the signature space, our normalizing approach draws on research in SL linguistics on the usage and delimitation of space. We define the signature area based on the head metric, which is suggestive and relatively independent of the individual body structure and distance of the signer from the camera.

* Proposed Architecture:

Following is the summarization of our model:

**Figure 9: Standard Transformer is used with an Encoder and Decoders**

*Summarized architecture of a standard transformer consisting of*

*encoder and decoder layers*

This system receives a pre-processed body position as input. The Word Level American Sign Language (WLASL) dataset and the LSA64 dataset were the two datasets used to create this model (3200 video samples of 64 unique glosses from Argentinian SL). This data undergoes positional encoding, which gives us the input sequence that is sent to the encoder levels of the transformer. Then, much like in the original transformer, it passes via a self-attention module and a two-layer feed forward network. One command is handled by the transformer's decoder as an input, and it goes via a Multi-Head Projection module. The original multi-headed attention module's multiple parallel projection heads are maintained. The last linear layer of the internal hidden dimension then concatenates and processes these projections. The decrypted class query is sent into a linear layer that contains an equal number of neurons as the number of classes, and softmax activation is used to predict each class's confidence. This process is repeated after passing through another multi-headed module.

Finally, the following table shows the result of the dataset on this model:

WLASL:

**Table 10: Results Sign pose-based transformer on WLASL**

*Results of WLASL100 and WLASL300 on our model (SPOTER) and 6 other models*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **App.** | **Pose** | **Backbone** | **WLASL100** | **WLASL300** |
| **I3D** | ✔ | ✗ | ✔ | 65.89 | 56.14 |
| **TK-3D** | ✔ | ✗ | ✔ | **77.55** | **68.75** |
| **Fusion-3** | ✔ | ✗ | ✔ | 75.67 | 68.30 |
| **GCN-BERT** | ✗ | ✔ | ✗ | 60.15 | 42.18 |
| **Pose-TGCN** | ✗ | ✔ | ✗ | 55.43 | 38.32 |
| **Pose-GRU** | ✗ | ✔ | ✗ | 46.51 | 33.68 |
| **SPOTER** | ✗ | ✔ | ✗ | **63.18** | **43.78** |

LSR:

**Table 11: Top1 average recognition accuracy**

*Top1 average results of LSA64 dataset on our model (SPOTER) and 6 other models*

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **App.** | **Pose** | **Accuracy** |
| **LSTM + LDS** | ✔ | ✗ | 98.09 ± 0.59 \* |
| **LSTM + DSC** | ✔ | ✔ | 99.84 ± 0.19 \* |
| **DeepSign CNN** | ✔ | ✗ | 96.00 |
| **MEMP** | ✔ | ✗ | 99.06 |
| **ELM + MN CNN** | ✔ | ✔ | 97.81 |
| **I3D** | ✔ | ✗ | 98.91 |
| **SPOTER (Ours)** | ✗ | ✔ | **100.00 ± 0 \*** |

## Literature Review Summary Table

This table contains the summary of various past research papers from 2014-2021.

#### *Table 12: Work done on Sign Language Recognition*

*The table contains the summary of various past research papers on*

*Sign Language Recognition from 2014-2022.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Author** | **Year** | **Model** | **Description** | **Results (Output)** | **Dataset (Input)** |
| 1. | Songyao Jiang, Bin Sun | 2021 | SAM-SLR,  SL-GCN,  SSTCN,  3D-CNN | Action identification relies on skeletons to separate the topic from background variation using RGB-based techniques, neural networks, and hand key points. | In 2021, SAM-SLR performed best in both RGB (98.42%) and RGBD (98.53%) tracks. | AUTSL Dataset |
| 2. | Prem Selvaraj, Pratyush Kumar | 2021 | SL-GCN,  ST-GCN,  LSTM,  Transformer | Pretraining corpus on Indian-SL with over 1,100 hours of posture data, pretrained models on this corpus for all 3 pre-training techniques, and models fine-tuned for 4 distinct SLs on top of the pretrained model. Library includes pose-based datasets for the 6 SLs. | Average accuracies are 69.38, 73.47, 77.4, and 80.69 for LSTM, Transformer, ST-GCN, and SL-GCN respectively. | AUTSL,  CSL,  WLASL, GSL,  LSA64,  DEVISIGN,  INCLUDE |
| 3. | Cleison Correia de Amorim, David Macedo | 2019 | ST-GCN | The first step is to estimate the skeletons of the people in the input movies and build space-time graphs based on those graphs. After that, more ST-GCN convolution layers are used to create progressively more advanced feature maps for the shown graphs. They are then sent to a classifier to determine the appropriate action. | In terms of sign recognition, the framework was able to reach an average accuracy of 61.04% during the epochs 330 and 380. The average top-5 accuracy for each was 86.36. | ASLLVD-Skeleton-20 |
| 4. | Youngmin Kim, Minji Kwak | 2022 | NMT, GRU, | By using an augmentation approach that 5 accentuates the middle frame of the picture since the frames at the beginning and conclusion of the movie are those with less hand movement, the length of the input value must first be fixed without any motion frame loss. Using these two techniques, a sampling and augmentation technique that accounts for the variations in frame rates between each video. "Stochastic Augmentation and Skip Sampling (SASS)" is the name we give to this approach. | Among keypoint-based SLT models, the performance of the KETI-dataset is the best. In the PHOENIX dataset, the BLEU score was also the highest. | Halpe dataset,KETI, RWTH-PHOENIX-Weather 2014 |
| 5. | 2018 | Kalpit Thakkar, P J Narayanan | PB-GCN, LSTM | PB-GCN is used to identify actions in movies, and to improve identification performance, geometric and motion data are used in place of 3 dimensional joint positions at each vertex. | With NTURGB+D'Dataset got an accuracy of 93.2% and HDM05(smaller dataset) got an accuracy 88.19% | NTURGB+D and HDM05 |
| 6. | Dom Huh,  Sai Gurrapu | 2020 | C3D,  rC3D | In order to increase feature richness without running the danger of becoming unworkable, a generative multi-stream architecture, avoiding the need for extra hardware, and a compact spatiotemporal residual block were added to the traditional 3d convolutional framework. | approaches outperformed previous models in terms of validation accuracy and variance by 0.45% and 5.53%, respectively, achieving 95.62% validation accuracy with a variation of 1.42% from training. | FASL Dataset |
| 7. | Bowen Shi, Diane Brentari | 2022 | Coarticulated sign pretraining,  Local ROI encoding,  Fusion and sequence modeling | OpenASL is a sizable collection of ASL to English translation data based on web signing videos with subtitles. a model based on local feature fusion and coarticulated sign pretraining. | -------------- | OpenASL Dataset |
| 8. | Yong Cao1, Wei Li | 2022 | TIN-SLT,  Transformer Network, | By fusing the instruction mechanism and the learning-based feature fusion technique into a Transformer network, a task-aware instruction network known as TIN-SLT for sign language translation is created. | Testing on the difficult benchmark datasets PHOENIX-2014-T and ASLGPC12 reveal that the technique performs 1.65 and 1.42 BLEU-4 points better than the previous best solutions, respectively. | PHOENIX2014-T and ASLG-PC12 |
| 9. | Jiangbin Zheng ,Zheng Zhao, | 2020 | T-Conv, GRU (DH-BiGRU), BLEU, FSDC algorithm | The first frame-level frame stream density compression (FSDC) approach enables unsupervised pixel comparison at the image level by reducing duplicate frames in the temporal domain. Second, we replace the standard encoder in the NMT module with a better design to further improve the linkage among lengthy phrase video frames. | Optimized absolute values of BLEU in comparison to the baseline following application of the FSDC method. Both models' peaks are located close to the 95% cutoff. | RWTH-PHOENIX-Weather 2014T |
| 10. | Dongxu Li , Cristian Rodriguez Opazo | 2020 | VGG-GRU, TGCN | Retraining of the VGG backbone and GRU, a representation for convolutional recurrent networks, are available for appearance-based approaches. To use GRU to simulate the temporal movements of the gestures in pose-based approaches, first extract human postures from the original movies and utilize them as input features. | On 2,000 words, both the pose-based technique and the appearance-based approach produce categorization performance up to 62.63%. | Purdue RVL-SLLL ASL Database, Boston ASLLVD and RWTH-BOSTON-50 . |
| 11. | Mangesh B , Mayur K, Rujali P | 2020 | CNN | Generating grayscale images of sign language alphabets using image processing and converting it to binary images. The smoothing of the images Gaussian median blur techniques and finally sending the output to CNN (supported by Keras and TensorFlow) for recognition | 85% accuracy was achieved on the test dataset after training the CNN model | Local dataset consisting of 1200 images of sign language alphabets |
| 12. | Lionel Pigou, Sander Dieleman, Pieter-Jan Kindermans, and Benjamin Schrauwen | 2014 | CNN + ANN | Video samples from Kinect passing by threshold and background noise removal (Preprocessing). The data is then fed to CNN for features extraction and the output goes to ANN (with one hidden layer) for classification | Validation- 91.70% accuracy (8.3% error rate)  Testing- 95.68% (4.13% error rate) | CLAP14 with 20 different Italian gestures and 6600 gestures. |
| 13. | Necati Cihan Camgoz, Oscar Kollerq, Simon Hadfield and Richard Bowden | 2020 | Transformer model (Recognition and Translation) + Sign2Gloss2Text model + NMT model | A novel transformer-based architecture which focuses on extracting sign-glosses from videos and inputting it with CTC loss to the transformer model for accurate results. | The recognition doubled the performance of some previous techniques used before for sign language translation. Different comparisons between Rnn and our model where the Sign2(Gloss+Text) model outperformed really well. | PHOENIX14T dataset |
| 14. | Matyas Bohacek,  Marek Hruz | 2022 | Transformer model  (SPOTER method + Traditional method) | Word-Level Sign Language Translation on Transformer model on Isolated dataset. The data is preprocessed to extract 2D positional landmarks of human body pose, which then goes through normalization and augmentation and then transformer encoding process to generate an output for classification. | From a 100 gloss subset from isolated dataset, the model recognized 63.18% (which is a 5% improvement) -From 300 samples - 43.78% recognition and from LSA-64 a 100% accuracy | LSLSA64 dataset (3200 videos) and WLASL dataset (4-subsets) |

