Heriot-Watt University

F20PA: Final Year Dissertation

American Sign Language Recognition Using Deep Learning Models

Author:
Gauri Revankar

Supervisor:
Ms. Smitha Kumar

A thesis submitted in fulfillment of the requirements
for the degree of BSc. (Hons) Computer Science (Data Science)

in the

School of Mathematical and Computer Sciences

April 2024

Declaration of Authorship

I, Gauri Revankar, declare that this thesis titled, 'American Sign Language Recognition Using Deep Learning Models' and the work presented in it is my own. I confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed: Gauri Revankar

Date: 15/04/2024

Student Declaration of Authorship

HERIOT WATT UNIVERSITY
UK | DUBAI | MALAYSIA

Course code and name:	F20PA – Research Methods and Requirements Engineering
Type of assessment:	**Individual**
Coursework Title:	Final Year Dissertation
Student Name:	Gauri Revankar
Student ID Number:	H00373987

Declaration of authorship. By signing this form:

- **I declare** that the work I have submitted for individual assessment OR the work I have contributed to a group assessment, is entirely my own. I have NOT taken the ideas, writings or inventions of another person and used these as if they were my own. My submission or my contribution to a group submission is expressed in my own words. Any uses made within this work of the ideas, writings or inventions of others, or of any existing sources of information (books, journals, websites, etc.) are properly acknowledged and listed in the references and/or acknowledgements section.

- I confirm that I have read, understood and followed the University's Regulations on plagiarism as published on the University's website, and that I am aware of the penalties that I will face should I not adhere to the University Regulations.

- I confirm that I have read, understood and avoided the different types of plagiarism explained in the University guidance on Academic Integrity and Plagiarism

Student Signature *(type your name):* *Gauri Revankar*

Date: *15/04/2024*

"The greatest glory in living lies not in never falling, but in rising every time we fall."

Nelson Mandela

Abstract

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

Keywords - Sign Language Recognition, Isolated Sign Language Recognition, Deep Learning, American Sign Language, Transfer Learning, Computer Vision.

Acknowledgements

I would like to express my gratitude to Ms. Smitha Kumar for her invaluable guidance and encouragement, which supported me throughout this project. Her insightful feedback and advice contributed to the refinement of my work and helped me complete this project. I would also like to thank my family and friends for their constant motivation and support.

Contents

List of Figures

List of Tables

List of Abbreviations

SLR	Sign Language Recognition
CSLR	Continuous Sign Language Recognition
ASL	American Sign Language
WLASL	World Level American Sign Language
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
Bi-LSTM	Bi-directional Long-Short Term Memory
ViT	Vision Transformer
GPU	Graphics Processing Unit

Chapter 1

Introduction

Language is a cornerstone of human civilization, playing an important role as a means of communication to convey thoughts, emotions, and feelings. It takes various forms, including written symbols, gestures, and vocalizations.

Sign Language is a visual language primarily used by the deaf and mute community to communicate. It involves a combination of hand shapes, hand and arm movements, and facial expressions to convey a message. The World Health Organization (WHO) reports that there are around 466 million people with speech or hearing impairment [69][41]. The United Nations Convention on the Rights of Persons with Disabilities encourages one to facilitate and promote sign languages to ensure that individuals with disabilities exercise equal rights as others [35]. There are over 300 Sign languages to this date [46] and over 70 million deaf people communicate using sign languages, across the world [35].

There is no one universal sign language used around the world. Different sign languages have been developed across different countries that are unique to their region and culture. For

instance, American Sign Language, or ASL, is a sign language used most by people in the United States. British Sign Language (BSL)[60], Indian Sign Language (ISL) [63], French Sign Language (FSL) [15], etc. are a few well-known Sign Languages [46].

Sign Language Recognition (SLR) is considered a part of behaviour recognition, which is a field of research that involves identifying and understanding human movements, and gestures through pattern recognition, computer vision, etc [40]. SLR involves recognition, and classification of human gestures and delivering the corresponding words or phrases that they convey [59]. Various attempts have been made for sign language detection and recognition using sensors and wearable gears to track hand and body movements, but a major drawback of such an approach is that it requires costly instruments and high-end processing power. On the other hand, Deep Learning has made great advancements in the technological field and is being adopted in other industries such as healthcare, agriculture, finance, and many more. Deep learning has transformed computer vision tasks such as image classification, object detection, and facial recognition and is also being used to aid the process of SLR. Deep learning also provides a cost-effective solution to the problem stated above.

SLR can be broadly categorized into 3 main types - Static, Isolated, and Continuous. Static SLR deals with recognizing static hand gestures depicted as still images. They often represent alphabets, digits, and a fixed set of characters. Isolated SLR, on the other hand, involves recognizing individual signs performed within a short video. It is a more challenging and complex task compared to Static SLR. Continuous SLR involves recognizing a sequence of signs performed one after the other. This requires an additional task of identifying individual signs in the sequence [55]. Hence, Isolated SLR can be considered a part of Continuous SLR.

Automating SLR is a complex and challenging task, requiring the integration of feature extraction techniques and classification methods [56]. Hence this project seeks to explore various hybrid deep learning models and compare their performances in Sign Language Recognition on an American Sign Language (ASL) dataset.

1.1 Motivation

To understand sign language, several factors must be considered and understood. These include hand movements, hand and shoulder orientation, and facial expressions. Automating SLR serves as a possible solution to reduce the communication gap between the hearing and vocally impaired community and the rest of the world. The motivation for this research stems from the pressing need to narrow this gap between those with speech impairment, hearing impairment, or both, and the rest of society.

Given that we are working with multi-dimensional spatiotemporal data (videos), the deep learning models that we develop must understand and interpret the signs demonstrated by the signer in 3D space[11]. This is quite a challenging task. Hence, although many academic works have been published in this field, a solution assuring a real-time application has not been found yet [1].

Hence, this project seeks to develop and propose efficient Deep Learning models that can recognize American Sign Language and, evaluate and compare their performances.

1.2 Aim

This study aims to explore the efficiencies and performances of different combinations of Deep Learning models for Sign Language Recognition from videos.

1.3 Objective

The overall objectives of this project can be summarized as follows:

1. Conduct comprehensive research for a suitable dataset that aligns with the goals of Isolated SLR.
2. Implement and analyze data preprocessing techniques on the dataset found.
3. Investigate and identify a combination of diverse Deep Learning network models to improve the performance of SLR.
4. Train the chosen models using the acquired dataset.
5. Conduct an evaluation of the trained models and compare their performances.

1.4 Organization

The sections of the paper are organized as follows:

1. **Chapter 1** contains the **Introduction** section which states the Aim, Objective, and Motivation of the project.

2. **Chapter 2** contains the **Project Background** which describes the history of Sign Language, the creation of American Sign Language, and the evolution of the Deep Learning Network, followed by the Related Work section comprising a summary and analysis of Literature Reviews of other works in the same field.

3. **Chapter 3** contains the **Implementation** section of the project that describes the various techniques applied to prepare the dataset and build the model architectures.

4. **Chapter 4** contains the **Model Testing and Evaluation** section showing a comparison between the model performances.

5. **Chapter 5** contains the **Conclusion and Future Work** along with the challenges faced during the implementation and the limitations of the project.

6. **Appendix A** contains the **Project Management** section that includes the Initially proposed methodology, project plan, and risk management.

7. **Appendix B** contains the **PLES section** of the project.

8. **Appendix C** contains the **Project Journals**.

Chapter 2

Project Background

This section begins with giving an overview of the history of sign language followed by the history and evolution of deep learning models. The section later elaborates on the commonly used deep learning models in the industry, followed by the data preprocessing techniques and evaluation criteria used to evaluate the models. The section then highlights the related work in the same field and concludes with the critical analysis section.

2.1 History of Sign Language

Sign Language is a non-verbal language that uses hand and body movements, and facial expressions as a primary means of communication instead of spoken words [46]. Although sign language is a way of communicating with the deaf and mute, it is also a way of communicating with those with autism, learning disabilities, or speech disorders. Just like spoken languages, different countries have their own sign languages. There are around 300 sign languages used around the world to this date [46].

Back in the 1500s, Geronimo Cardano, an Italian mathematician and physician, played a significant role in the history of sign language. He was the first to argue that deaf people could learn to read and write without learning how to speak first. In the 16th century, Pedro Ponce de León, a Spanish Benedictine monk, proposed a formal sign language for the hearing impaired [30]. During their daily periods of silence, the Benedictine monks used sign language to communicate their messages. Inspired by the practice, Ponce de León developed a method for teaching the deaf to communicate by using the gestures he practiced at his monastery [30]. This method helped to establish systems that are currently in use worldwide. In 1750, a French Catholic priest named Abbe de L'Epee, established the first social and religious association for the deaf in Paris, which later served as the basis for French Sign Language (FSL) [30].

The roots of ASL go back to the early 1700s, when there existed a large proportion of the deaf population in Martha's Vineyard in Massachusetts. To overcome the communication barrier between the deaf and hearing residents, the community created a new sign language. On the other hand, Laurent Clerc and Thomas Hopkins Gallaudet were the first to establish an American school for the Deaf in 1817 [14]. Clerc was a deaf educator from France and he brought Old French Sign Language to the States. Together, they used the homemade hand signs of the American students, Martha's Vineyard Sign Language (MVSL), and FSL to create American Sign Language [64][14].

For the past thirty years, the Computer Vision community has delved into studying Sign Languages [57][61] with an aim to develop systems for production and translation that can translate sign languages into spoken languages and vice versa [34]. It is an attempt to ease the daily lives of the Deaf [26][16].

Computer Vision is a field of Artificial Intelligence or AI that enables machines to understand

and interpret information from visual data, including images and videos. It instructs computers to comprehend images pixel-by-pixel [58]. It encompasses key components like image recognition, object detection, facial recognition, image segmentation, feature extraction, and many more.

Machine Learning falls under the umbrella of AI, which allows computers to autonomously learn from data and use the previously gained knowledge to identify patterns and make predictions with minimal human intervention [33]. Supervised Learning, Unsupervised Learning, and Reinforcement Learning are the different types of machine learning [33].

Deep Learning is a subset of Machine Learning. It analyzes the significant features on its own by processing data in multiple steps and creates an advanced machine learning model through multiple transformations of features. The word "deep" in deep learning pertains to using multiple layers within the network structure, based on the concept of Artificial Neural Networks (ANNs). In today's world, deep learning has played an important role in the recent advancements in Computer Vision, Natural Language Processing (NLP), and speech and audio recognition [42].

2.1.1 History and Evolution of Deep Learning Networks

The concept of Deep Learning originated in 1943, when Walter Pitts and Warren McCulloch developed a computer model inspired by the neural architecture of the human brain [23].

Alexey Grigoryevich Ivakhnenko (developer of the Group Method of Data Handling) and Valentin Grigor'evich Lapa (author of Cybernetics and Forecasting Techniques) pioneered the development of deep learning algorithms in 1965 [23]. They used models with polynomial activation

functions, which then underwent statistical analysis [23]. The most suitable features, chosen through statistical methods from each layer, were then passed to the next layer. This was a slow and manual process [23].

The Convolutional Neural Networks (CNNs) were first developed by Kunihiko Fukushima. The architecture of neural networks was developed with multiple pooling and convolutional layers [23]. In 1979, he introduced Neocognitron, an ANN characterized by a hierarchical, multilayered structure that allowed computers to "learn" and recognize visual patterns [23]. Inspired by Neocognitron, the first work on modern CNNs was built by Yann LeCun to recognize handwritten digits. It was named LeNet after Yann LeCun himself. Since LeNet had a simple architecture it worked only on low-resolution images [23].

The concept of Backpropagation while training deep learning models was introduced in 1970. Seppo Linnainmaa had authored a FORTRAN code for backpropagation for his master's thesis. Yann LeCun made the first practical application of backpropagation at Bell Labs in 1989 [52]. LeCun integrated backpropagation with CNNs to recognize "handwritten" digits. This system was ultimately employed for reading the numbers on handwritten checks [52].

In 1995, Dana Cortes and Vladimir Vapnik developed the Support Vector Machine (SVM), a system for identifying and mapping comparable data. Long Short-Term Memory (LSTM) was developed in 1997, by Sepp Hochreiter and Juergen Schmidhuber [23].

In 1999, a crucial advancement in the evolution of deep learning occurred as computers gained increased processing speed, and Graphics Processing Units (GPUs) were developed. This was the time when neural networks began to compete with SVMs. Although neural networks might have been slower than SVMs, they yielded better results when processing the same data [23].

2.1.2 Application of Deep Learning in Image Recognition

Hinton and Alex Krizhevsky developed AlexNet in 2012, introducing deeper and wider CNNs. ReLU activation function replaced Sigmoid, which accelerated training and addressed gradient dispersion issues. The introduction of the Dropout concept aimed to reduce neuron complexity and mitigate overfitting. In 2014, the inception model was introduced by GoogleNet which enabled efficient resource utilization in CNNs. The Inception model evolved with versions – v1, v2, v3, v4, and Inception-ResNet, which resulted in the improvement in the image classification accuracy and parameter utilization [73].

2.2 Deep Learning Networks

Below are a few deep learning networks that are commonly used in the industry:

1. **Convolutional Neural Network (CNN)**

   CNN is a type of Artificial Neural Network (ANN) tailored for tasks involving image or object recognition and classification. Renowned for their effectiveness in computer vision applications, CNNs excel in tasks such as localization, segmentation, video analysis, and more [25]. CNN is composed of mainly three types of layers - a convolutional layer, a pooling layer, and a fully connected layer. The image pixels of the image are fed as input, in the form of arrays. The convolutional layer extracts features from visual data or images. The fc layer or fully connected layer classifies the image and gives the output. There can be multiple hidden layers to perform calculations and aid in feature extraction. Activation functions allow the model to learn complex relationships in data [25].

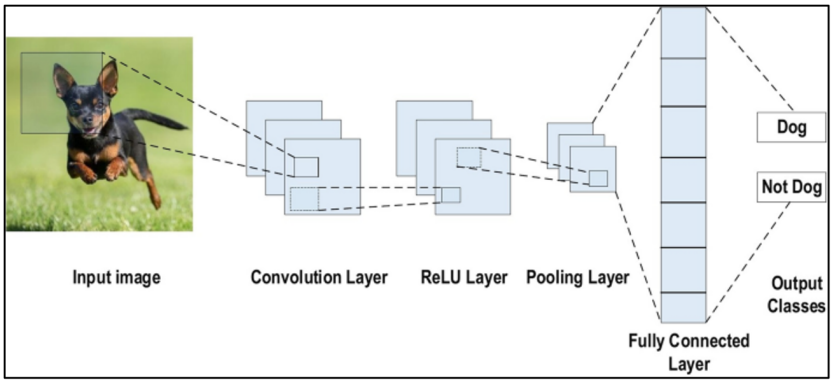

FIGURE 2.1: CNN Architecture [37]

2. **CapsNet (Capsule Network)**

   CapsNet is an ANN designed to enhance the modeling of hierarchical relationships [49]. In a capsule network, each capsule contains a cluster of neurons, with each neuron's output

representing a unique characteristic of the same feature. This approach provides the benefit of recognizing the entire entity by first identifying its components. The input to a capsule is the output (features) from a CNN. The main difference between CapsNet and traditional CNN lies in the use of vectors for more detailed representation, as opposed to scalars [65]. CapsNet offers distinct benefits over CNNs, such as the retention of spatial information, minimization of loss of features from pooling, requiring less data for training, and expedited training duration [65]. However, in the case of complex datasets, capsules may struggle to handle dense data volumes, causing the model to underperform [65].

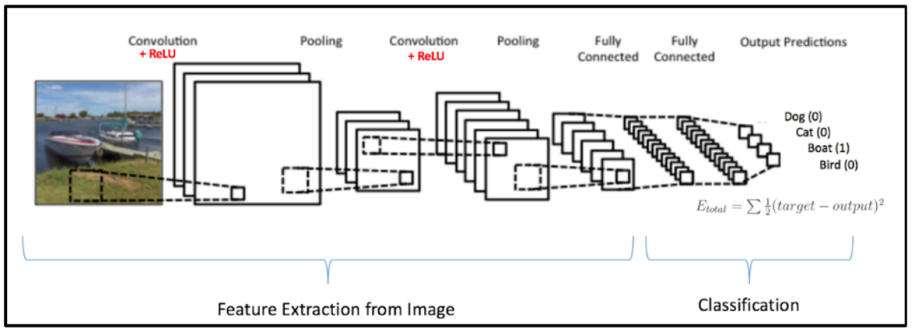

FIGURE 2.2: CapsNet Architecture [36]

3. **SKNet (Selective Kernel Network)**

SKNet is a type of CNN that employs selective kernel units in its architecture along with selective kernel convolutions. Selective Kernel Networks enhance the ability of neural networks to adaptively adjust the receptive field size for different regions of an image. SKNet uses a "selective" operation that enables each channel to adaptively adjust its convolutional kernel size. This adaptability is intended to capture both local and global contextual information in an image. This improves the model's ability to handle diverse patterns and structures within an image [51].

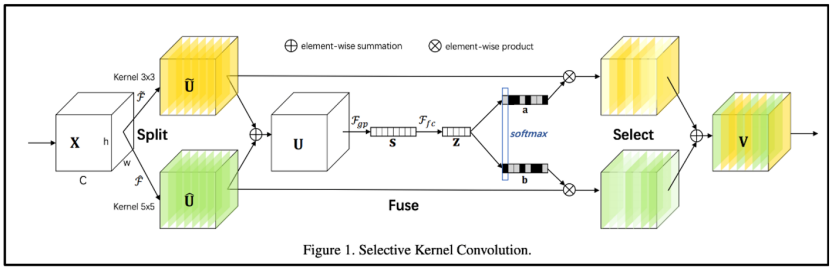

FIGURE 2.3: SKNet Architecture [51]

4. **ResNet (Residual Network)**

ResNet is a type of CNN that was introduced to address the vanishing gradient problem in deep neural networks. The reason behind incorporating more layers in a neural network model is that these layers progressively learn more complex features. However, augmenting layers on top of a network can lead to performance degradation. Take a traditional CNN for instance, it is observed that the error rate for a 56-layer network is higher than that for a 20-layer network [45]. This was evident across both training and test datasets. Optimization function, initialization of the network, and notably the vanishing gradient problem may be a few of the reasons [45]. ResNet uses skip connections or residual blocks which establish an identity mapping to activations earlier in the network to effectively mitigate performance degradation, a problem commonly encountered in deep neural architectures [48]. The core idea is to have shortcut connections that bypass one or more layers, allowing the gradient to backpropagate easily.

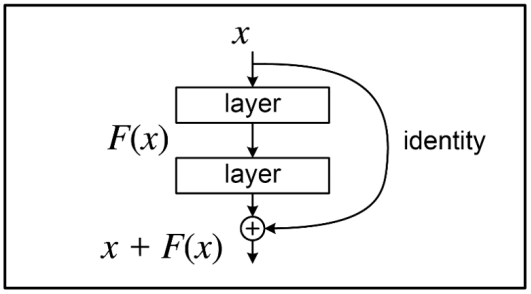

FIGURE 2.4: ResNet Architecture [71]

5. **Xception**

Introduced by Google in 2017, Xception stands for Extreme Inception. It is considered to be an extreme version of the Inception model. Its architecture uses depthwise separable convolutions which proves to be more efficient in terms of computational time [22]. In this architecture, pointwise convolutions are followed by depthwise convolutions. Additionally, the architecture uses residual connections also known as skip or shortcut connections [13]. On evaluating the model on the ImageNet and JFT datasets, it is observed that it outperforms the Inception model on comparison with the state-of-the-art results [12].

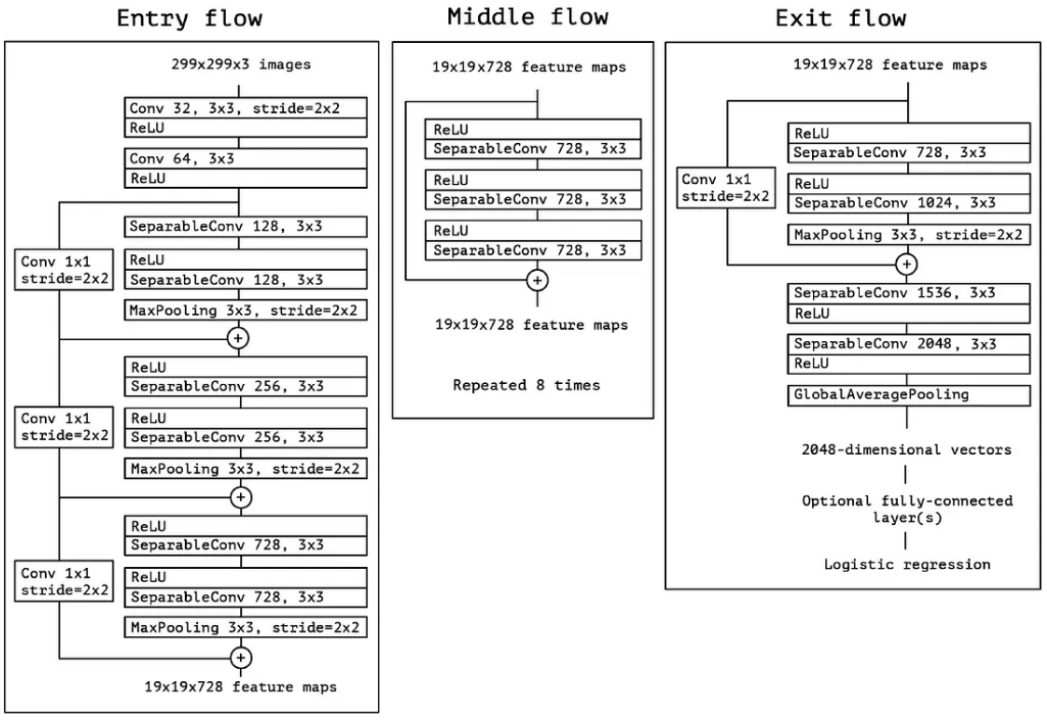

FIGURE 2.5: Xception Architecture [12]

6. **Recurrent Neural Network (RNN)**

An RNN is a type of ANN designed for processing sequential or time series data. RNNs incorporate feedback loops, like backpropagation, in their computational framework to feed information back into the network [38]. These feedback loops facilitate the retention of information and this effect is often referred to as "memory". This mechanism interlinks inputs enabling RNNs to effectively process sequential and temporal data [38].

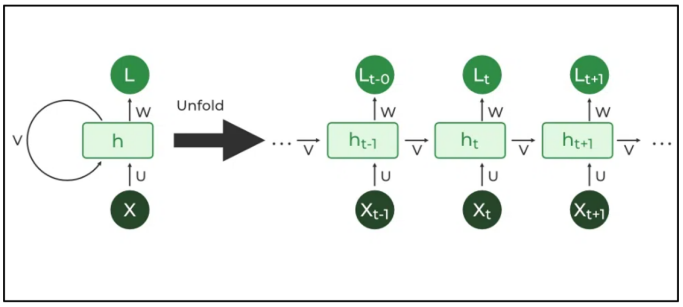

FIGURE 2.6: Standard RNN and unfolded RNN Architecture [2]

7. **Long-Short Term Memory (LSTM)**

LSTMs, a type of RNNs, are adept at capturing long-term dependencies within sequential data. They make use of memory cells and gates to regulate the flow of information [7]. This mechanism allows selective retention and discardal of information as required, thereby avoiding the issue of vanishing gradient commonly encountered by traditional RNNs [7]. There are primarily three types of gates in an LSTM architecture: the input gate, the forget gate, and the output gate. The input gate is responsible for regulating the flow of information into the memory cell. The forget gate regulates the exit of information from the memory cell. The output gate oversees the flow of information out of the LSTM and into the output [7].

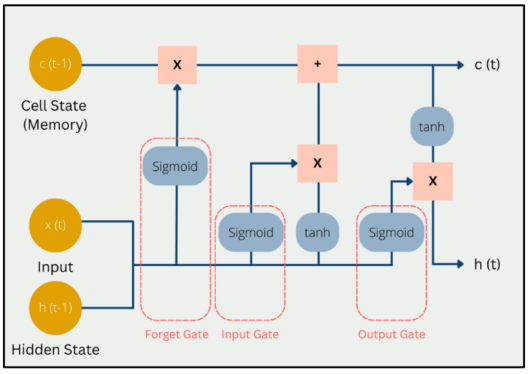

FIGURE 2.7: LSTM Architecture [18]

8. **Bi-directional Long-Short Term Memory (Bi-LSTM)**

Bi-LSTM is an RNN that processes sequential inputs in both forward and backward directions [68]. Bidirectional LSTMs function by presenting the sequential data in forward and backward directions to two independent LSTM networks whose outputs are then merged at each time step, either by summation, multiplication, averaging, or concatenation [76]. Hence, Bi-LSTM contains comprehensive information about all elements of the sequential data before and after each element in a particular sequence.

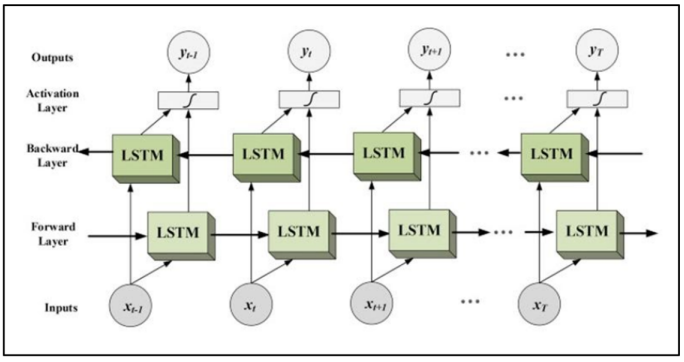

FIGURE 2.8: Bi-LSTM Architecture [68]

9. **Transformers**

Transformers are a type of neural network that heavily rely on attention mechanisms to establish global connections between input and output [67]. An attention mechanism is a

fundamental component of neural network architectures that facilitates focusing on specific parts of the input data while processing it. It enables the network to weigh the importance of different elements in the input data when making predictions. Self-attention, also known as intra-attention, is an attention mechanism that correlates various positions within a single sequence to generate a representation of the entire sequence [67]. Vision Transformer (ViT), first proposed by [3], is an application of the Transformer architecture to perform computer vision tasks. They adapt the Transformer's self-attention mechanism to process 2D image data. ViTs use patches of an image as input tokens and apply self-attention to these tokens. They have shown excellent performance in image classification tasks as mentioned in [3].

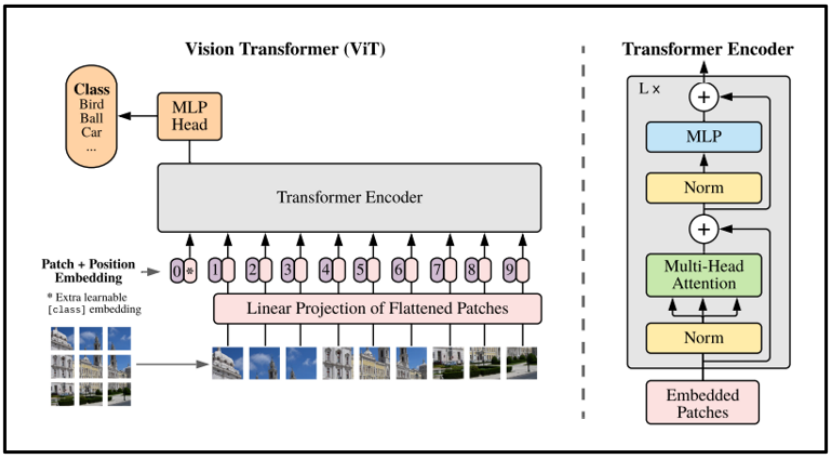

FIGURE 2.9: Vision Transformer Architecture [3]

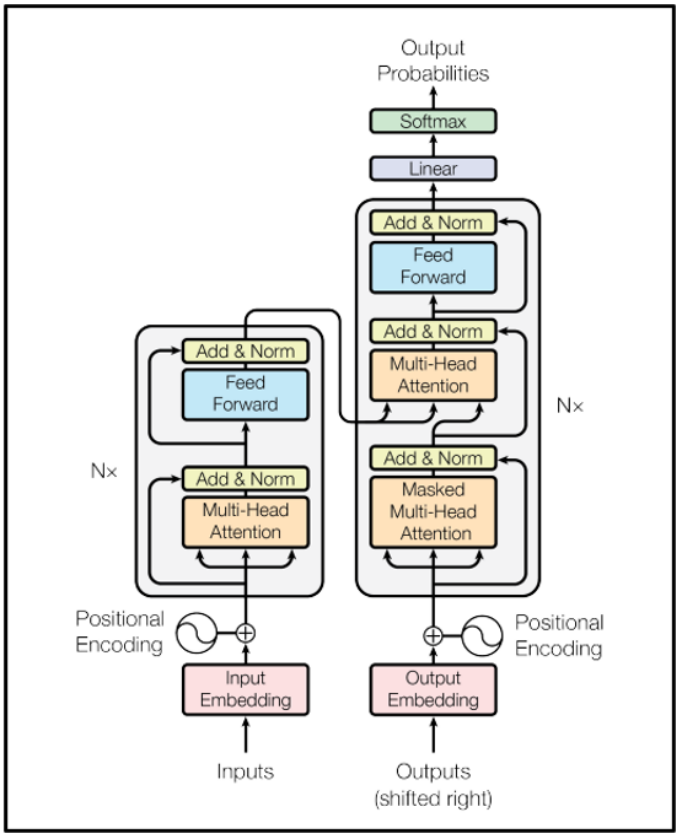

FIGURE 2.10: Transformer Architecture [67]

2.3 Data Preprocessing

Data Preprocessing is the first and crucial step in implementing and training a machine learning model. It is the process of refining the raw data to make it suitable for a model to increase its accuracy and efficiency [17]. Listed below are a few data preprocessing techniques:

Frame Extraction

Efficient and accurate video classification heavily relies on the extraction of spatial and temporal features. Hence, extracting significant key frames from videos is very important in video processing and analysis as it significantly reduces the required computing resources and time [10].