# 

# Software Requirement Specifications

This chapter describes all modules of requirements and design in clear English text along with the necessary diagram and figures. It describes functional requirements, design constraints, and other factors necessary to provide a complete and comprehensive description of the requirements for the software.

## Functional Requirements

Following are the functional requirements of our model that fully describe the external behavior of the system:

* Take input from the camera
* Draw keypoints and save them as .npy files
* Detect signs in the input
* Recognize and label signs according to the model training
* Display continuous output to the screen

## Hardware and Software Requirements

The hardware and software requirements that will be required to develop and deploy this project are listed in this chapter.

### Hardware

The hardware requirements necessary to deploy this project are as follows:

* GPU should be capable of ML and DL training and testing.
* Enough cache memory to support a large dataset.
* RAM 8GBs and above
* CPU 8th Generation or above, and 4 cores or above
* SSDs are preferred over HDDs
* Windows 7 minimum
* 1080p webcam

### Software

The following software was utilized to write, compile and execute the code of this project:

* Jupyter
* Anaconda
* Stable browser: chrome or Firefox
* GitHub for collaborative work and version control

#### Libraries

* MP Holistic
* Open CV
* Tensorflow
* Keras
* Numpy
* Matplotlib
* Sklearn
* Os
* Time

## Use Cases

This section lists our single use case from our model that represents the most significant, central functionality of our prototype.

### Sign Language Recognition

The following table describes the details of our prototype use case.

**Table 13: Use case Details Table**

*Table provides the details of the Sign Language Recognition prototype use case*

| **Name** | | Recognition | | |
| --- | --- | --- | --- | --- |
| **Actors** | | User | | |
| **Summary** | | The user (through the camera) will perform an action representing a sign language word that will be translated to gloss by our model. | | |
| **Pre-conditions** | | The user must have a camera (external or internal) to run the prototype. | | |
| **Post-Conditions** | | The user will receive the exact translation of the sign they are inputting. | | |
| **Special Requirement** | | none | | |
| **Basic Flow** | | | | |
| **Actor Action** | | | **System Response** | |
| 1 | User acts out a sign representing a word | | 2 | System detects the action and displays the corresponding translation on the interface |
| **Alternative Flow** | | | | |
| **Actor Action** | | | **System Response** | |
| 1 | User isn’t showing any sign | | 2 | Nothing is displayed on the interface |

# Proposed Approach and Methodology

The following section covers our proposed approach towards Sign Language Recognition using MediaPipe Holistic [15] and LSTM model. This approach will also cover the “real time” factor and use faster machine learning techniques for detection and recognition. For input, we would fetch frames from the camera using OpenCV library, detect keypoints from frames and send concatenated coordinates information to train and test our LSTM. The data must be preprocessed and labeled afterwards.

## Dataset

We proposed our own dataset with the purpose of facilitatating research in sign language recognition and translation. It consists of 240 unstructured video files in MP4 format, with each file being 6.5 seconds long and containing a sample of one of 20 different American Sign Language (ASL) signs.

The dataset contains 12 video samples for each sign and a total of 240 video samples. The signs included in the dataset are: Yes, No, Help me, Please, Thank you, Sorry, Goodbye, Hello, Name, I love you, Where? When?, Why?, How?, Eat, Drink, More, Finished, Bathroom, and Friend. Each sign is recorded against two different backgrounds (whiteboard and library) with the same clothing for each background. There are six samples with a whiteboard background and six with a library background for each sign, with three performed by a male signer and three performed by a female signer.

The dataset includes metadata about the signers, such as their gender, height, age, and clothing. The male signer is 5 feet 8 inches tall and 22 years old, while the female signer is 5 feet 5 inches tall and 22 years old. The videos were captured using an iPhone 11 Pro's back camera in landscape mode. Each video has HD resolution, 30 frames per second, and pixel dimensions of 1920x1080. The camera was positioned 4 feet 10 inches above the ground, and the signer was positioned 3 feet 11 inches away from the camera.

This dataset can be used to train and evaluate machine learning models for sign language recognition and translation. The metadata about the signers, including their height and age, as well as the camera position and distance, and variable backgrounds may also be useful for analyzing the impact of these variables on the accuracy of recognition and translation models.

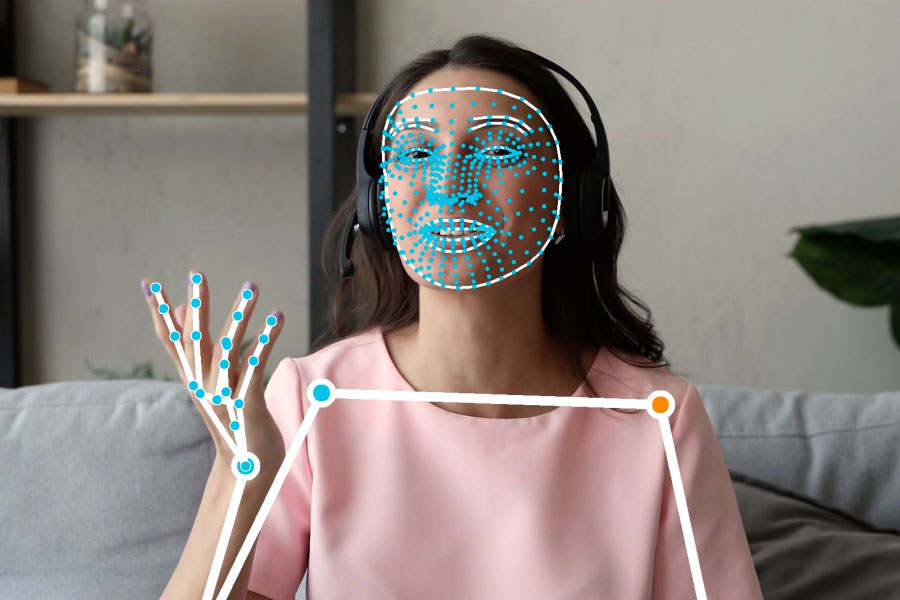
## Key Points Using MediaPipe Holistic

Following steps will be followed for extracting keypoints using MP holistic:

1. We will be creating two variables (one for media pipe holistic and one for media pipe drawing utilities). MediaPipe holistic variable will be to download and leverage the holistic model and drawing utilities will handle the drawing of our keypoints.
2. Then a function for MediaPipe detection will be created, with preprocessed image and holistic model as input parameters. Preprocessing process includes converting the image from BGR to RGB and setting the writable feature as false. The detection process will be completed through (**model.process()**) function that takes in the video frame as input. After detection, image settings will be converted to default.
3. Afterwards, we’ll be creating a drawing utility function to join the landmarks properly (forming a stick figure image). The Drawing utility variable will be used to draw landmarks of face, pose, left and right arms. This function will use a connection map that will show which landmarks (representing a specific body part) should relate to each other to form a figure. E.g., right shoulder to right elbow and so on. Optional formatting option (extra parameters) can be used for better representation.
4. Finally, a function will be required to access our webcam using OpenCV library through a loop (frame looping). The input method will be to capture frames from camera input in a sequential manner and show them one after another to the user (hence displaying a video). Users can exit the camera pop-up by pressing any button. The purpose of capturing frame by frame input is to process each individual frame separately and pass it on to the media pipe detection function (along with a holistic model) to detect landmarks. The resulting frame will be passed to the drawing utility function to join lines between landmarks and form a proper stick figure of the person. Using Cv2, the user can view his landmarks through the popup video output.

### MediaPipe Holistic

Separate models for stance, face, and hand components are individually integrated into the MediaPipe Holistic pipeline and are each optimized for their respective fields. However, the input to one component is not appropriate for the others due to their disparate areas of expertise. For instance, a smaller, fixed resolution video frame (256x256) is used as input for the pose estimation model. But the image quality would be too low for precise articulation if the hand and facial regions were cropped out of that image and given to the appropriate models. As a result, MediaPipe Holistic is a multi-stage pipeline that processes the various regions according to the suitable image resolution for each one.

With BlazePose's pose detector and subsequent landmark model, we first assess the human pose. Then, we create three regions of interest (ROI) crops for each hand and the face using the inferred pose landmarks, and we use a re-crop model to enhance the ROI. Then, using task-specific face and hand models, we crop the full-resolution input frame to these ROIs in order to estimate the appropriate landmarks. Finally, we combine every landmark with every position model landmark to produce all 540+ landmarks.

**Figure 10: MP Holistic Demo**

*The diagram shows the demo of live keypoints detection*

We employ a tracking strategy like that used for standalone face and hand pipelines to speed up the identification of ROIs for faces and hands. The estimation from the previous frame is used as a reference to the object region on the current one on the assumption that the item doesn't change noticeably between frames. The detector must delocalize the object in the image if the tracker loses it during rapid movements. Pose prediction is used by MediaPipe Holistic as an additional ROI (on every frame) before the pipeline's response time is sped up in response to rapid movements. By preventing a mismatch between the left and right hands or other body parts of one person in the frame with those of another, this also helps the model maintain

semantic consistency throughout the body and its parts.

Additionally, because the posture model's input frame's resolution is so low, the face and hand ROIs that are produced are still too imprecise to serve as a guide for re-cropping those regions, which need a precise input crop to stay lightweight. We employ simple face and hand re-crop models that act as spatial transformers and only consume around 10% of the inference time of the comparable model to close this accuracy gap.

The pipeline is constructed as a MediaPipe graph that renders using a dedicated holistic renderer subgraph and uses a holistic landmark subgraph from the holistic landmark module. A position landmark module, hand landmark module, and face landmark module are all used internally by the holistic landmark subgraph. Check them out for implementation information.

### Open CV

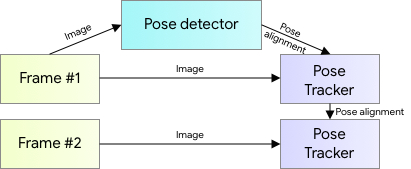
For video capturing, we are using the OpenCV library to access our camera input. For that we need to create a Video Capture object and we pass a camera id (hardware index). After that, **.read()** functionality returns a frame which is converted from BGR to RGB for better display. A continuous loop keeps extracting and displaying frame by frame image, where each frame is passed through landmark detection and connecting landmarks by lines.

## Extract Keypoint Values

The result of the detection function of MP contains information about our landmarks (coordinates of our pose right and left hand and face). We need to extract them (or concatenate x, y, z and visibility value) in the form of a NumPy array for structuring our data. Each landmark data is then concatenated to form key points of hands, face and pose and flattened all together to represent coordinates of a single frame.

## Collect Keypoint Values

Detection is divided into 4 parts: hand (left and right), face and pose (that covers the torso of the user).

Detection is done through BlazePose, which uses machine learning to detect 33 2-D landmarks (for pose) of the body in a single video frame.

**Figure 11: BlazePose Detection Model**

*The diagram shows human pose estimation pipeline overview.*

Each landmark contains two-dimensional information (x, y coordinates) and visibility.

The pose detector sends the results to the Pose tracker for drawing lines between landmarks.

### Models

#### Landmark Model

To create a total of 543 landmarks, MediaPipe Holistic uses the stance, face, and hand landmark models from MediaPipe Pose, MediaPipe Face Mesh, and MediaPipe Hands, respectively (33 pose landmarks, 468 face landmarks, and 21 hand landmarks per hand).

#### Hand Recrop Model

When the pose model's accuracy is insufficient, we run an additional, lightweight hand re-crop model that serves as a spatial transformer and takes about 10% more time to infer than the hand model in cases where the resulting ROIs for hands are still too inaccurate.

## Preprocessing

To preprocess our dataset, we started by manually editing and cropping the recorded videos to standardize their length to 6 seconds and 18 frames. We then extracted 30 frames from each sample at equal intervals between each frame. To avoid using frames during the initial pause in the videos, we ignored the first 18 frames. We then extracted every 6th frame from the remaining frames to ensure we included frames depicting movement and avoided redundancy. Next, we passed each frame through the MediaPipe Holistic model, which outputs the keypoint positions for various landmarks such as the left arm, right arm, pose, and face. We then saved the keypoint positions as NumPy arrays in the system for training and testing purposes. This preprocessing step allows us to generate a standardized input format for our LSTM model to effectively learn from the temporal patterns in the sign language gestures.

## LSTM Neural Network

Long Short-Term Memory (LSTM) networks are a type of Recurrent Neural Network that can learn order dependence. By default, the LSTM may hold data for a very long time. It is employed in the processing, prediction, and categorization of time-series data. In contrast to standard feed-forward neural networks, LSTM contains feedback connections. It can manage whole data streams (such as speech or video) in addition to single data points (like photographs).

Four neural networks and many memory cells, which are arranged in a chain pattern, make up the LSTM. A cell with an input gate, an output gate, and a forget gate make up a typical LSTM unit. These three gates regulate the information flow into and out of the cell, and the cell retains values for arbitrary time periods. Time series of indeterminate length may be categorized, analyzed, and predicted with the LSTM algorithm.

* **Input Gate:** It determines which of the input parameters should be applied in order to modify the memory. The sigmoid function decides whether to pass through 0 or 1 values. The tanh function also adds weight to the acquired data, ranking its significance on a scale from -1 to 1.

.

* **Forget Gate:** It identifies the information that has to be deleted from the block. A sigmoid function is used to decide this. It examines the prior state (ht-1) and the content input (Xt) for each number in the cell state Ct-1 to create a number between 0 (omit this) and 1 (keep this) for each cell.

* **Output Gate:** The output is determined by the input and memory of the block. It is determined whether to pass through 0 or 1 data using the sigmoid function. And which numbers can pass between 0 and 1 is determined by the tanh function. Additionally, the tanh function gives the provided values weight by evaluating their relevance on a scale from -1 to 1 and multiplying it by the sigmoid output.

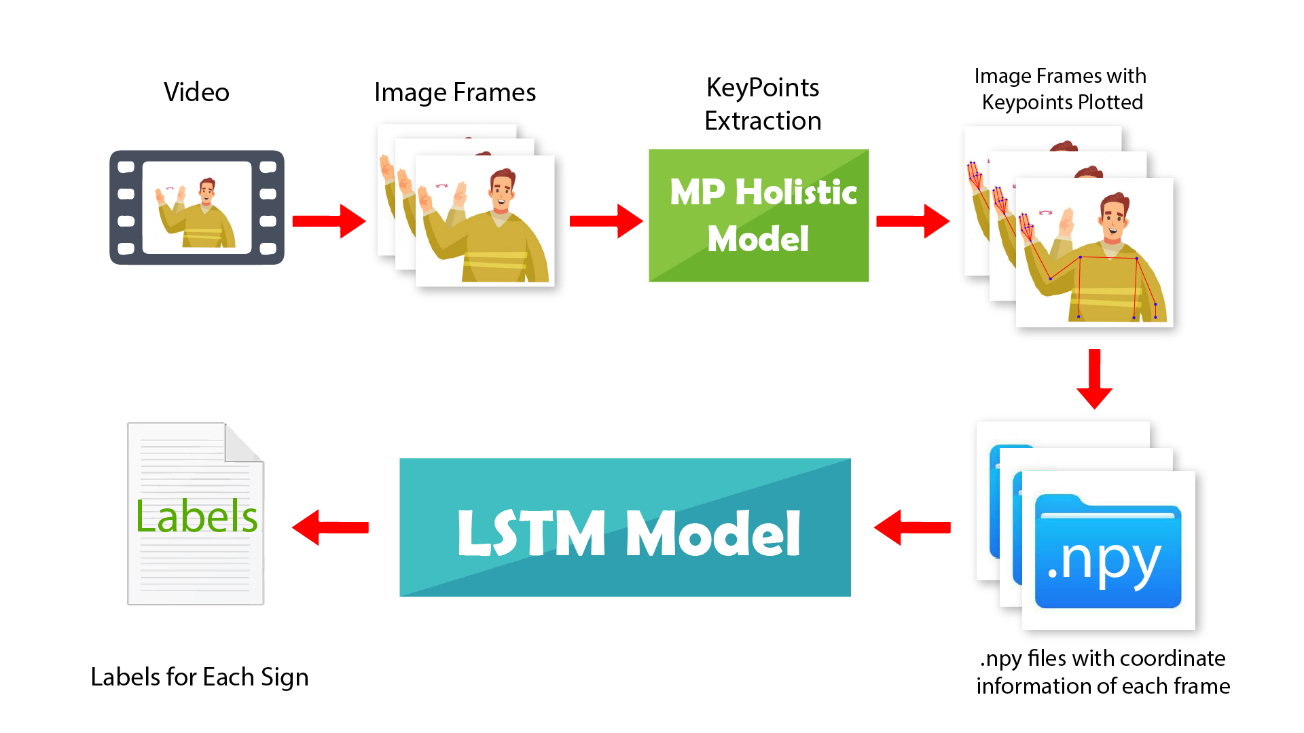
.

The cell state, which is meticulously managed by structures known as gates, may be altered by the LSTM by deleting or adding information. An LSTM has three of these gates to safeguard and control the cell state. Information can be passed through gates in a controlled manner. They are composed of a sigmoid neural net layer and a pointwise multiplication step. The sigmoid layer generates values from 0 to 1 that represent the amount of each component that should be allowed to pass. When a value is zero, "nothing" should be permitted to pass, and when a value is one, "everything" should be allowed to pass.

## LSTM and MP Holistic

State-of-the-art models commonly use several LSTM layers followed by a number of LSTM layers. We trained a simple neural network with a similar number of video sequences (or samples) for each word/phrase as the state-of-the-art model. Our initial training was for 3 actions with 30 samples for each making it 90 samples in total. When testing a model that does not use LSTM the accuracy was nowhere near the level of accuracy of state-of-the-art models that was acutely going to be useful. Switching to Media Pipe Holistic Mapping and LSTM Model provides the following advantages:

* **Less Data Required:** We need fewer data to produce a hyper-accurate model because the LSTM model is quite dense.
* **Faster To Train:** The LSTM neural network is much denser. So rather than having around 30 to 40 million parameters, as one would in a simple neural network, we will have less than half a million parameters which means it will be a lot after to train the LSTM model rather than one without it.
* **Faster Detections**: The LSTM neural network is a lot simpler, due to having lesser parameters, which means that it will perform faster when it comes to detecting sign language in real-time.



**Figure 12:** **Flow Diagram of Our System**

*The diagram shows the basic high-level flow of our model from taking input to returning output*

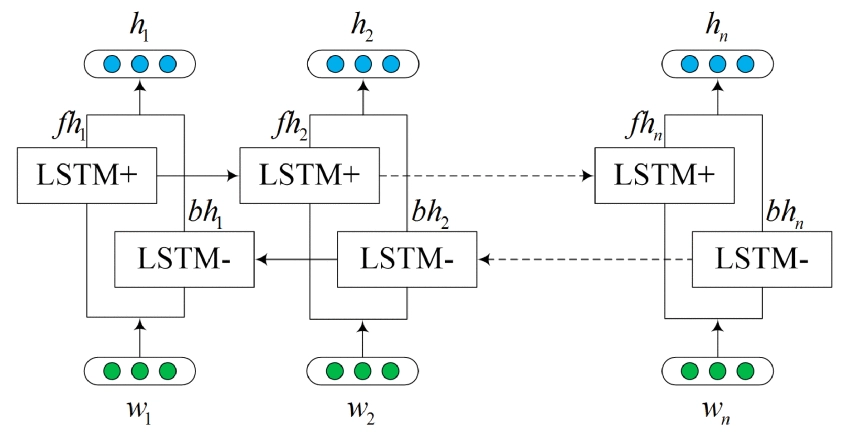
## Bi-LSTM Neural Network

Bidirectional LSTM networks process input sequences by presenting each training sequence in both forward and backward directions to two independent LSTM networks, which are connected to the same output layer. This allows the Bi-LSTM to capture comprehensive, sequential information about all points in a particular sequence both before and after each point.

In other words, instead of encoding the sequence in the forward direction only, we also encode it in the backward direction and concatenate the results from both forward and backward LSTMs at each time step. The encoded representation of each element in the sequence now incorporates information about both the preceding and succeeding elements.

The basic architecture of a Bi-LSTM consists of two separate LSTM layers, one processing the input sequence in the forward direction and the other in the backward direction. These two LSTM layers are then concatenated at each time step before passing the output to the next layer or the output layer. This allows the Bi-LSTM to learn from the entire sequence in both directions and capture long-term dependencies that may be missed by a unidirectional LSTM.

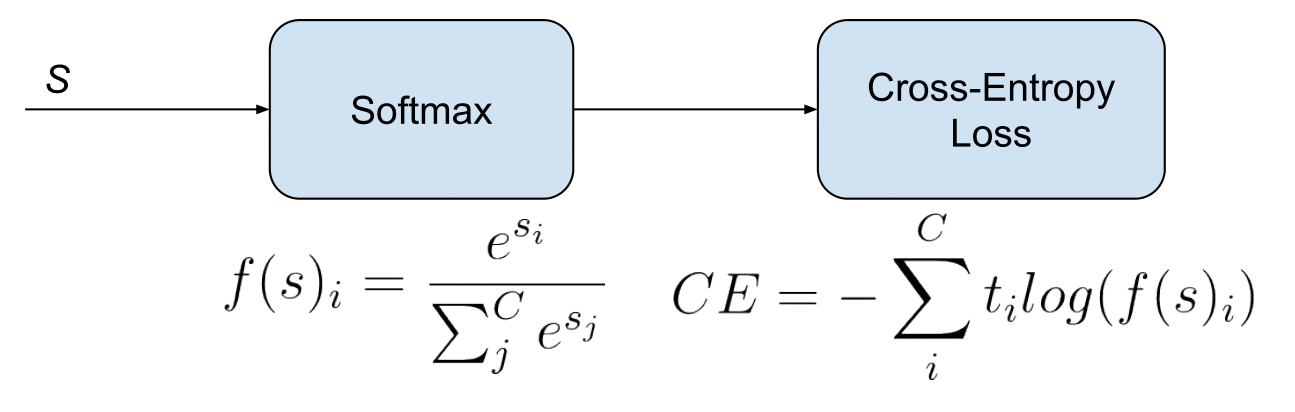
Below is the basic architecture of Bi-LSTM.