Image Enhancement

Image enhancement involves altering digital images to make them better suited for subsequent image analysis. This can include actions like noise reduction, sharpening, or brightening, all aimed at identifying the important features within the image. [28].

Feature Extraction

Image feature extraction involves extracting all the unique attributes from an image. This is done to capture the key characteristics or features that best represent the image. It is a key element in image processing and helpful for pattern recognition, and machine learning [39].

Sign Language detection and Recognition from videos involves extracting spatial and temporal features. Spatial refers to space and temporal relates to time. Hence, 'spatiotemporal' data refers to data that is collected across both space and time, such as a video [74]. A video is composed of many image frames. Hence, recognition from a video not only involves understanding each image frame but also the relationship between these frames [74]. Here the spatial features refer to the features of each image frame and temporal features refer to the relationship between frames or a sequence of frames.

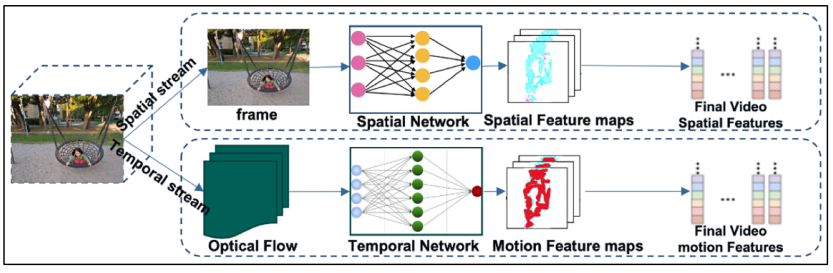

FIGURE 2.11: Spatial and Temporal Features of a Video [66]

Tracking the shape and motion of hands is a vital component of sign language recognition. **MediaPipe** is an open-source framework developed by Google with a set of pre-trained models

FIGURE 2.12: MediaPipe holistic results [43]

to help developers carry out tasks related to image and video analysis, gesture recognition, object tracking, and much more. The advantages of using this method are as follows [75]:

- It converts 2D video frames to 3D spatial features. It converts raw data to sequential data.
- Media Pipe Holistic combines three key components: Media Pipe Hands, Media Pipe Pose, and Media Pipe Face Mesh, which detect keypoints on hand, body, and face, and track hand movements, body poses, and facial expressions.
- It offers instantaneous and real-time detection of hand, body pose, and facial landmarks, thereby, making it a useful tool for SLR.

2.4 Transfer Learning

Transfer learning is a powerful technique in machine learning that leverages the knowledge gained by a pre-trained model from one task and applies it to solve another related task [27]. In our implementation, we take the knowledge of image recognition of the pre-trained models and apply it to recognize Sign Language. When the dataset is small, only the output layer needs to be modified to suit the number of output classes in the new dataset. The weights and

layers of the pre-trained model remain unchanged. By knowing how to detect low-level features such as lines, dots, and curves, transfer learning not only accelerates the learning process but also enables the model to learn and understand the new dataset better, even when the data is limited.

Keras provides several pre-trained models such as Xception, VGG16, ResNet50, InceptionV3, and many more. All these models are trained on the ImageNet Dataset [62]. The pre-trained Xception and ResNet50 models will be used in this implementation.

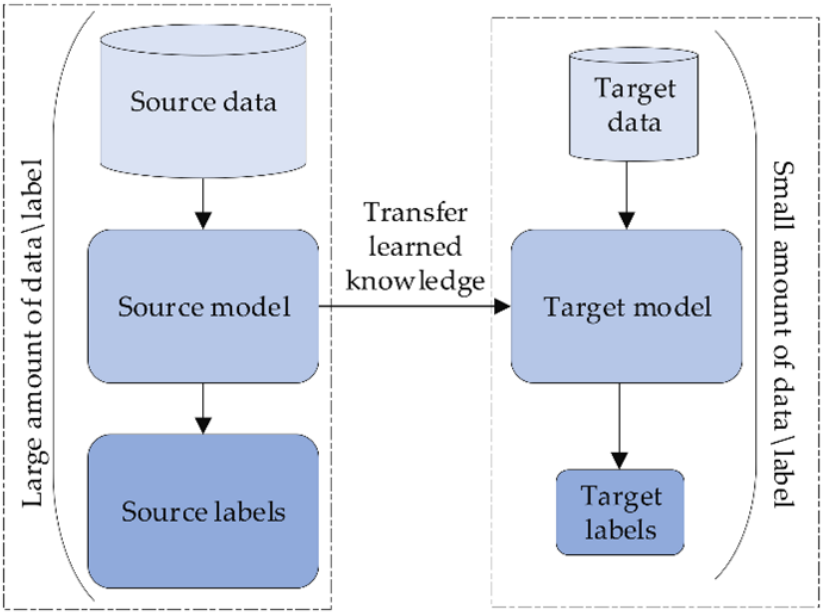

FIGURE 2.13: Transfer Learning [4]

2.5 Evaluation Criteria

Accuracy is an important evaluation metric used to evaluate a deep learning model. It is also the metric used by most researchers in the Related Works section.

After training the models, their performances will be evaluated by generating the following evaluation metrics [44]:

1. **Accuracy**: It is the ratio of correct predictions to the total number of instances evaluated.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

2. **Precision:** It is the ratio of true positive predictions to the total number of positive predictions made.

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

3. **Recall**: It is the ratio of positive instances that are correctly classified.

4. **F-Measure**: It is the harmonic mean of precision and recall values. It is also known as F1 Score.

$$F - Measure = \frac{2 * Recall * Precision}{Recall + Precision} \tag{2.3}$$

Additionally, accuracy and loss graphs will be plotted to visually represent the training and validation accuracies and losses of the models while training. Confusion matrices will also be created to visually describe the correctly classified and misclassified samples while evaluating the models on validation and test datasets.

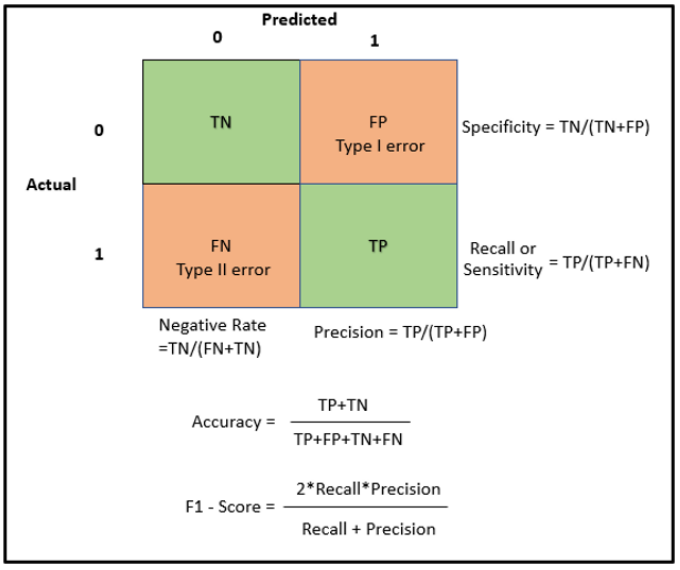

FIGURE 2.14: Confusion Matrix [29]

2.6 Related Work

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

Tunga, A., Sai, V., Nuthalapati and Wachs, J. [5] proposed Pose Based Sign Language Recognition using GCN and BERT. This involved the capturing of spatial and temporal features separately and performing late fusion. The proposed architecture uses a Graph Convolutional Network (GCN) to capture the spatial interactions in the video and Bidirectional Encoder Representations from Transformers (BERT) to capture the temporal dependencies between frames. The model was trained on the WLASL dataset[20]. The model achieved an accuracy of 88.67%, improving the prediction accuracy by 5% compared to GRU and TGCN models proposed by [21]. Top-K accuracy was employed to assess the performance of the model.

B. Natarajan, E. Rajalakshmi, R. Elakkiya, Ketan Kotecha, Ajith Abraham, Lubna Abdelkareim Gabralla, And V. Subramaniyaswamy [6] proposed an End-to-End Deep Learning Frameworks for Sign Language Recognition, Translation and Video Generation. The proposed model used the MediaPipe library and a hybrid CNN + BiLSTM model for pose detection and text generation. A hybrid Neural Machine Translation (NMT) + MediaPipe + Dynamic Gesture Adversarial Network (GAN) model was implemented for transforming multilingual sentences into sign words. The CNN+Bi-LSTM model achieved an accuracy of over 95% compared to the other existing frameworks such as LSTM + KNN, CNN + SVM, Transformer + CTC, and many more. RWTH-PHOENIX-Weather 2014T dataset (German sign language), ISL-CSLTR dataset (self-created Indian sign language dataset), and How2Sign dataset (American Sign Language) are the datasets used for training the models.

Shin, J., Musa Miah, A.S., Hasan, M.A.M., Hirooka, K., Suzuki, K., Lee, H.-S. and Jang, S.-W [32] multi-branch network that combines convolutional and transformer architectures. This design leverages the transformer's ability to compute long-range dependencies along with CNN's capability for local feature extraction, thereby enhancing sign language recognition. The initial features are extracted using the grained model which are then concatenated with the long-range dependency features extracted by Transformer and CNN. The combined features are then applied for classification. The model was inspired by the architectures of CMT, ResNet-50, and DeiT. The proposed model achieved an accuracy of 89% when evaluated against the Korean Sign Language (KSL) dataset. The datasets used to train the model are the KSL benchmark dataset and a self-created dataset which consisted of 4-second videos or 120 frames from each action performed by an individual.

Dongxu Li, Cristian Rodriguez Opazo, Xin Yu, Hongdong Li [21] proposed a pose based Temporal Graph Convolution Network (TGCN) to capture the spatial and temporal components of the human body movement. Their implementation focused on implementing and comparing 2 types of models – a holistic visual appearance-based approach and a 2D human pose-based approach. Their proposed model further boosted the performance of the pose-based approach achieving an accuracy of 62.63% top-10 accuracy on the complete WLASL dataset consisting of 2000 words. The proposed model stacks multiple residual graph convolutional blocks and uses the average pooling result across temporal dimensions. The temporal motion is tracked using the 2D pose and hand keypoints, extracted using OpenPose. In addition to the model, the authors, too, introduced the large-scale WLASL dataset.

Lu, J., Nguyen, M. and Yan, W.Q. [40] propose the Capsule Network (CapsNet) to improve the recognition accuracy. The authors also implemented a Selective Kernel Network (SKNet) with an attention mechanism to extract spatial information and the ResNet model. Furthermore, the authors have also experimented with various combinations to create different hybrid models such as – SKNet + LSTM and combining CapsNet + SKNet + LSTM. The evaluation of the models showed that CapsNet attained an overall accuracy of 98.72% and SKNet with attention mechanism attained 98.88% accuracy. ResNeXt alone achieved 97.82% accuracy. ResNeXt with attention mechanism achieved 98.19% accuracy. A self-created dataset was used to train the proposed models. The dataset comprised nine videos of four classes with the tags - Hello, Nice, Meet, You - and contained 3,596 frames in total, out of which, 2500 frames were chosen for model training and 1,096 for model testing. The evaluation metrics focused upon were the accuracy and loss of the proposed models.

Kothadiya, D., Bhatt, C., Sapariya, K., Patel, K., Gil-González, A.-B. and Corchado, J.M

[19] proposed a hybrid LSTM + GRU model for recognizing Isolated Indian Sign Language (ISL). Feature extraction from frames was performed using InceptionResNetV2. The resulting feature vector array was then fed into an RNN to predict the correct word. Four variations of RNN, combining LSTM and GRU in different sequential configurations, were employed on the self-created dataset, IISL2020. Among these configurations, the model comprising a single layer of LSTM followed by GRU achieved an accuracy of 97% across 11 different signs. In their future work, the authors propose exploring ViTs to obtain more accurate results compared to feedback-based models.

Yufeng Jiang, Fengheng Li, and Zongxi Li [75] attempted to enhance CSLR with a self-attention mechanism and Media Pipe Holistic. They made use of MediaPipe Holistic to convert raw video into sequential data and used sampling strategies to fix the number of input frames. Self-attention layer was employed to extract the spatiotemporal features. They applied feature selection and after much experimentation, they decided on using 60 frames as an input to the self-attention model. LSTM and Bi-LSTM were chosen as base models to compare the evaluations of the proposed model. Accuracy was chosen as their evaluation metric. An accuracy of 40.4% was achieved with face mesh and 59% without face mesh by self-attention mechanism. The authors encourage the use of video augmentation techniques and transfer learning to address the problem of insufficient training samples. The models were trained on the first 10 glosses of the WLASL dataset.

Lu, J., Nguyen, M. and Qi Yan, W. [31] proposed a novel YOLOv4 + LSTM network for real-time recognition. LSTM was used to extract temporal information and YOLO nets were used for spatial information exraction. Finally, the two networks were combined using score fusion. YOLOv3 + LSTM model gave an accuracy of 97.58%, whereas YOLOv4 + LSTM gave

an accuracy of 97.87%. For comparison, SKNet with attention mechanism was implemented. A convolution operation with a 7-by-7 filter was applied to generate feature maps followed by normalization using a sigmoid function to produce the final feature maps. The model gave an accuracy of 98.64%. A CNN + LSTM model was also implemented which gave an accuracy of 98.53%. The models were evaluated on 2 datasets - Weizmann dataset, and a self-created dataset. The videos were converted into a sequence of feature vectors to present features from each video frame.

Chung, W.-Y., Xu, H. and Lee, B.-G. [70] proposed Chinese Sign Language Recognition with Batch Sampling by implementing ResNet and a combination of RNN + Bi-LSTM and comparing their performances. The hand skeletal data served as the features and were fed as an input to the Bi-LSTM model for SLR. The models were evaluated on a subset of the DEVISIGN-L dataset consisting of 8 subjects with 1200 sample videos covering 100 signs. The proposed RNN-Bi-LSTM model gave an accuracy of 98.75% and ResNet gave an accuracy of 99%. However, this study encountered limitations due to inconsistencies in handshape and hand movement trajectories within the DEVISIGN-L dataset. This inconsistency posed challenges for handshape detection using ResNet and hand keypoints detection, leading to frequent false detections, particularly when finger views were unclear or obstructed by other fingers. For future research, the authors plan to incorporate hand skeleton detection and expand the scope to sentence-based sign language recognition. Additionally, they aim to optimize the recognition model and explore its applicability with other sign language datasets.