 **Figure 13: Bi-LSTM Neural Network**

*Basic architecture of Bi-LSTM Neural Network*

## Categorical Cross-Entropy Loss

Also known as Softmax Loss, it consists of a Softmax activation and a Cross-Entropy loss. If we employ this loss, we will train a CNN to provide a probability over the C classes for each video. It is utilized for multi-class categorization.



**Figure 14: Visual Representation of** **Categorical Cross-Entropy Loss**

*The diagram shows the sequence of formulas used in*

*Categorical Cross-Entropy Loss*

The labels are one-hot in in the specific (and usual) case of multi-Class classification, meaning that only the positive class Cp retains its word in the loss. The Target vector t only has one element, Ti=Tp, that is not zero. Eliminating the summation elements that are 0 because of target labels, we can write:

Where Sp is the CNN score for the positive class.

After defining the loss, we must now calculate its gradient with respect to the CNN's output neurons in order to backpropagate the loss through the network and optimize the specified loss function by optimizing the parameters of the network. The gradient of CE Loss with respect to each CNN class score in s must thus be calculated. The loss terms coming from the negative classes are zero. Since the SoftMax of the positive class is also dependent on the scores of the negative classes, the loss gradient regarding those negative classes is not, however, eliminated. Since the ground truth class Cp's score (sp) is in the nominator, the gradient expression will be the same for all C except for Cp.

After some calculus, the derivative respect to the positive class is:

And the derivative respect to the other (negative) classes is:

Where sn is the score of any negative class in C different from Cp.

## Methodology Conclusion

In this section, we discussed our proposed methodology about sign language recognition and the logic behind using MP Holistic and LSTM and how they work to fulfill our functionality. The proposed structure is designed to give us desired results in real time with higher accuracy.

# Implementation and Test Cases

This chapter contains the elaboration of the implementation that you have done so far. The clear details of the algorithms that were implemented along with the platform used are also described.

## Implementation

Following is the implementation process of our project

### Environment Setup

To implement our LSTM model the first step is to set up the environment. We chose Anaconda Navigator and its Python distribution. Jupyter Notebook was deployed through Anaconda to write and test our code. The second step was to install all the following libraries:

* Tensorflow (2.4.1)
* Tensorflow-gpu (2.4.1)
* Opencv-python
* Mediapipe (MP Holistic)
* Sklearn
* Matplotlib

After installation is complete, we will import all the library features we need for implementation. Tensorflow will be used to import the LSTM model and all the extra requirements we may have to build, train and test our model. OpenCV will be utilized to capture video from our web camera so they can be converted into frames for feature extraction. Mediapipe will then plot whole body key points on these extracted frames. Sklearn is for model training purposes and Matplotlib is for displaying our results on the prototype.

### Preprocessing

Our objective is to gather sample frames from pre-recorded videos to extract key points that will be used as input for our Sign language recognition model.

1. We first import the necessary libraries, such as MediaPipe, OpenCV, and NumPy.
2. Prerecorded dataset, containing videos of length 6 seconds and 18 frames, is used as input.
3. To label the actions in our recognition model, we use an action array that contains 9 separate videos for each label, with each video consisting of 198 frames.
4. We iterate through each input video file using a for loop and process each frame using OpenCV (cv2).
5. To improve efficiency, we filter out frames within each video while looping by skipping the first 18 frames and only taking multiples of 6 (mod 6) to obtain a total of 30 frames. This step eliminates redundant frames and ensures that frames showing movement are included.
6. The media pipe detection method preprocesses the frames and converts them into writable (BGR) images.
7. We then use MP Holistic to detect and join landmarks from the left arm, right arm, pose, and face to extract key points.
8. Finally, we convert the extracted key points into NumPy arrays, preparing them to be passed on to the LSTM model.

### Feature Extraction

As mentioned, before we will make use of MediaPipe Holistic to extract the key points/features from our video frames. This process will begin by first importing the MP Holistic model and its drawing utilities. RGB input will be taken through OpenCV and passed to the MP model by saving in an image variable. The MP model will be passed input frame by frame over which it will draw connections on the following:

* Face
* Pose
* Left Hand
* Right Hand

These connections will be drawn in from coordinate points that will further be joined to each other by straight lines. MP Holistic returns this coordinate information in the form of .npy files that can then be used to train and test the model. This returned information will also be utilized by Matplotlib to plot the key points on the web camera display that OpenCV is providing on the screen in front of a popup. The above process is repeated in an infinite loop until the user quits by pressing a specific key.  The technical details of all the steps are as follows:

1. Create a function for extracting key points that take in the results of the detection function.
2. From the results returned by the detection function, we will concatenate information of each landmark (x, y, z, and visibility value) into a single NumPy array.
3. a NumPy array of arrays (NumPy array containing arrays) in our pose will be gathered. This will be equal to the number of landmarks detected in our pose. (Length of array = no. of landmark)
4. Flatten them into a single NumPy array to pass it to the upcoming model. E.g., if 10 landmarks are detected with 4 values each then the length of the final list will be 40.
5. Repeat it for other landmarks as well (pose, left hand, right hand, and face).
6. If a hand or posture is not detected in a frame, then a NumPy array of zeros will be replaced with length (no of landmarks \* no. of features in each landmark).
7. Finally concatenate all the key points by np. concatenate and pass them face, pose, left and right arm landmarks arrays (as keypoints).

This loop is employed when taking input for dataset collection as well as when the user is testing the model.

### Sign Language Recognition Models

We will use Tensor flow and Keras to export the following:

* Sequential Model that will allow us to build a sequential neural network which will be easy to build up while training and testing.
* LSTM Layer for the Temporal Component
* Dropout Layer to prevent model overfitting.
* Dense Layer which is a normal fully connected layer
* TensorBoard to help us monitor our model's performance as it is training.

#### Architecture 1

Our sequential model will consist of 3 LSTM layers with 64, 128, and 64 LSTM units in each layer respectively. The activation function used is *relu* and the input is 30 frames per prediction with 1662 parameters (key points) per frame. Following these LSTM layers will be 2 Dense layers with 64 and 32 Dense units (or fully connected neural network neurons) respectively, and the activation function *relu*. Then there will be a final Dense layer that will have Dense units equal to the number of labels (actions) which was initially set to 3. The activation function of this final year will be softmax.

Our model will finally be compiled using the Adam Optimizer and Keras Library’s Categorical Cross-entropy Function.

Shape

Description automatically generated with medium confidence

**Figure 15: Sequential LSTM Model Architecture 1**

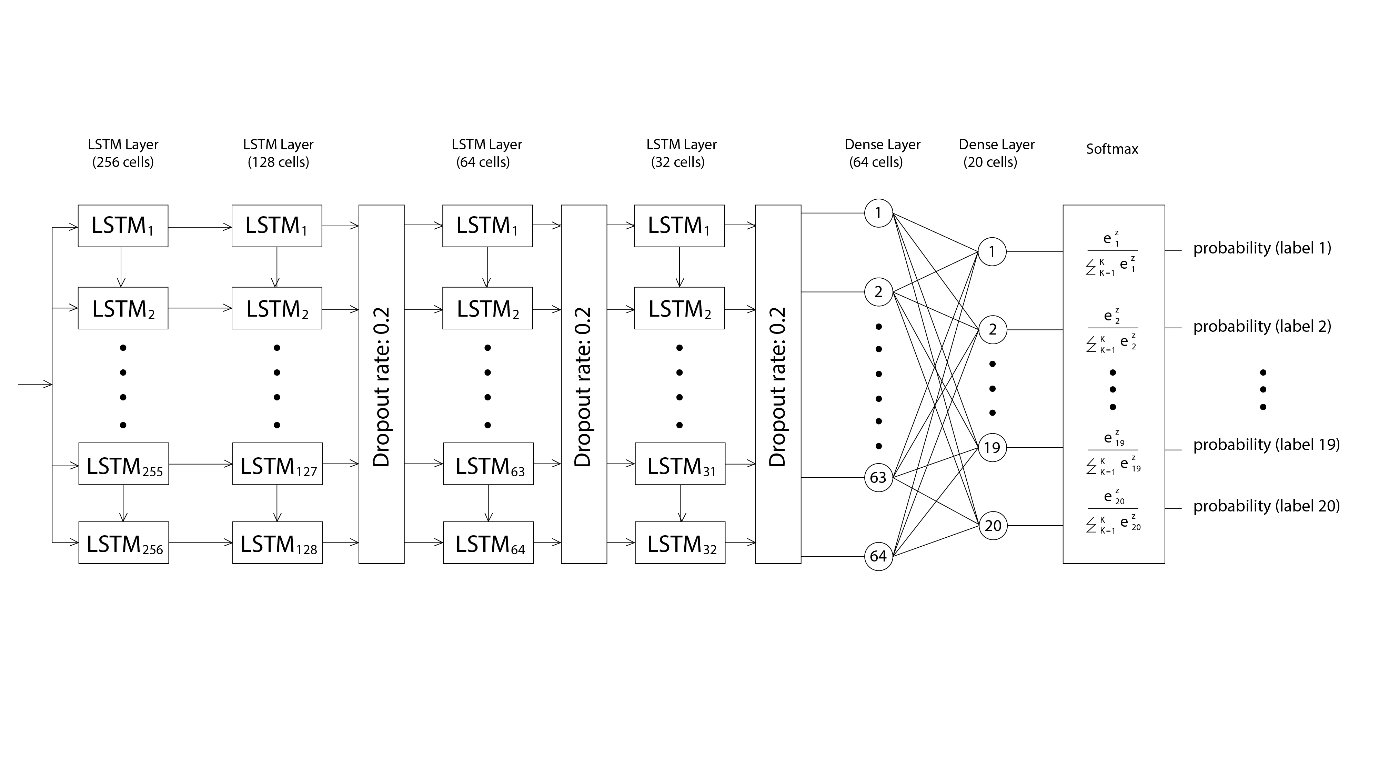
*This diagram represents the architectural details of our sequential model with multiple LSTM and dense layers*

#### Architecture 2

Our code utilizes a Sequential model built using the Keras API. This model architecture consists of a sequence of LSTM layers and dense layers.

The first LSTM layer has 256 units with a hyperbolic tangent (tanh) activation function and an input shape of (30, 1662), where 30 is the number of time steps (frames) we want to consider, and 1662 is the number of features for each frame. The second LSTM layer has 128 units with a tanh activation function. We add a dropout layer with a rate of 0.2 after this layer to reduce overfitting. The third LSTM layer has 64 units with a tanh activation function. We add another dropout layer with a rate of 0.2 after this layer. The fourth LSTM layer has 32 units with a tanh activation function.

We then add two dense layers. The first dense layer has 64 units with a rectified linear unit (ReLU) activation function, and the second dense layer has 20 units with a softmax activation function. The softmax activation function is used because we are performing multi-class classification, and we want the output to represent the probability distribution of each class.

**Figure 16: Sequential LSTM Model Architecture 2**

*This diagram represents the architectural details of our sequential model with multiple LSTM and dense layers.*

## Test Case Design and Description

### Test Case No.1

**Table 14: Test Case No.1**

*Table shows the details about test case no.1*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sign Recognition** | | | | | |
|  | | | | | |
| Test Case ID: | | *1* | QA Test Engineer: | | *Muhammad Usama* |
| Test case Version: | | *1* | Reviewed By: | | *Muhammad Usama* |
| Test Date: | | *13-May-2023* | Use Case Reference(s): | | *Use case 1* |
| Revision History: | | *-* | | | |
| Objective | | *The goal of this test case is to validate the accuracy and usefulness of the prototype's sign language recognition capability. This involves verifying the system's ability to recognize and properly translate sign language motions.* | | | |
| Product/Ver/Module: | | *Part of overall system being build that includes Sign language recognition model that is the core module/ feature of the entire system.* | | | |
| Environment: | | *Hardware: Computer with a functioning camera (internal)*  *Software: Jupyter notebook (and Anaconda) + Windows OS* | | | |
| Assumptions: | | *-* | | | |
| Pre-Requisite: | | *Access to Functional Camera*  *Camera captures actor's upper body posture accurately* | | | |
| Step No. | Execution description | | | Procedure result | |
| 1 | *User performs the sign for the word “Hello” (ASL)* | | | *System detects the sign and out of 20 labels show the highest probability of the label HELLO and concatenates it with results.* | |
| 2 | *User performs the sign for the word “Thank you” (ASL)* | | | *System detects the sign of “Thank you” and shows its probability being the highest and concatenates with results.* | |
| 3 | *User stays idle and don’t show any sign* | | | *System does not detect any sign hence nothing is concatenated with results.* | |
| Comments: The system will be tested in different backgrounds and with different actors to assess its performance. Timing noted to ensure the pace at which the recognition model works | | | | | |
| *Passed* *Failed* *Not Executed* | | | | | |

## Test Metrics

### Sample Test Case Matric.No.1

**Table 15: Test Case Matric.No.1**

*Table shows the details about test Matric.No.1*

|  |  |
| --- | --- |
| Metric | Purpose |
| Number of Test Cases | 1 |
| Number of Test Cases Passed | 1 |
| Number of Test Cases Failed | 0 |
| Test Case Defect Density | 0 |
| Test Case Effectiveness | 100% |
| Traceability Matrix | REQ1 - The system should accurately detect and translate the sign for "hello"  REQ2 - The system should accurately detect and translate the sign for "thank you"  REQ3 - The system should not display any translation if the user performs a gesture that is not recognized as a sign |

## Prototype

Our Prototype aims to develop a real-time human action recognition system that captures and analyzes actors' movements using state-of-the-art computer vision and machine learning techniques. The system uses OpenCV, a widely used open-source computer vision library, to capture and process video streams, enabling it to identify and track the actors in the scene. Simultaneously, it utilizes the MPholistic library to extract and analyze key body pose features, allowing for accurate and efficient recognition of different actions.

The core of our system comprises two primary components: the actor detection and tracking module, and the sign classification module. The actor detection and tracking module employs OpenCV to capture and MediaPipe Holistic to process video streams in real-time. MP Holistic identifies and tracks the actors in the scene by analyzing their body shapes, movements, and postures. It then extracts key body pose features from the actors, such as joints, limbs, and angles, which serve as the basis for recognizing specific actions.

Once the actors are detected and tracked, the sign classification module takes over. The key features extracted are then passed onto our Sign Language Detection Model that. By analyzing these key features, the model can predict the most likely action being performed by each actor.

To visualize and communicate the results, our system displays probability bars for each action label, with the percentages representing the confidence level of the predicted actions. This enables users to quickly and easily understand the system's predictions and the degree of certainty associated with each action.

The highest probability label, representing the most likely action performed by the actor, is concatenated with a string placed above the actor's head in the video stream. This real-time display provides immediate feedback on the system's predictions, allowing users to validate the accuracy and reliability of the action recognition.

## 

# Experimental Results and Discussion

This chapter lists the detailed requirements and setup of our prototype and experiments conducted to test it. Furthermore, analysis and a description of the results are also explained herein.

## Dataset Testing

We partitioned our dataset into training and testing data using a ratio of 3:1. Consequently, our training data comprises 9 samples for each of the 20 signs, resulting in a total of 180 samples. The testing data, on the other hand, consisted of 3 samples for each of the 20 signs, totaling 60 samples.

The results obtained from testing our model on various architectures using the testing data are summarized in the table below.