Fikri Nugraha, Esmeralda C. Djamal [47] proposed a Two-stream CNN to classify for sign language recognition and classification. The two-stream CNN is composed of a spatial and temporal stream. These two streams were combined using the Average Fusion function. The

two-stream separated training not only reduced the training time but also helped overcome resource limitations. The study used 10 words for classification from the American Sign Language Lexicon Video Dataset. The best results were given by using Xception SGD for spatial flow and Xception Adam for temporal flow configuration. The architecture gave a precision of 89.4%. However, the use of Average fusion also served as a limitation to their implementation. It led to bias which lowered the overall accuracy of the model.

Luke T. Woods and Zeeshan A. Rana [72] proposed an encoder-only transformer and used pose keypoint data for SLR. The authors used the enhanced version of WLASL and trained their proposed model on 10, 50, 100, and 300 classes using 2D keypoint coordinates (x and y). The keypoints were extracted using OpenPose. Since the model did not have a decoder, it greatly reduced the complexity of the model. Experiments were performed with 1,2,3,4,6 and 9 attention heads, but the number of encoders layers remained the same. The model was evaluated on top-1, top-5, and top-10 mean accuracies. The model achieved a 96.88% accuracy on 10 signs, 87.22% accuracy on 50 signs, 83.16% accuracy on 100 signs, and 70.52% accuracy on 300 signs when evaluated on top-1 accuracy.

2.7 Critical Analysis

From the papers discussed above,

[5] proposed Pose Based SLR using GCN and BERT which achieved an accuracy of 88.67% when evaluated on the WLASL dataset. However, the model has difficulty distinguishing between signs that exhibit subtle differences. The subtle rotation of the signer in the frame makes the poses for the words 'black' and 'candy' look similar, hence making it challenging for the model to differentiate between them. The model relies solely on 2D spatial information and does not differentiate well between in-plane and out-of-plane movements. This limitation suggests that the model may not fully capture the three-dimensional aspects of the signing gestures.

[70] implemented the ResNet model and RNN + Bi-LSTM model which gave very good accuracies on evaluation, however, there were high occurrences of false detections due to inconsistency in the orientation and movement of the hand for the same sign in the dataset. Hence the quality of the dataset raises questions regarding the performances of the models.

[40] proposed a CapsNet mechanism and SKNet + attention mechanism to improve the recognition accuracy which was trained on a self-created dataset consisting of 4 words. The models gave an accuracy of 98.72% and 98.88% respectively. A ResNeXt model was also implemented which alone gave an accuracy of 97.82%. The main limitation of their work was the lack of training data.

A dataset has a significant contribution in the performance and accuracy of a deep learning model. In the above-mentioned literature reviews, it is noticeable that many state either the size or the quality of the datasets as a limitation to their work. A few have used self-created datasets to train and test their proposed models however, these datasets are small and contain limited

classes. This is why the WLASL dataset will be used in this paper to train the implemented models. It is a publicly available dataset containing more than 12K videos, and covering a diverse range of American Sign Language (ASL) vocabulary.

On performing a comparative study on the pros and cons of deep learning models, the following is understood:

Although CNNs excel in capturing local features from images and possess the ability to autonomously learn features from raw data without prior knowledge, they fail to understand the correlation among feature attributes such as relative positions, size, direction of the feature, etc. [40]. Hence, the CapsNet model, first proposed by Sabour [53], is much more efficient for image processing. It integrates the advantages of CNN structure while taking into account factors such as the relative position, angle, and other information that are neglected by CNN, thereby improving recognition. [40].

On the other hand, ResNet offers a distinct advantage over other architectural models like CNNs, as its performance remains stable even with increased depth. The skip connections in ResNet address the vanishing gradient problem commonly encountered in deep neural networks. They provide a shortcut path for the gradient to flow through, making it easier for the information to pass and preventing it from disappearing [45].

RNNs are highly capable of processing sequential data and handling variable-length sequences, making them flexible for real-world applications, however, here are a few drawbacks of standard RNNs [75]:

- Limited ability to capture global dependencies: RNNs face challenges in comprehending overall relationships and can only model actions in a sequential manner. They can only

access information from preceding elements in the sequence.

- Gradient vanishing – A potential issue during training with backpropagation arises as RNNs may experience gradient vanishing when processing relatively long sequences.
- Don't support parallelism – RNNs lack the capability for parallel training on GPUs, and hence require longer training times.

LSTM is an RNN architecture that addresses the vanishing gradient issue present in standard RNNs. It can capture and remember long-term dependencies within sequences making it a fitting choice for sequence-based tasks [7]. Gating mechanisms in LSTMs allow better control over information flow.

When processing the temporal features of a video, each element within the input sequence contains information from both the past and the future. Hence the bi-directional processing of input data in Bi-LSTMs allows them to capture information and improve their understandings of both past and future contexts simultaneously by combining the outputs of the LSTM layers from both directions [76].

Transformers are a type of neural network architecture that rely heavily on attention mechanisms. The key innovation in the original Transformer model, as introduced in [67], was the self-attention mechanism. Vision Transformers (ViT) are a specific application of the Transformer architecture to computer vision tasks. They adapt the Transformer's self-attention mechanism to process 2D image data by applying self-attention to patches of an image, which are input tokens to the model. They have shown excellent performance in image classification tasks as mentioned in [3]. While there are many advantages of a Transformer, transformer models have long training times which further vary depending on the size of the dataset it is training on,

hardware, and batch size [50]. The concept of Early Stopping applied to reduce the training time, can serve as a solution. Instead of training the model for a fixed number of epochs, the model's performance on the validation dataset is monitored and training is halted when the performance begins to degrade. This approach not only saves time but also computational resources while obtaining a reasonably well-performing model [50].

When compared with the Inception model in terms of size, both the Xception and Inception models have nearly the same number of parameters. Hence, the improvement in the model performance on the ImageNet and JFT datasets in [12] indicates better use of model parameters in the Xception model. The residual connections in the Xception model aid in the convergence, in terms of speed and better classification performance.

To conclude, MediaPipe offers components for tracking hand movements, body poses and facial expressions, making working with sign language gestures easier. ResNet models offer lighter computation calculations and enhanced network training capabilities compared to other architectural models [45]. The bidirectional nature of Bi-LSTMs allows them to consider information from past and future time steps, making them capable of capturing complex temporal patterns. The Xception model hasn't been used much for Isolated Sign Language Recognition. Although, there have been many works published on using the model for Static Sign Language Recognition, which is why I will be using the model for my implementation. Furthermore, since a hybrid ResNet + Bi-LSTM model hasn't been implemented and evaluated on the WLASL dataset, this model combination will be implemented in this paper for Sign Language Recognition. Further details regarding the implementation are included in Chapter 3.

Chapter 3

Implementation

This section begins with the requirements analysis, followed by the datasets used in the project and the programming environment. Then comes the implementation section that describes the implementation and workflow of the project according to the aims and objectives mentioned earlier.

3.1 Requirements Analysis

This section highlights the functional requirements (shown in Table 3.1) and non-function requirements (shown in Table 3.2). The prioritization technique used is MoSCoW, where M, S, C, and W indicate Must have, Should have, Could have, and Would have respectively.

3.1.1 Functional Requirements

Sno.	Requirement Description	Priority
FR-1	**Finding a suitable dataset for the project.** The dataset must contain short video clips of signers performing American Sign Language. The videos must be of good resolution and the singers must be visible at least up to the torso.	M
FR-2	**Frame Extraction from videos.** Frame extraction must be performed from videos to facilitate the extraction of spatial and temporal features of the video.	M
FR-3	**Feature Enhancement** Feature Enhancement techniques should be performed to enhance the features of the video frames.	M
FR-4	**Feature Extraction** Feature Extraction must be performed to extract the spatial and temporal features.	M
FR-5	**Selection of an appropriate Deep Learning Algorithm.** Appropriate algorithms must be used (which may or may not be combined to create a hybrid model) for Sign Language Recognition.	M
FR-6	**Recognition of Sign Language and Classification** The model must be able to recognize the sign language performed in the video and classify it accordingly.	M
FR-7	**Testing of Model and Evaluation** The models should be tested with the test dataset and their performances must be evaluated.	M
FR-8	**Comparing the performances of Models** The performances of the model should be compared with other models trained on the same dataset.	M

Table 3.1: Functional Requirements

3.1.2 Non-Functional Requirements

Sno.	Requirement Description	Priority
NFR-1	**Code Quality** The code must be readable and understandable. The logic of the code should be explained with well-written comments.	M
NFR-2	**Version Control** The code must be pushed to GitHub with progressive commits. Any changes to the code should be documented in the commit message.	M
NFR-3	**Code Cleaning and Optimization** Any repetitive lines of code should be replaced with reusable code. Code should be optimized to use work more efficiently using fewer resources.	S
FR-4	**Optimization of Model Performance** The model should be optimized to improve the accuracy of gloss recognition and classification, lowering the error rate.	S

Table 3.2: Non-Functional Requirements

3.2 Dataset

The dataset that is used for training and testing purposes is the World Level American Sign Language (WLASL) video dataset [8] which is publicly available on Kaggle. The preprocessed dataset contains twelve thousand videos of the WLASL dataset [21], which is a comprehensive collection of annotated video samples encompassing various American Sign Language vocabulary. It was created to facilitate research and development of American Sign Language Recognition systems [21].

This project will be using glosses from the WLASL dataset to train the models. Gloss is another term for the word that is being enacted in the video using hand signs and facial expressions.

FIGURE 3.1: A glimpse of the ASL dataset[21]

3.3 Programming Environment

This project has been implemented in Python programming language. Jupyter Notebook was used to write and run the code and develop the deep learning models. The code was run on an Intel core i9 - 13900H with 32 GB RAM, paired with NVIDIA graphics support GeForce RTX 4060.

3.4 Source Code

Throughout the implementation, progressive commits were made to GitHub to back up the latest version of the code. The dataset along with the ResNet50 and Xception models could not be pushed to the repository due to the file size restriction, hence, they were backed locally on a hard disk. The hybrid models namely ResNet50 + BiLSTM and Xception + BiLSTM are uploaded under the 'Logs' folder in the repository. A ReadMe file has been attached to give a brief description of the project overview and the file structure. The link to the repository is mentioned below:

https://github.com/GauriR011/ASL-Detection-using-Deep-Learning

3.5 Project Implementation Workflow

Figure 3.3 depicts the project implementation workflow:

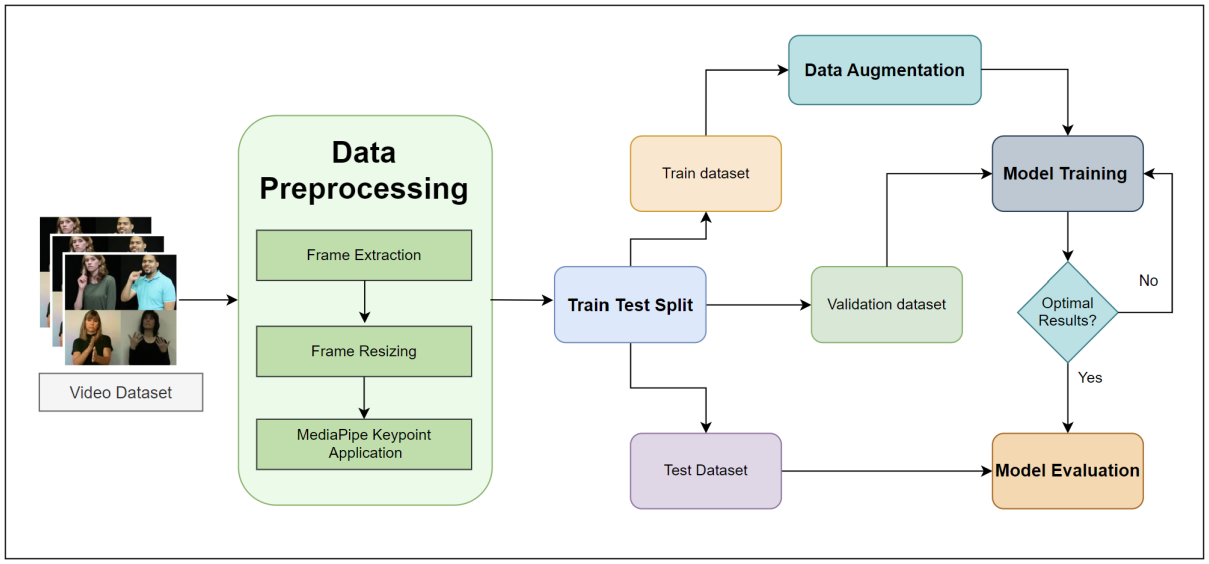

FIGURE 3.2: Overview of the Project Implementation workflow

3.6 Data Preprocessing

The process of preparing the dataset can be broken down into 3 simple steps:

1. Frame Extraction
2. Frame Resizing
3. Application of MediaPipe Holistic keypoints

This implementation uses a subset of the WLASL dataset. The top 10 glosses were selected based on the number of videos per class. The total number of videos under each class in the dataset were counted. The top 10 classes with the highest number of videos under them were selected. The following classes of videos are used for the implementation – 'before', 'computer',

'cool', 'cousin', 'drink', 'go', 'help', 'take', 'thin', and 'who'. On average, there are around 11 videos per class.

3.6.1 Frame Extraction

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

```
# This function determines the frames to be skipped to extract frames from videos of variable lengths.
def frameSkipping(DATA_PATH, video_paths, n):
    for action in video_paths.keys():

        #Looping through each video of the action
        for sequence in range(len(video_paths[action])):
            cap = cv2.VideoCapture(video_paths[action][sequence])

            total_frames = cap.get(cv2.CAP_PROP_FRAME_COUNT)  #getting the total frame count
            frames_step = total_frames//n    # getting the number of frames to skip

            for frame_num in range(n):
                #position the pointer to the frame after skipping frames
                cap.set(1,frame_num*frames_step)
                success,image = cap.read()
                #save the image
                cv2.imwrite(os.path.join(DATA_PATH, action, str(sequence), 'frame'+ str(frame_num) + '.jpg'), image)

            cap.release()
```

FIGURE 3.3: Code for frame skipping for frame extraction

3.6.2 Frame Resizing

The process of frame extraction was followed by resizing the images. Pretrained models such as ResNet50 and Xception model, were trained on the ImageNet dataset that included images of size 224 x 224 pixels. Hence the video frames needed to be resized to meet the requirements of the model input structure. A center square was cropped out of the image to remove the extra background margins and only include the signer in the frame. Cropping the center square

helped preserve the quality of the frames as images often tend to become blurred while resizing due to loss of pixels. The Open CV library was used to resize the video frames.