**Table 16: Testing Data Results**

*Table shows the performance of model on multiple architectures.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Architecture Number** | **3 Labels** | **5 Labels** | **10 Labels** | **20 Labels** |
| **1** | 90.00% | 13.33% | - | - |
| **2** | 80.20% | 20.00% | - | - |
| **3** | - | 93.00% | - | 20.00% |

## Live Testing

After our LSTM model used the above-described dataset to train the weights, it was ready for testing. Testing was done using the same 1080p web camera. The same signers who were used to collect the dataset were used for testing purposes as well. Four experiments were conducted, which are described in detail in the following sections.

### Experimental Setup

In these experiments, a 1080p web camera was employed as the hardware for both dataset preparation and testing purposes.

For each sign, it was tested against two distinct backgrounds: a whiteboard and a library setting. The same clothing was worn during each background variation.

Two signers participated in the live testing sessions. The male signer has a height of 5 feet 8 inches and is 22 years old, while the female signer has a height of 5 feet 5 inches and is also 22 years old. The details of the setup are summarized in the table below.

**Table 17: Experimental Setup Summary**

*Table shows the details about live experimental setup.*

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment Number** | **Distance from camera** | **Background** | **Signer** |
| 1 | 3 feet 11 inches | Bookshelves | Female |
| 2 | 3 feet 11 inches | White Board | Female |
| 3 | 3 feet 11 inches | Bookshelves | Male |
| 4 | 3 feet 11 inches | White Board | Male |

### Experimental Results

The signs performed and tested by signers were: Yes, No, Help me, Please, Thank you, Sorry, Goodbye, Hello, Name, I love you, Where, When, Why, How, Eat, Drink, More, Finished, Bathroom, and Friend.

The tables below summarize the results of live testing experiments. A tick (✔) indicates that our model correctly predicted the performed sign, while a cross (✘) indicates that the predicted sign did not match the performed sign.

**Table 18: Live Testing Results**

*Table shows the details of results of live testing experiments.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Label** | **Model Prediction for Experiment** | | | |
| **1** | **2** | **3** | **4** |
| **Yes** | ✔ | ✘ | ✘ | ✘ |
| **No** | ✘ | ✔ | ✔ | ✘ |
| **Help** | ✘ | ✘ | ✔ | ✔ |
| **Please** | ✔ | ✘ | ✘ | ✘ |
| **Thank You** | ✘ | ✔ | ✘ | ✔ |
| **Sorry** | ✘ | ✘ | ✔ | ✘ |
| **Goodbye** | ✔ | ✘ | ✘ | ✘ |
| **Hello** | ✘ | ✘ | ✔ | ✘ |
| **Name** | ✔ | ✘ | ✘ | ✘ |
| **I Love You** | ✘ | ✘ | ✘ | ✘ |
| **Where** | ✔ | ✔ | ✔ | ✘ |
| **When** | ✘ | ✘ | ✘ | ✘ |
| **Why** | ✔ | ✘ | ✔ | ✘ |
| **How** | ✘ | ✘ | ✘ | ✘ |
| **Eat** | ✘ | ✔ | ✘ | ✔ |
| **Drink** | ✘ | ✘ | ✘ | ✘ |
| **More** | ✘ | ✘ | ✘ | ✘ |
| **Finished** | ✘ | ✘ | ✘ | ✘ |
| **Bathroom** | ✘ | ✔ | ✘ | ✘ |
| **Friend** | ✘ | ✘ | ✘ | ✘ |

## Results and Discussion

During the testing phase on our dataset, we utilized the third architecture and obtained a training accuracy of 86-93% for 5 labels. Our observations shed light on the impact of certain architectural components on the accuracy of the model.

One noteworthy finding is that the inclusion of dropout, maxpooling, and time distributed layers in our model led to an improvement in accuracy. These layers introduce regularization techniques and help enhance the overall performance of the model. By incorporating dropout, we mitigate the risk of overfitting by randomly deactivating certain neurons during training, allowing the model to generalize better to unseen data. Similarly, the addition of maxpooling layers aids in reducing the spatial dimensions of the input, extracting the most relevant features, and enhancing the model's ability to capture important patterns. Time distributed layers, when used in sequence models, enable the application of operations to each time step independently, thereby enriching the model's temporal understanding.

On the other hand, we observed a significant decrease in accuracy when the number of labels doubled. This highlights the challenge of training a model with a larger number of classes. As the complexity of the task increases, the model faces difficulties in distinguishing between a higher number of distinct signs. This decrease in accuracy can be attributed to the increased complexity and the potential overlap between similar signs, which pose challenges for the model in accurately classifying them.

To further improve the accuracy, future iterations of our model could benefit from exploring advanced techniques such as fine-tuning pretrained models, optimizing hyperparameters, or incorporating more sophisticated architectures specifically designed for sign language recognition. Additionally, increasing the size and diversity of the dataset could provide more robust training examples, enabling the model to better generalize and adapt to a wider range of sign variations.

Overall, these results emphasize the importance of architectural choices, regularization techniques, and the number of labels in training accurate sign language recognition models. They also indicate potential avenues for further research and development in this domain.

# Conclusion

## Critical Analysis of The Research Items

### Strengths

* Skeleton-based SLR technique that makes use of characteristics from trained whole-body posture estimators as well as whole-body Keypoints and separates the background from the subject.
* Open-Source Libraries make sign language research widely accessible by decreasing training time and facilitating effective inference.
* A generative multi-stream architecture that does not require extra hardware and enhances feature richness without running the risk of impracticability.
* Using image processing and convolution neural networks increases the application's integrity and flexibility when compared to previous project work.
* Unlike other models, emphasis has been laid on learning sign language recognition and translation. The transformer encoder with CTC loss trains to recognize and learn representation for end-level goals of achieving accurate results.
* With an accuracy of 91.7%, CNN with (ANN classification) holds a Jaccard Index of 0.789 (Gesture spotting competition) due to its simpler and more effective approach to training its models.
* The Sign-Pose-based Transformer performance has shown significant improvement on different datasets (WLASL and LSA64 datasets) as compared to the previous state-of-the-art model.
* Instead of using a translation based on sign language grammar, which is still possible when gloss is not available, there is a direct method for translating from spoken to video.

### Weakness

* Accuracy of Image processing and CNN limit dataset to only English Alphabets and do not cover proper dictionary words and even numerical representations.
* Limitations of the Transformer encoder model are the inability to understand the relationship between various sign articulators (e.g., faces, hands, and bodies).
* CNN models are usually too dependent on preprocessing input data.

Sign Pose SPOTER model has questionable computational efficiency for some datasets.

## Relevance Of Review to Proposed Research Work

The goal of our project is to make a real-time and easily accessible model that can bridge the gap between the deaf community and the hearing community. The above-discussed advantages are relevant to our goal in the following ways:

* Our major focus is to use a skeleton-based model that can easily distinguish between the background and the subject and provide high accuracy so that the recognition process is efficient and fast which will, in turn, assist us in making our project real-time.
* Generative architecture will make our project applicable in real-world scenarios where proper hardware is almost never available and the data input quality is much lower than that of the training datasets.
* Sign language recognition is not our end goal but is in fact a necessary step needed to translate the signs into complete English language sentences. For this step it is necessary to use works that focus specifically on translation that is both ISL and CSLR.
* Sign to Gloss translation approach is not enough and requires further processing to accurately translate it into appropriate English sentences. The papers referred to provide techniques to translate sign language into English sentences directly without involving gloss which might be beneficial to us in implementing a simpler and more efficient solution to the problem.

## Motivation Behind Methodology

Our model covers up the objective of obtaining accurate results in Real Time. The model takes a continuous sequence of frames as an input to learn the signer's behavior and due to this means of detection, its ability of recognizing the sign against a demonstrated action is higher. The continuous sequence is the reason why it’s able to detect proper words instead of alphabets and numbers like other models we discovered. GPU consumption in this program is stable and not extensive so low-end systems can also execute this program. Also, due to feature extraction the training data is minimized enough for smooth training.

A Sign Language Recognizer model needs keypoints information of face, pose and hands that can be accurately deduced by MP Holistics. The accuracy achieved by this model in keypoints detection is up to 95.8%, Other pose estimator models require these components (hand, face etc) to be cropped separately from the frames hence reducing the image resolution. MP holistic addresses the various regions using an area of an image resolution in a multi-stage pipeline preserving the entire resolution of the image. The framework of MediaPipe, which was created expressly for complicated perceptual pipelines using accelerated inference (an extensive GPU) , currently provides quick and precise, yet distinct solutions for the process of landmark recognition. All of this is attained while considering the real time approach hence making it suitable for our project.