FIGURE 3.4: Before Resizing - 1920 x 1080 pixels

FIGURE 3.5: After Resizing – 224 x 224 pixels

```
# Cropping the center of the image (cropping out the extra background margins of the video.)
def crop_center_square(frame): # takes image as a parameter
    y, x = frame.shape[0:2]
    min_dim = min(y, x)
    start_x = (x // 2) - (min_dim // 2)
    start_y = (y // 2) - (min_dim // 2)
    return frame[start_y:start_y+min_dim,start_x:start_x+min_dim]
```

FIGURE 3.6: Code for cropping the center square from the video frames

3.6.3 Applying MediaPipe Holistic

After resizing the video frames, MediaPipe Holistic landmarks were applied to the video frames. Only pose and hand landmarks were applied. MediaPipe Landmarks help to analyze and track the body and hand movements of the signer. The landmark drawing specifications for thickness and circle radius were set to 1.

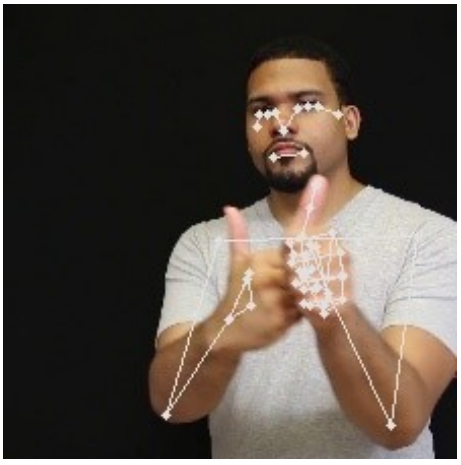

FIGURE 3.7: Video frame after applying MediaPipe keypoints

As of now, our dataset has 151 videos in total, each video containing 22 video frames per video, with each video frame of size 224 x 224 in RGB format. The structure of the dataset is as follows:

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

FIGURE 3.8: Shape of the dataset

3.7 Splitting the Dataset into Train, Validation and Test datasets

To train and evaluate the performance of the models, the train test split module of the sklearn library was used to split the dataset into train, validation, and test datasets in an 70:15:15 ratio. The random state value to split the dataset was set to 42.

Stratification was applied while splitting the datasets to ensure an equal number of videos from each class were used for model evaluation. The total number of records under train, validation, and test datasets are 105, 23, and 23 respectively.

```
# Splitting data in 70:15:15 train:validation:test ratio
def trainTestSplit(X,y):
    x_train, x_temp, y_train, y_temp= train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)
    x_val, x_test, y_val, y_test = train_test_split(x_temp, y_temp, test_size = 0.5, random_state=42, stratify=y_temp)
    return x_train, y_train, x_val, y_val, x_test, y_test
```

FIGURE 3.9: Code for cropping the center square from the video frames

3.8 Data Augmentation

To increase the number of training video samples in the dataset, video augmentation was used. The Image and Image Enhance modules of the PIL library were used to augment the videos. The following video augmentation methods were used:

1. Frame mirroring, that is, flipping the video frames horizontally.
2. Rotating the video frames (Frames rotated clockwise by 8 degrees).

FIGURE 3.10: Original Frame

Horizontal flip

Rotating the frame

FIGURE 3.11: Effects of Video frame augmentation on the video frames

The process of data augmentation increased the number of training videos to up to 30 to 33 videos per class or gloss.

So, our final dataset contains 315 videos for training, 23 videos for validation, and 23 videos for testing purposes. Now we proceed to train our models.

3.9 Building Models

Model Checkpoint and Early stopping modules of the Keras library were used to save the weights of the best model and stop the model training if performance did not improve after a

fixed number of epochs.

4 models were developed to train on the dataset:

1. **Pre-trained ResNet50 model**

   Below is the model architecture:

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 22, 224, 224, 3)] 0
_________________________________________________________________
time_distributed (TimeDistri (None, 22, 7, 7, 2048)    23587712
_________________________________________________________________
time_distributed_1 (TimeDist (None, 22, 2048)          0
_________________________________________________________________
flatten (Flatten)            (None, 45056)             0
_________________________________________________________________
dense (Dense)                (None, 10)                450570
=================================================================
Total params: 24,038,282
Trainable params: 450,570
Non-trainable params: 23,587,712
_________________________________________________________________
```

FIGURE 3.12: ResNet50 Architecture

   The input shape of the pre-trained model is set to (224, 224, 3). The input passed to the ResNet model is of the shape (Number of videos, 22, 224, 224, 3). The Time Distribution layer facilitates the processing of sequential data and enables the ResNet model to process each frame of the video individually. The output of the first time distribution layer results in a feature map of size 7 x 7 with 2048 channels, for each frame of a video. The second time distribution layer further reduces the special dimensions to (22, 2048). The output is flattened before passing to the dense layer. The dense layer or fully connected layer contains 10 neurons for the 10 output classes.

   To preserve the weights of the pre-trained model, all the weights of all the layers of the pre-trained model are frozen and the topmost layer of the model is excluded (since there are 10 output classes).

```python
resnet_model = tf.keras.applications.ResNet50(weights='imagenet', include_top=False, input_shape=(224, 224, 3), pooling="avg")

# Freeze the ResNet50 layers
for layer in resnet_model.layers:
    layer.trainable = False
```

FIGURE 3.13: Code snippet for freezing layers

2. **ResNet50 + BiLSTM model**

This hybrid model structure uses the ResNet50 model as a feature extractor for all the videos in the dataset. Each video is passed frame by frame through the model for feature extraction and the extracted features are stored in a CSV file (7942 rows [361 videos x 22 frames] and 2048 columns (features extracted)).

```
def fe_PretrainedM(df, pretrained_model):
    fe_video = []
    count = 1
    for frame in df:
            features = extract_features(frame, pretrained_model)
            fe_video.append(features)
            print(count)
            count+=1

    # return fe_video, new_y
    return fe_video
```

```
def extract_features(frame, model):
    img = np.expand_dims(frame, axis=0)  # Add batch dimension
    features = model.predict(img, verbose = 1)  # Extract features
    return features.flatten()  # Flatten the features
```

FIGURE 3.14: Feature Extraction code snippet

After extracting the features, the CSV file is read and every 22 frames are grouped together to be stored as a video. Hence the final shape of the complete dataset is (361, 22, 2048). The shape of the training set is (315, 22, 2048), the validation set is (23, 22, 2048), and the test set is (23, 22, 2048). Now, the BiLSTM model is ready to train on and be evaluated on these datasets.

```python
# grouping every 22 frames to form  a video
def frames_to_video(X_arr):
    fr_count = 0
    X_df = []
    video = []
    max_frames = X_arr.shape[0]

    while(fr_count<max_frames):
        for i in range(22):
            video.append(X_arr[fr_count])
            fr_count+=1
        X_df.append(video)
        video = []

    return X_df
```

FIGURE 3.15: Grouping frames to create videos

Below is the architecture of the BiLSTM model:

```
Model: "model_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_9 (InputLayer)         [(None, 22, 2048)]        0
_________________________________________________________________
bidirectional (Bidirectional (None, 22, 128)           1081856
_________________________________________________________________
bidirectional_1 (Bidirection (None, 22, 256)           263168
_________________________________________________________________
bidirectional_2 (Bidirection (None, 64)                73984
_________________________________________________________________
dense_4 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_5 (Dense)              (None, 64)                4160
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_6 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_7 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_8 (Dense)              (None, 10)                330
=================================================================
Total params: 1,433,898
Trainable params: 1,433,898
Non-trainable params: 0
_________________________________________________________________
```

FIGURE 3.16: ResNet50 + BiLSTM Architecture

There are 3 BiLSTM layers with 64, 128, and 32 neurons respectively. These layers are followed by 4 dense layers with 64, 64, 64, and 32 neurons respectively. These layers have a ReLU activation function applied and l2 kernel regularizer with the value set to 0.001. The l2 regularizer helps in controlling the complexity of the model and improves its generalization to new data. Finally, a dense layer with softmax function applied to it converges the model architecture to 10 output classes.

Below is a diagram to describe the process of feature extraction using a pre-trained model (ResNet50) and how the data is passed to the BiLSTM model to train:

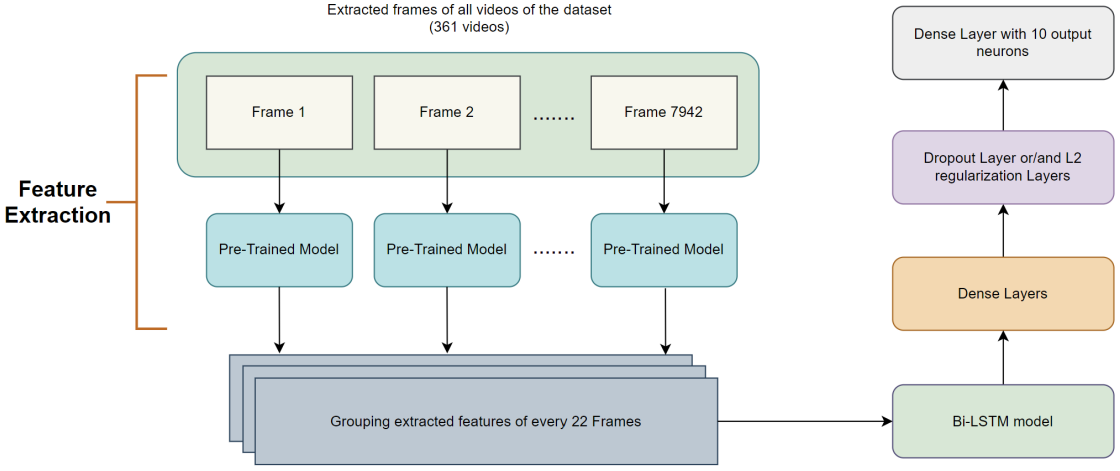

FIGURE 3.17: Multi-layer system architecture using ResNet50 and BiLSTM

3. **Pre-trained Xception model**

Below is the Xception model architecture:

```
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_6 (InputLayer)         [(None, 22, 224, 224, 3)] 0
_________________________________________________________________
time_distributed_4 (TimeDist (None, 22, 7, 7, 2048)    20861480
_________________________________________________________________
time_distributed_5 (TimeDist (None, 22, 2048)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 45056)             0
_________________________________________________________________
dropout_1 (Dropout)          (None, 45056)             0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                450570
=================================================================
Total params: 21,312,050
Trainable params: 450,570
Non-trainable params: 20,861,480
_________________________________________________________________
```

FIGURE 3.18: Xception Architecture

Just like the ResNet 50 model, all the weights of all the layers of the pre-trained model are frozen and the topmost layer of the model is excluded. The model follows an architecture similar to the ResNet model, with an addition of a dropout layer with a dropout value of 0.5.

4. **Xception + BiLSTM model**

The process is the same as the ResNet50 + BiLSTM model. Xception model is used as a feature extractor and the output is stored in a csv file. This csv file is read and the records are passed to the BiLSTM model for classification. The architecture of the model is as follows:

```
Model: "model_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_9 (InputLayer)         [(None, 22, 2048)]        0
_________________________________________________________________
bidirectional (Bidirectional (None, 22, 64)            532736
_________________________________________________________________
bidirectional_1 (Bidirection (None, 32)                10368
_________________________________________________________________
dense_3 (Dense)              (None, 32)                1056
_________________________________________________________________
dropout_2 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_4 (Dense)              (None, 16)                528
_________________________________________________________________
dropout_3 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_5 (Dense)              (None, 10)                170
=================================================================
Total params: 544,858
Trainable params: 544,858
Non-trainable params: 0
_________________________________________________________________
```

FIGURE 3.19: Xception + BiLSTM Architecture

The architecture consists of 2 layers of BiLSTM with 32 and 16 neurons respectively. These layers are followed by 2 dense layers with 32 and 16 neurons respectively with relu activation function applied to them. 2 dropout layers were added to the model with 0.35 and 0.3 dropout values respectively. Finally, an dense layer was added with 10 neurons.

Chapter 4

Model Evaluation and Testing

4.1 Model Performance

The 4 models were compiled as follows:

1. **ResNet 50 model**

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

   Following are the accuracy and loss graphs of the model while training:

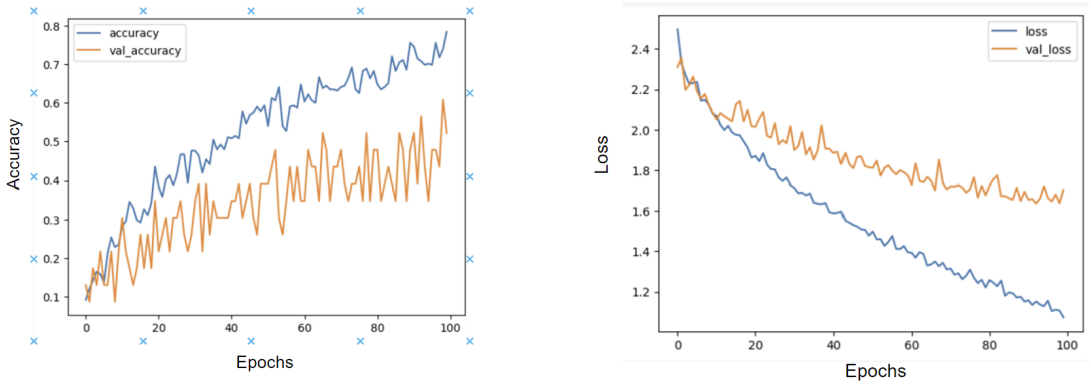

FIGURE 4.1: Accuracy and loss graphs of ResNet50

   Following are the confusion matrices for validation and test datasets:

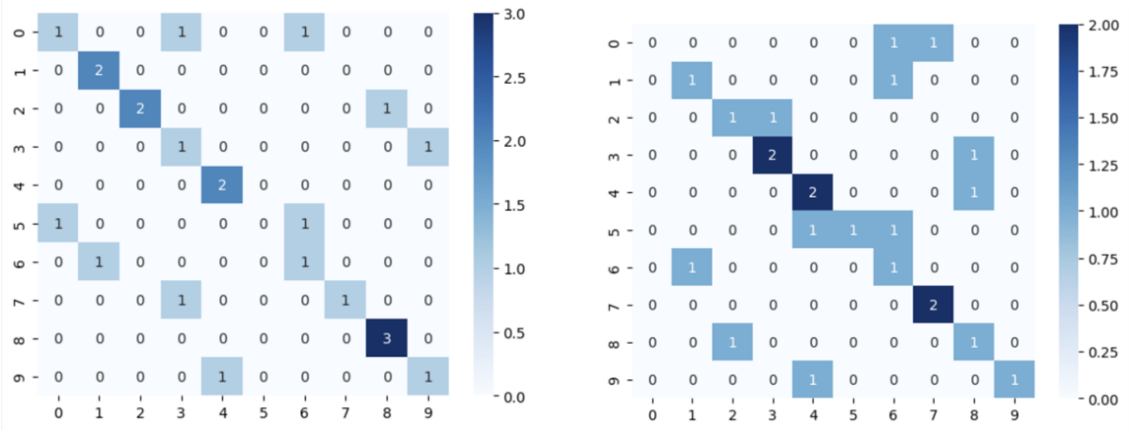

FIGURE 4.2: Confusion matrices for validation and test datasets for ResNet50 model

Following are the scores of the evaluation criteria used:

```
Validation Accuracy: 0.6086956521739131
Test Accuracy: 0.5217391304347826
Validation Precision: 0.5978260869565217
Test Precision: 0.5652173913043478
Validation Recall: 0.6086956521739131
Test Recall: 0.5217391304347826
Validation F1-score: 0.5784679089026915
Test F1-score: 0.504968944099379
```

FIGURE 4.3: Evaluation metrics scores for ResNet50 model

2. **ResNet50 + BiLSTM**

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

Following are the accuracy and loss graphs of the model while training:

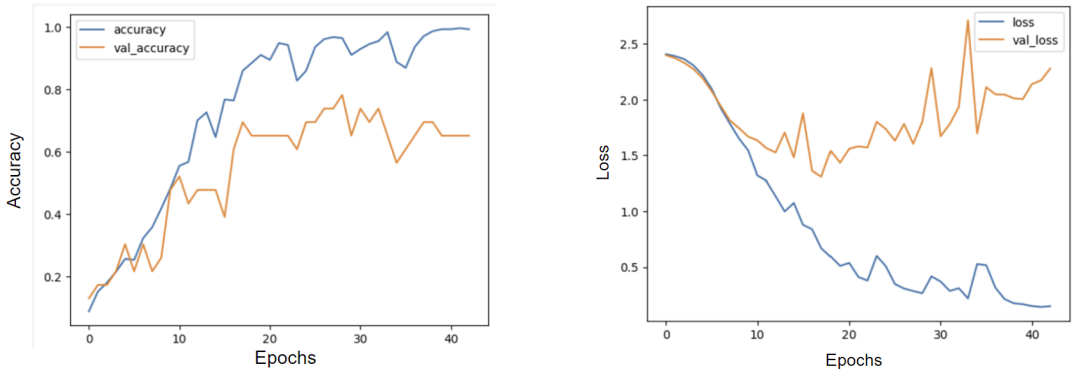

FIGURE 4.4: Accuracy and loss graphs of ResNet50 + BiLSTM

Following are the confusion matrices for validation and test datasets:

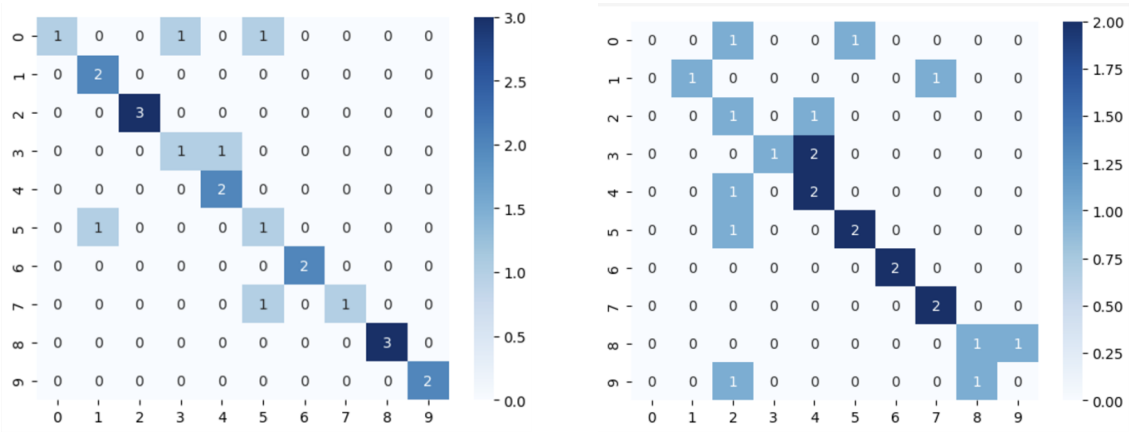

FIGURE 4.5: Confusion matrices for validation and test datasets for ResNet50 + BiLSTM

Following are the scores of the evaluation criteria used:

```
Validation Accuracy: 0.6956521739130435
Test Accuracy: 0.5217391304347826
Validation Precision: 0.8173913043478261
Test Precision: 0.6028985507246377
Validation Recall: 0.6956521739130435
Test Recall: 0.5217391304347826
Validation F1-score: 0.7004140786749481
Test F1-score: 0.5159420289855072
```

FIGURE 4.6: Evaluation metrics scores for ResNet50 + BiLSTM model

3. **Xception**

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

Following are the accuracy and loss graphs of the model while training:

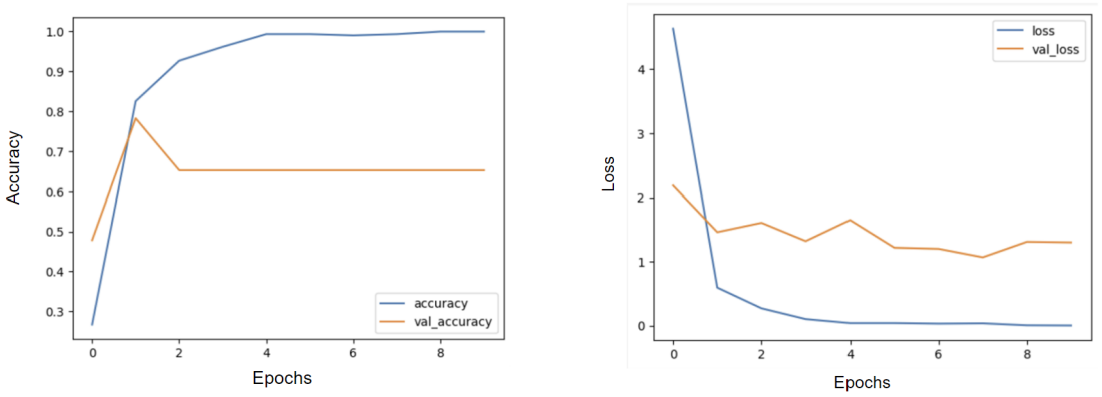

FIGURE 4.7: Accuracy and loss graphs of Xception model

Following are the confusion matrices for validation and test datasets:

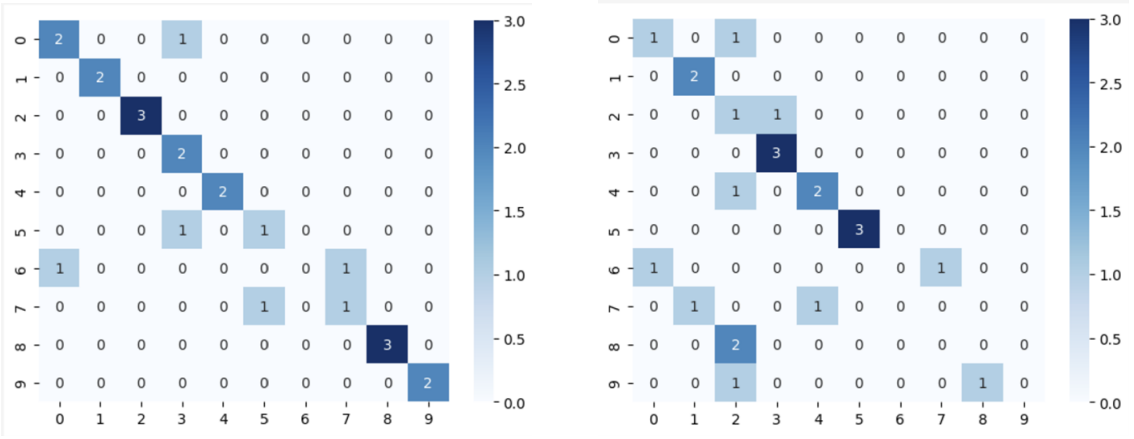

FIGURE 4.8: Confusion matrices for validation and test datasets for Xception model

Following are the scores of the evaluation criteria used:

```
Validation Accuracy: 0.782608695652174
Test Accuracy: 0.5217391304347826
Validation Precision: 0.7391304347826086
Test Precision: 0.43115942028985504
Validation Recall: 0.782608695652174
Test Recall: 0.5217391304347826
Validation F1-score: 0.753623188405797
Test F1-score: 0.46397515527950306
```

FIGURE 4.9: Evaluation metrics scores for Xception model

4. **Xception + BiLSTM**

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

Following are the accuracy and loss graphs of the model while training:

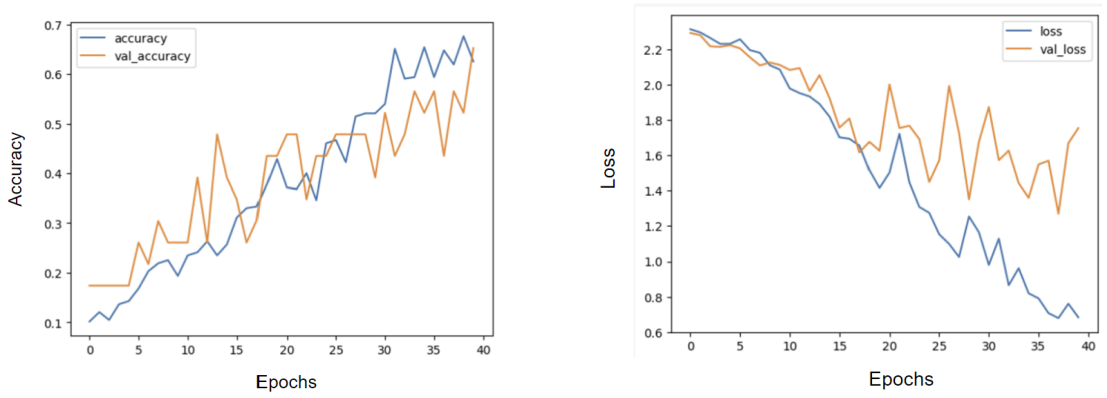

FIGURE 4.10: Accuracy and loss graphs of Xception + BiLSTM [43]

Following are the confusion matrices for validation and test datasets:

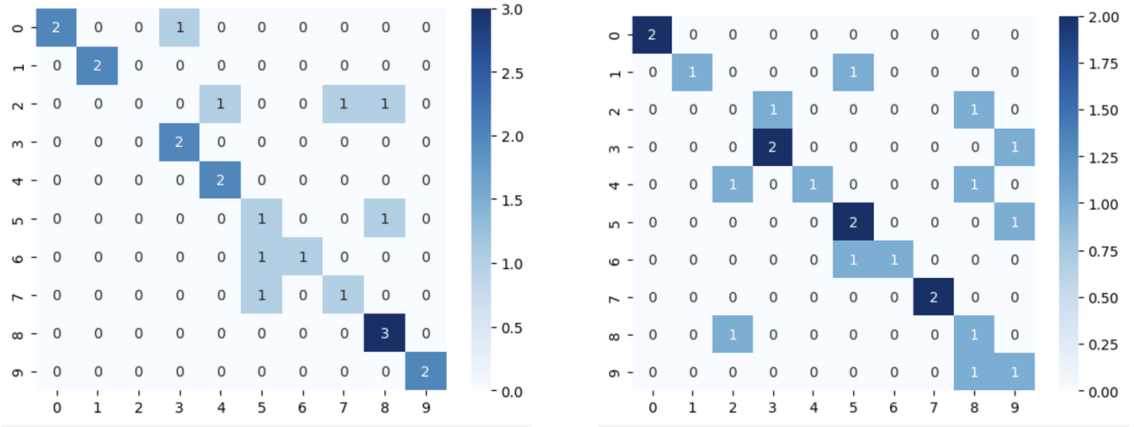

FIGURE 4.11: Confusion matrices for validation and test datasets for Xception + BiLSTM

Following are the scores of the evaluation criteria used:

```
Validation Accuracy: 0.6956521739130435
Test Accuracy: 0.5652173913043478
Validation Precision: 0.6579710144927536
Test Precision: 0.6811594202898551
Validation Recall: 0.6956521739130435
Test Recall: 0.5652173913043478
Validation F1-score: 0.6514492753623189
Test F1-score: 0.5803312629399585
```

FIGURE 4.12: Evaluation metrics scores for Xception + BiLSTM model

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

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

TABLE 4.1: Comparing Train, Validation and Test Accuracies

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

TABLE 4.2: Comparing Precision and Recall

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

TABLE 4.3: Comparing F1 Scores

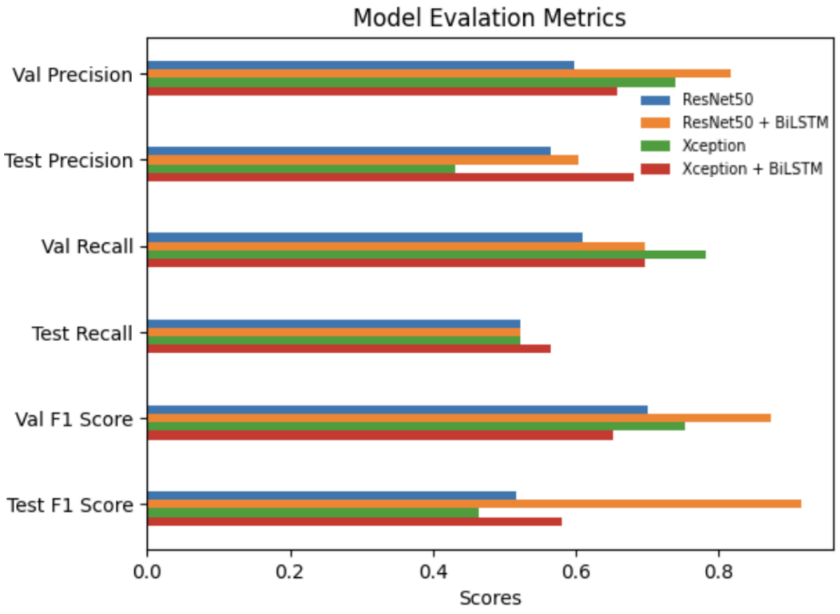

FIGURE 4.13: Model Evaluation Metrics graph

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

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

4.2 Comparative Analysis

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

Sno.	Model	Train Accuracy	Val Accuracy	Test Accuracy
1.	Encoder-Only Transformers [72]	100.00%	95.61%	93.46%
2.	Media Pipe + Self-attention [75]	80.00%	59.00%	-
3.	ResNet50 (Ours)	73.97%	60.86%	52.17%
4.	ResNet50 + BiLSTM (Ours)	79.37%	69.56%	52.17%
5.	Xception (Ours)	82.54%	78.26%	52.17%
6.	Xception + BiLSTM (Ours)	66.89%	69.56%	56.52%

TABLE 4.4: Comparative Analysis based on Train, Validation and Test accuracies on 10 classes WLASL subset

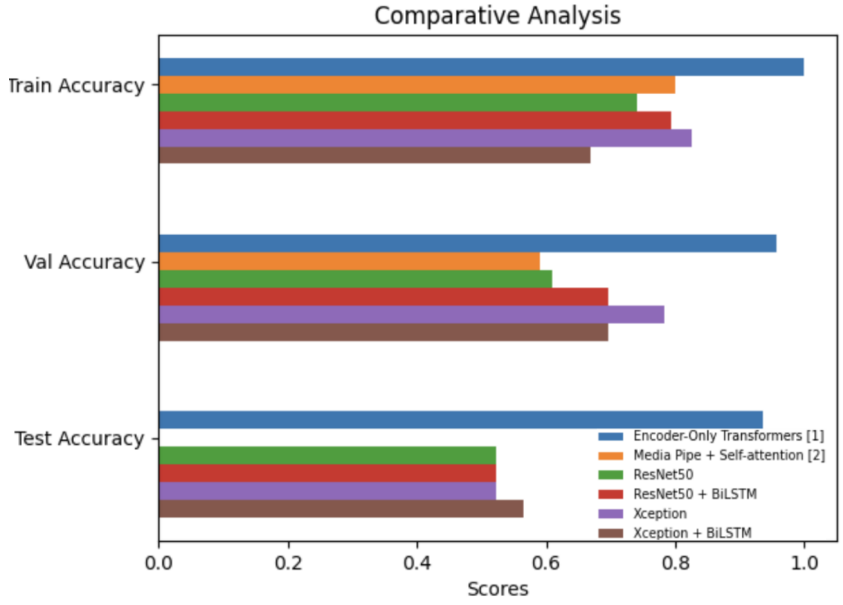

FIGURE 4.14: Comparative Analysis graph

The Encoder-only Transformer achieved this performance with 1 encoder layer with 1 attention head. [72] have used Open Pose to extract pose and hand 2D keypoints (x and y coordinates) from the video frames. These keypoints were later normalized before being passed to the model. The model had the architecture of a traditional transformer but it was modified to utilize only the encoder part. Positional encodings were applied to mark the sequence positions of the input. During model training and evaluation, random batching was applied to ensure every sample of

the dataset split has been processed by the model at least once. The model was run for 200 epochs. Experiments were carried out using 1,2,3,4,6 and 9 attention heads while maintaining the same number of encoder layers.

However, it is to be noted that [72] trained its model on the enhanced version of WLASL dataset, containing improved and enhanced classes/glosses. Hence the comparison of the performances is only indicative, which means that the authors acknowledge that using an improved version of the dataset may affect the validity of direct comparisons. Moreover, the model was trained over 200 records when no data augmentation techniques were used to increase the number of training records. This indicates that the improved version of the WLASL may contain more videos under each class.

[75] proposed the use of Self-Attention mechanism for SLR. Media Pipe was used to extract keypoints from the videos. The self-attention layer was used to extract spatiotemporal features from the data. A sequence of these keypoints served as an input to the model. Experiments were carried out with and without the face mesh as well as with different sequence lengths - 30, 60, 100, and 140. The validation accuracy mentioned in the table above was observed after excluding the face mesh and by adopting a sequence length of 60. A validation accuracy of 40.4% was achieved with face mesh. The model was trained for 2000 epochs. Although the authors haven't mentioned the training accuracy of the model explicitly anywhere in their research paper, the value was inferred from the training and validation accuracy plot of the Self Attention model. The number of heads in the multi-head attention was set to 8. No data augmentation techniques were performed to increase the training data. The large gap in the training and validation accuracies shows signs of overfitting.

Although our models performed better than the Self Attention model approach proposed by [75] in terms of validation accuracy, they still are overfitting and do not generalize very well to new data. The Encoder-only Transformer does surpass our models by a large margin in terms of the accuracies recorded.

Chapter 5

Conclusion and Future Work

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

The following sections describe the challenges faced during the implementation, the limitations of the implementation, and possible future work.

5.1 Challenges faced

A few challenges were faced during the implementation of the project. Creating the hybrid model architecture was the first. Initially, the hybrid model architecture had layers of the pre-trained model followed by BiLSTM and dense layers. This architecture experienced a large drop in training and validation accuracies. After much research on the implementation of building a hybrid model, the procedure of using the pre-trained model as a feature extractor

and passing these extracted features to the BiLSTM model was adopted. Hyperparameter tuning of the structured model was another challenge. Hyperparameter tuning algorithms such as random search and grid first search could not be used to tune the hyperparameters since the implementation dealt with deep learning models. Using these algorithms would be very computationally expensive and would also take a long time. Hence, hyperparameter tuning was manually performed through the process of trial and error. Attempts to train the models on the 50 gloss subset of WLASL dataset (WLASL-50) were made to be able to compare the proposed model with the model architectures proposed in other research papers. But, due to the limitation of computational resources, it hindered the training process of the models.

5.2 Limitations and Future Work

A possible limitation of the project may be the small number of videos per class in the dataset. Although it is impressive that the dataset covers vocabulary up to 2000 words, the number of videos available under each gloss or class averages to 11. Hence, various video augmentation techniques were adopted to increase the number of videos under each class and, in turn, increase the size of the dataset. Another limitation of the chosen dataset is that many a time, 2 different glosses or classes contained the same sign videos. For example, the classes 'like' and 'some' had a few videos in common in their respective folders. This can have a serious impact on the performances of the models. Therefore, the subset was selected after thoroughly checking the videos of the classes and making sure they were unique to their respective classes. One potential limitation concerning the trained models may be the indications of overfitting. They also struggle to generalize well to new data. This, again, could be due to the lack of data to train the models.

As a part of future work, we aim to use the enhanced version of the WLASL dataset (WLASL-alt dataset) as suggested by [72]. It contains the corrected and improved versions of the glosses present in the original WLASL dataset proposed by [21]. In addition to using the WLASL-alt dataset, we aim to train the Xception models on larger subsets of the dataset, which are - WLASL-100, WLASL-300, WLASL-1000, and WLASL-2000 and compare their performances with the models proposed by other researchers.

Appendix A

Project Management

A.1 Proposed Methodology

The implementation will use MediaPipe Holistic for keypoint extraction from the raw videos from the dataset. MediaPipe Holistic comprises of three key components: MediaPipe Hands, MediaPipe Pose, and MediaPipe Face Mesh, hence providing real-time, instantaneous detection and tracking of hand, pose, and facial landmarks, making it a vital tool for SLR.

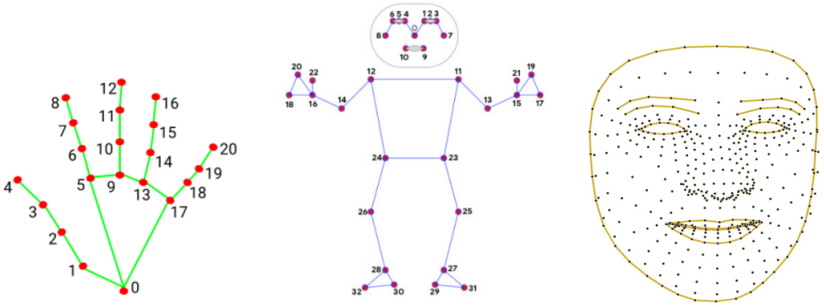

FIGURE A.1: Hand, Pose and Face Mesh key points extracted by MediaPipe [24]

The models to be implemented for the purpose of SLR will be hybrid models, to learn the spatial and temporal components of the dataset after feature extraction.

ResNet + Bi-LSTM hybrid model will be implemented for the purpose of SLR. ResNet will be used to learn the spatial features and Bi-LSTM will be used to learn the temporal features of

the video input. Optionally, Xception + Bi-LSTM hybrid model may also be implemented for the purpose of comparisons of the performances of the models.

A combination of ViT (Visual Transformer) + self-attention mechanism may be a secondary model implementation that may be optionally implemented. The original architecture of the Transformer model comprises two primary components - encoder and decoder. The encoder converts an input sequence into a hidden state, which the decoder then utilizes to generate an output sequence [54]. This work may use a Transformer model consisting of only the encoder component.

A.2 Project Plan

This section describes the timelines and the plan for the project implementation through Gantt Charts. A Gantt chart is a project management tool used to represent the schedule of a project visually. It helps keep note of tasks and their respective deadlines, allowing better tracking of the progress of the project. Agile methodology will be used for project management. Weekly meetings will be scheduled with the project supervisor for feedback on the progress made.

Figure A.1 below is a Gantt Chart showing the Project Plan for Semester 1 that was followed to implement Deliverable 1.

Figure A.2 is the tabular representation of the Gantt Chart with the tasks, their start and end dates, and the total duration.

FIGURE A.2: Gantt Chart for Semester 1

Name	Start Date	End Date	Duration
SPRINT 0: Choosing a dissertation topic and …	Sep 11, 2023	Sep 17, 2023	7 days
Deciding on the topic for dissertation	Sep 11, 2023	Sep 17, 2023	7 days
Choosing a supervisor for dissertation	Sep 14, 2023	Sep 14, 2023	1 day
SPRINT 1 (Week 2) : Background research on…	Sep 18, 2023	Sep 22, 2023	5 days
Meeting with supervisor (CHECK Journal)	Sep 18, 2023	Sep 18, 2023	1 day
Finalizing on topic	Sep 18, 2023	Sep 20, 2023	3 days
Doing a background study on topic	Sep 20, 2023	Sep 22, 2023	3 days
SPRINT 2 (Week 3): Going through research …	Sep 23, 2023	Oct 01, 2023	9 days
Going through research papers	Sep 23, 2023	Sep 30, 2023	8 days
Creating an excel table to summarize Resear…	Sep 23, 2023	Sep 30, 2023	8 days
Searching for an appropriate dataset	Sep 28, 2023	Oct 01, 2023	4 days
SPRINT 3 (Week 4) : Drafting Literature Revie…	Oct 02, 2023	Oct 08, 2023	7 days
Meeting with Supervisor	Oct 02, 2023	Oct 02, 2023	1 day
Writing Lit Reviews	Oct 03, 2023	Oct 08, 2023	6 days
SPRINT 4 (Week 5): Aim, Objective and Motiv…	Oct 09, 2023	Oct 14, 2023	6 days
Formulating Aim of Project	Oct 09, 2023	Oct 10, 2023	2 days
Formulating Objective of Project	Oct 11, 2023	Oct 12, 2023	2 days
Formulating Motivation of Project	Oct 13, 2023	Oct 14, 2023	2 days
Submission of Journal Entry 1	Oct 13, 2023	Oct 13, 2023	1 day
SPRINT 5 (Week 6): Introduction and Project …	Oct 16, 2023	Oct 22, 2023	7 days
Meeting with Supervisor	Oct 16, 2023	Oct 16, 2023	1 day
Implementing changes suggested during me…	Oct 17, 2023	Oct 18, 2023	2 days
Researching on Project Background	Oct 19, 2023	Oct 20, 2023	2 days
Preparing draft version of Section 1 (Introduc…	Oct 21, 2023	Oct 22, 2023	2 days
SPRINT 6 (Week 7): Draft Versions of Chapter…	Oct 23, 2023	Oct 28, 2023	6 days
Preparing draft version of Section 2 (Project …	Oct 23, 2023	Oct 25, 2023	3 days
Preparing draft version of Section 3 (Related …	Oct 25, 2023	Oct 27, 2023	3 days
Preparing draft version of Section 4 (Method…	Oct 27, 2023	Oct 28, 2023	2 days
Submission of Journal Entry 2	Oct 27, 2023	Oct 27, 2023	1 day

▾ SPRINT 7 (Week 8 and 9): Project Plan	Oct 31, 2023	Nov 15, 2023	16 days
Meeting with Supervisor	Oct 31, 2023	Oct 31, 2023	1 day
Implementing changes suggested during me…	Oct 31, 2023	Nov 08, 2023	9 days
Formulating a Project Plan (Gantt Chart)	Nov 09, 2023	Nov 15, 2023	7 days
Submission of Journal Entry 3	Nov 10, 2023	Nov 10, 2023	1 day
▾ SPRINT 8 (Week 10): Requirements Analysis	Nov 16, 2023	Nov 20, 2023	5 days
Meeting with Supervisor	Nov 16, 2023	Nov 16, 2023	1 day
Implementing changes suggested during me…	Nov 16, 2023	Nov 17, 2023	2 days
Working on Requirements Analysis section	Nov 18, 2023	Nov 20, 2023	3 days
▾ SPRINT 9 (Week 11): Risk Management	Nov 20, 2023	Nov 25, 2023	6 days
Working on Risk Management section	Nov 20, 2023	Nov 23, 2023	4 days
Meeting with Supervisor	Nov 23, 2023	Nov 23, 2023	1 day
Reviewing Sections 1, 2, 3, 4	Nov 24, 2023	Nov 25, 2023	2 days
▾ SPRINT 10 (Week 12): Final Review and Sub…	Nov 27, 2023	Nov 29, 2023	3 days
Meeting with Supervisor	Nov 27, 2023	Nov 27, 2023	1 day
Implementing changes suggested during me…	Nov 27, 2023	Nov 28, 2023	2 days