Utilizing An LSTM model benefited in Several ways. Firstly, we required less data to produce a hyper-accurate model because the LSTM model is quite dense. Secondly, it was faster to train as the LSTM neural network is much denser. So rather than having around 30 to 40 million parameters, as one would in a simple neural network, we will have less than half a million parameters which means it will be a lot faster to train the LSTM model rather than one without it. Finally, the LSTM neural network is a lot simpler, due to having lesser parameters, which means that it will perform faster when it comes to detecting sign language in real-time.

## Results and Accuracy

The training yielded a category accuracy of 90.02%. We first and principally noticed the outcomes' independence from the background during the experiments. This was linked to the fact that we employed a skeleton model, which solely tracks important body parts and is unaffected by additional data or background noise. Due to the model's usage of the identical hand motions when making signals, it was reported that it confused the recognition of "thank you" and "I love you" This may be because we prepared the training dataset ourselves by demonstrating signs in front of the webcam. After integrating the MS-ASL Dataset, we anticipate seeing a significant increase in our findings.

## Challenges Faced

We faced several challenges in this project. Firstly, due to unavailability of the Nvidia graphic card we were unable to execute a state-of-the-art model that used GCNN, STNN, and Transformer. Thus we decided to shift to a lightweight model like LSTM that was within the constraints of the hardware available to us. Secondly, we faced a challenge in achieving highly accurate results for each label as some signs had similar movements involved in them. Namely the model was confused between the recognition of “thank you” and “hello” because of the same movements of hand while performing the signs.

## Completed Work

Our team has completed a comprehensive project that involved the creation of a model utilizing LSTM and BiLSTM. We trained and tested this model to ensure its accuracy and effectiveness. To facilitate this process, we created our own dataset for training and testing, which we preprocessed to optimize its quality.

With our dataset and model in place, we developed a prototype that enabled us to live test the model and demonstrate its real-time output. This step was particularly important, as it allowed us to see how our model performed in real-world scenarios and make any necessary adjustments to improve its performance. Overall, we are proud of the work we have done and believe that our model has the potential to be a valuable tool in a variety of applications.

## References

List all important sources of information which have been consulted for this project,

[1] S. Jiang, et al., "Skeleton aware multi-modal sign language recognition," in   
 Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern   
 Recognition, 2021. [Online].Available: <https://arxiv.org/pdf/2103.08833v5.pdf>.   
 [Accessed: October 2, 2022].

[2] P. Selvaraj, et al., "OpenHands: Making Sign Language Recognition Accessible with   
 Pose-based Pretrained Models across Languages," arXiv preprint arXiv: 2110.05877,   
 2021. [Online]. Available: <https://arxiv.org/pdf/2110.05877v1.pdf>. [Accessed:   
 September 29 2022].

[3] C. C. Amorim, D. Macêdo, and C. Zanchettin, "Spatial-temporal graph convolutional   
 networks for sign language recognition," in International Conference on Artificial   
 Neural Networks, Springer, Cham, 2019. [Online]. Available: Papers with Code,   
 <https://paperswithcode.com/paper/spatial-temporal-graph-convolutional-networks>.   
 [Accessed: October 2, 2022].

[4] Y. Kim, et al., "Keypoint based Sign Language Translation without Glosses," arXiv   
 preprintarXiv:2204.10511,2022.[Online].Available:   
 <https://arxiv.org/pdf/2204.10511v2.pdf>. [Accessed: September 29, 2022].

[5] K. Thakkar and P. J. Narayanan, "Part-based graph convolutional network for action   
 recognition," arXiv preprint arXiv:1809.04983, 2018. Available:   
 <https://arxiv.org/abs/1809.04983>. [Accessed: September 29, 2022]

[6] D. Huh, et al., "Generative multi-stream architecture for American sign language   
 recognition," in 2019 IEEE MIT Undergraduate Research Technology Conference   
 (URTC), IEEE, 2019. [Online]. Available: ArXiv,   
 <https://arxiv.org/pdf/2003.08743v1.pdf>. [Accessed: October 3, 2022]

[7] B. Shi, et al., "Open-Domain Sign Language Translation Learned from Online Video,"   
 arXiv preprint arXiv:2205.12870, 2022.Available: arXiv,   
 <https://arxiv.org/abs/2205.12870>. [Accessed: October 3, 2022]

[8] Y. Cao, et al., "Explore More Guidance: A Task-aware Instruction Network for Sign   
 Language Translation Enhanced with Data Augmentation," arXiv preprint   
 arXiv:2204.05953, 2022. Available: arXiv, <https://arxiv.org/pdf/2110.05877v1.pdf>.   
 [Accessed: September 29, 2022].

[9] J. Zheng, et al., "An improved sign language translation model with explainable   
 adaptations for processing long sign sentences," Computational Intelligence and   
 Neuroscience, 2020. [Online]. Available:   
 <https://www.hindawi.com/journals/cin/2020/8816125> [Accessed: October 2, 2022].

[10] D. Li, et al., "Word-level deep sign language recognition from the video: A new large-   
 scale dataset and methods comparison," in Proceedings of the IEEE/CVF Winter   
 Conference on Applications of Computer Vision, 2020, pp. 2858-2867. Available:   
 <https://arxiv.org/pdf/1910.11006v2.pdf>. [Accessed: September 29, 2022].

[11] B., M., K., M., P., R., et al. "Sign Language Text to Speech Converter using Image   
 Processing and CNN." Student, Dept. of Information Technology, Vidyalankar   
 Institute of Technology, Mumbai, Maharashtra, India, 2020. Available:   
 <https://www.irjet.net/archives/V7/i4/IRJET-V7I4126.pdf>. [Accessed: September 29,   
 2022].

[12] L. Pigou, A. van den Oord, G. Droogmans, T. Mensink, and J. van de Weijer, "Sign   
 language recognition using convolutional neural networks," in Proceedings of the   
 European Conference on Computer Vision (ECCV), Cham, 2014, pp. 572-578.   
 Available:   
 <https://link.springer.com/content/pdf/10.1007/978-3-319-16178-5_40.pdf>   
 [Accessed: September 29, 2022].

[13] N. C. Camgoz, S. Hadfield, O. Koller, R. Bowden, "Sign language transformers: Joint   
 end-to-end sign language recognition and translation." in Proceedings of the   
 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020.   
 Available:   
 <https://paperswithcode.com/paper/sign-language-transformers-joint-end-to-end>.   
 [Accessed: September 29, 2022].

[14] M. Boháček and M. Hrúz, "Sign Pose-based Transformer for Word-level Sign   
 Language Recognition," in Proceedings of the IEEE/CVF Winter Conference on   
 Applications of Computer Vision (WACV), 2022, pp. 2273-2282. [Online].   
 Available: <https://openaccess.thecvf.com/content/WACV2022W/HADCV/papers/Bohacek_Sign_Pose-Based_Transformer_for_Word-Level_Sign_Language_Recognition_WACVW_2022_paper.pdf>.  
[Accessed: September 29, 2022].

[15] C. Lugaresi, K. Patel, G. Bouma, J. Kofler, A. Gruenstein, L. Saroufim, Y. Xie, F.   
 Naccarato, R. Chattopadhyay, D. Anguelov, and M. Agrawala, "Mediapipe: A   
 framework for building perception pipelines," arXiv preprint arXiv:1906.08172,   
 2019. [Online]. Available: <https://arxiv.org/pdf/1906.08172.pdf> [Accessed:   
 September 29, 2022].

[16] H. R. Vaezi Joze, O. Koller, "MS-ASL: A Large-Scale Data Set and Benchmark for   
 Understanding American Sign Language," arXiv e-prints, arXiv:1812, September 30,   
 2022. [Online]. Available: <https://arxiv.org/pdf/1812.01053.pdf>. [Accessed:   
 September 30, 2022].