Final Review of Deliverable 1 document	Nov 28, 2023	Nov 28, 2023	1 day
Submission of Deliverable 1	Nov 29, 2023	Nov 29, 2023	1 day

FIGURE A.3: Tabular form of Gantt Chart for Semester 1

Figure A.3 below is a Gantt Chart showing the Project Plan for Semester 2 that will be followed to implement the project.

Figure A.4 is the tabular representation of the Gantt Chart with the tasks, their start and end dates, and the total duration.

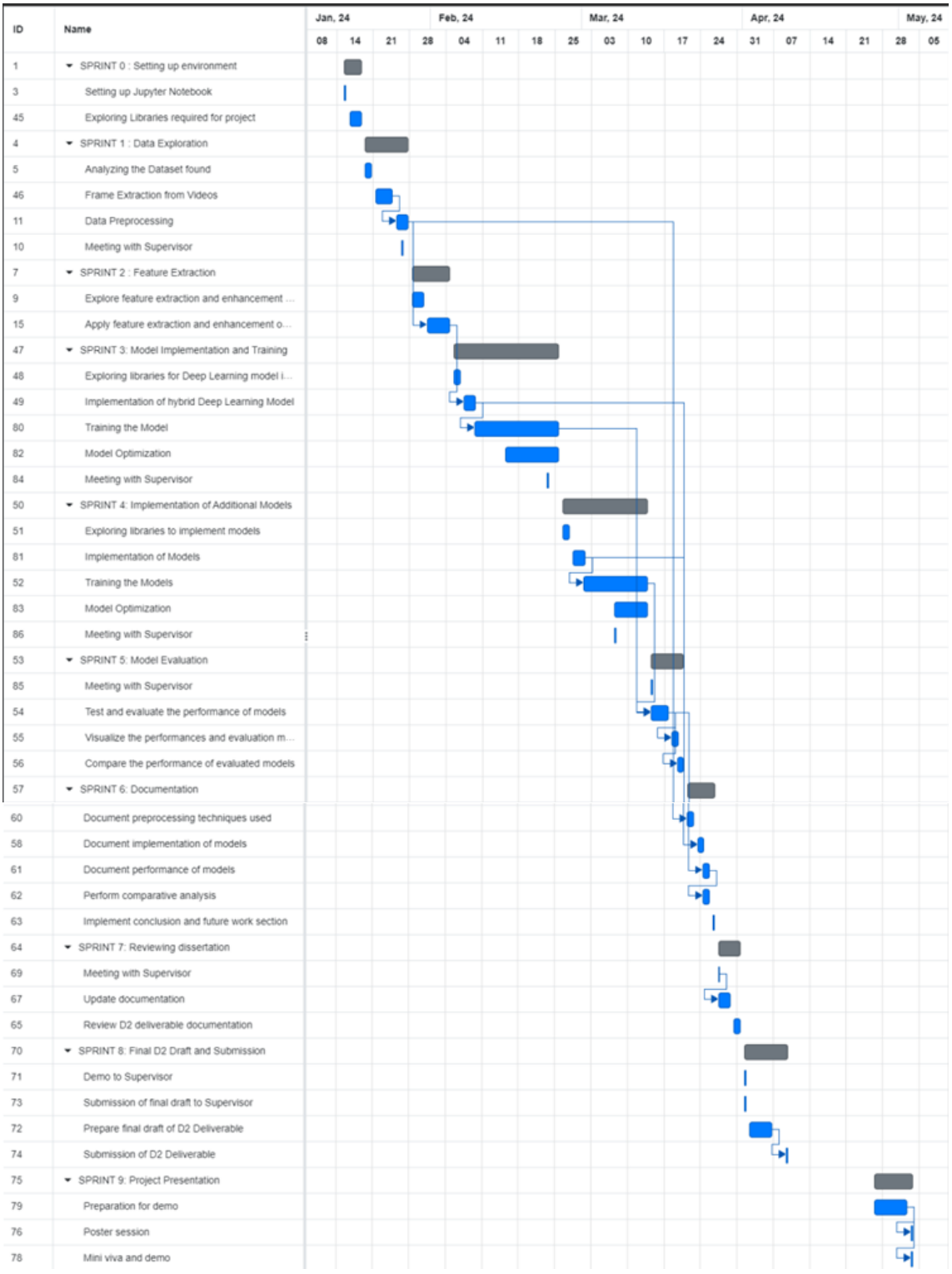

FIGURE A.4: Gantt Chart for Semester 2

Name	Start Date	End Date	Duration
SPRINT 0 : Setting up environment	Jan 15, 2024	Jan 18, 2024	4 days
Setting up Jupyter Notebook	Jan 15, 2024	Jan 15, 2024	1 day
Exploring Libraries required for project	Jan 16, 2024	Jan 18, 2024	3 days
SPRINT 1 : Data Exploration	Jan 19, 2024	Jan 27, 2024	9 days
Analyzing the Dataset found	Jan 19, 2024	Jan 20, 2024	2 days
Frame Extraction from Videos	Jan 21, 2024	Jan 24, 2024	4 days
Data Preprocessing	Jan 25, 2024	Jan 27, 2024	3 days
Meeting with Supervisor	Jan 26, 2024	Jan 26, 2024	1 day
SPRINT 2 : Feature Extraction	Jan 28, 2024	Feb 04, 2024	8 days
Explore feature extraction and enhancement ...	Jan 28, 2024	Jan 30, 2024	3 days
Apply feature extraction and enhancement o...	Jan 31, 2024	Feb 04, 2024	5 days
SPRINT 3: Model Implementation and Training	Feb 05, 2024	Feb 25, 2024	21 days
Exploring libraries for Deep Learning model i...	Feb 05, 2024	Feb 06, 2024	2 days
Implementation of hybrid Deep Learning Model	Feb 07, 2024	Feb 09, 2024	3 days
Training the Model	Feb 09, 2024	Feb 25, 2024	17 days
Model Optimization	Feb 15, 2024	Feb 25, 2024	11 days
Meeting with Supervisor	Feb 23, 2024	Feb 23, 2024	1 day
SPRINT 4: Implementation of Additional Models	Feb 26, 2024	Mar 13, 2024	17 days
Exploring libraries to implement models	Feb 26, 2024	Feb 27, 2024	2 days
Implementation of Models	Feb 28, 2024	Mar 01, 2024	3 days
Training the Models	Mar 01, 2024	Mar 13, 2024	13 days
Model Optimization	Mar 07, 2024	Mar 13, 2024	7 days
Meeting with Supervisor	Mar 07, 2024	Mar 07, 2024	1 day
SPRINT 5: Model Evaluation	Mar 14, 2024	Mar 20, 2024	7 days
Meeting with Supervisor	Mar 14, 2024	Mar 14, 2024	1 day
Test and evaluate the performance of models	Mar 14, 2024	Mar 17, 2024	4 days
Visualize the performances and evaluation m...	Mar 18, 2024	Mar 19, 2024	2 days
Compare the performance of evaluated models	Mar 19, 2024	Mar 20, 2024	2 days
SPRINT 6: Documentation	Mar 21, 2024	Mar 26, 2024	6 days
Document preprocessing techniques used	Mar 21, 2024	Mar 22, 2024	2 days
Document implementation of models	Mar 23, 2024	Mar 24, 2024	2 days

Document performance of models	Mar 24, 2024	Mar 25, 2024	2 days
Perform comparative analysis	Mar 24, 2024	Mar 25, 2024	2 days
Implement conclusion and future work section	Mar 26, 2024	Mar 26, 2024	1 day
▾ SPRINT 7: Reviewing dissertation	Mar 27, 2024	Mar 31, 2024	5 days
Meeting with Supervisor	Mar 27, 2024	Mar 27, 2024	1 day
Update documentation	Mar 27, 2024	Mar 29, 2024	3 days
Review D2 deliverable documentation	Mar 30, 2024	Mar 31, 2024	2 days
▾ SPRINT 8: Final D2 Draft and Submission	Apr 01, 2024	Apr 09, 2024	9 days
Demo to Supervisor	Apr 01, 2024	Apr 01, 2024	1 day
Submission of final draft to Supervisor	Apr 01, 2024	Apr 01, 2024	1 day
Prepare final draft of D2 Deliverable	Apr 02, 2024	Apr 06, 2024	5 days
Submission of D2 Deliverable	Apr 09, 2024	Apr 09, 2024	1 day
▾ SPRINT 9: Project Presentation	Apr 26, 2024	May 03, 2024	8 days
Preparation for demo	Apr 26, 2024	May 02, 2024	7 days
Poster session	May 03, 2024	May 03, 2024	1 day
Mini viva and demo	May 03, 2024	May 03, 2024	1 day

FIGURE A.5: Tabular form of Gantt Chart for Semester 2

Online Gantt, an online Gantt Chart making software was used to make the Gantt Charts. The Gantt Chart has been divided into sprints and under each sprint, the tasks associated with the sprint are mentioned. All tasks have been allocated a start and end date along with the duration in days.

A.3 Risk Management

This section highlights the potential risks and finds appropriate strategies to counter them. Risk Management is the process of identifying, analyzing, and proposing possible mitigation solutions to potential project risks that may occur during the project that might hinder its progress. Project risks refer to anything that does not go as planned.

Risk Management comprises of 4 main steps [9]:

1. **Risk Identification** This step involves identifying the potential risks associated with the project.

2. **Risk Analysis** This step involves assessing the likelihood and consequences of the risks. Likelihood and the Impact of the risk on the project take the following values:
   - Likelihood: High, Medium, Low
   - Impact: Catastrophic, Serious, Tolerable
3. **Risk Planning** This step involves devising strategies or plans to avoid or minimize the effects of the risk. These plans may be either preventive actions or mitigation actions.
4. **Risk Monitoring** This step involves monitoring the identified risks based on their likelihood of occurring.

Table A.1 lists the potential risks and finds appropriate strategies to counter them.

Sno.	**Risk**	**Risk Description**	**Strategy**
1.	**Training Time** Type: Time Likelihood: High Impact: Serious	The training time of the model is very long. The training time of the model exceeds the average training time by a large margin.	Use libraries and frameworks that can accelerate the training of the model. Use GPU or TPU to run the code instead of CPU, either by running the code on another hardware that uses these processors or using the GPU or TPU runtime on Google Collaboratory to run the code.
2.	**Loss of Data** Type: Tools Likelihood: Low Impact: Catastrophic	Problems were faced while saving data and the changes made weren't saved.	Making regular and progressive commits to GitHub after making changes to code and documenting changes made.

Continued on next page

Table A.1 – continued from previous page

Sno.	Risk	Risk Description	Strategy
3.	**Lack of Data** Type: Requirements Likelihood: Low Impact: Serious	The dataset chosen had insufficient records to train the model on.	Choose another dataset. Increase the training data using techniques like data augmentation.
4.	**Inability to Implement the Model** Type: Requirements Likelihood: Medium Impact: Catastrophic	Difficulties faced while implementing the model due to shortage of time or resources.	Research on previous implementations of the model and in an extreme case speak with the project supervisor about difficulties faced during model implementation and resolving the underlying issues.
5.	**Difficulties faced during Data Preprocessing** Type: Requirements Likelihood: Medium Impact: Catastrophic	The data preprocessing techniques and feature enhancement techniques may have led to lower accuracies.	Research and apply other preprocessing and feature enhancement techniques used in previous works in the same domain of research.

Continued on next page

Table A.1 – continued from previous page

Sno.	Risk	Risk Description	Strategy
6.	**Problems faced while using Project Tools** Type: Tools Likelihood: Low Impact: Serious	Resource utilization until exhaustion of capacity or insufficient resources leading to errors.	Research on resources compatible with the project. Use resources that have been previously used in related works.
7.	**Trailing behind the scheduled Project Plan** Type: Time Likelihood: Medium Impact: Serious	More than the predicted time is taken to complete a particular task or requirement.	Revise the project plan and stick to the time dedicated to each task.
8.	**Personal Issues (Health or Family Emergency)** Type: Requirements Likelihood: Medium Impact: Serious	Health issues or unpredicted events leading to a delay in completing the tasks.	Reallocate deadlines for the remaining tasks and complete them according to the priorities assigned. Discuss with the project supervisor about a mitigating circumstance and the revised project plan.

Table A.1: Risk Analysis

Appendix B

Profession, Legal, Ethical and Social Issues (PLES)

Professional and Legal Issues

The research papers used for the literature review have been referenced appropriately. The project would adhere to the relevant regulations like GDPR. The research papers and datasets used in this project are open source or have been granted permission to be used by the creators. Practices such as version control, tracking and managing progressive changes made to the code, creating a project plan, and project journals are also adopted.

Ethical and Social Issues

This project does not involve any use or collection of personal/private data. The dataset in this project is open-sourced and publicly available and is popularly used in research works related to American Sign Language Recognition.

Appendix C

Project Journal

A Project Journal is a documentation artifact that serves as a record of the progress of the project. It summarizes the meetings with the project supervisor, the work carried out since the last journal entry and the work to be done in the following 2 to 3 weeks. It gives a good reflection of whether the project is going as planned. Figures C.1, C.2, and C.3 are the Journals submitted in Week 5, Week 7, and Week 9 of Semester 1, respectively. Figures C.4, C.5, and C.6 are the Journals submitted in Week 5, Week 8, and Week 10 of Semester 2, respectively.

Journal Submission 1 (Week 5)

Student Name: Gauri Revankar

Student email address: dr2007@hw.ac.uk

Date of Submission: 13/10/2023

Title of dissertation and brief description	**Sign Language Recognition and Translation using Transformer-based Neural Network** Developing a convolution and transformer-based hybrid model to carry out Continuous Sign language recognition from videos and translation.
Communicating with supervisor	**Week 2 (18/09/2023)** Had a discussion on the following topics: - Components of a dissertation - Figuring out a topic for the dissertation - Going through research papers and datasets **Week 4 (02/10/2023)** - Going through more research papers. - Reviewing summarized lit reviews - Coming up with aim, objective, motivation, and datasets
References consulted	- IEEE - MDPI - Open Access - Springer Link - Google Scholar
Tools explored/used	- GitHub - Jupyter Notebook
Other work carried out	- Reading research papers. - Doing a background study on the topic, familiarizing with terms specific to the topic. - Coming up with aim, objective, motivation, and datasets.
Plan the next 2 to 3 weeks	- Structuring the Introduction section of D1 Draft. - Structuring the Related works and Lit Review section of the D1 Draft. - Researching on source code of models and implementation. - Exploring how work with the datasets found. - Get familiar with LaTeX.
Overall Reflection	So far, the project is on track, but will need to speed up the work. Challenges faced: - Choosing a topic for dissertation (I switched through several topics and went through past dissertation topics and proposals in my field of interest) - Reducing the problem space (I went through a lot of research papers and cent through their problem spaces) - Finding reliable research papers (I browsed through the research papers published on reliable sources such as IEEE, Springer Link etc.)
Additional section	

FIGURE C.1: Journal Entry 1

Journal Submission 2 (Week 7)

Student Name: Gauri Revankar

Student email address: dr2007@hw.ac.uk

Date of Submission: 27/10/2023

Title of dissertation and brief description	**American Sign Language Recognition using Transformer-based Neural Network** Developing a Transformer-based hybrid model to carry out Continuous American Sign Language Recognition from videos.
Communicating with supervisor	**Week 6 (16/10/2023)** Had a discussion on the following topics: - Reviewed Aim, Objectives, and Datasets of the project. - Reviewed research papers read so far and their proposals. - Given a task to create a draft version of Section 1, 2 and 3 of Deliverable 1.
References consulted	- IEEE - MDPI - Open Access - Springer Link - Google Scholar
Tools explored/used	- GitHub - ChatGPT - LaTeX
Other work carried out	- Read shortlisted research papers in detail. - Formulated aim and objectives of the project. - Did a background study on the topic, its history and evolution. - Created a draft version of Section 1, 2 and 3 of Deliverable 1.
Plan the next 2 to 3 weeks	- Creating a Gantt Chart. - Performing Requirement analysis. - Performing Risk Management. - Exploring how to work with the dataset found. - Getting familiar with LaTeX.
Overall Reflection	So far, the project is on track, but will need to speed up the work. Challenges faced: - Writing Literature reviews, summarizing the research works of other authors in the same field. - Finding reliable sources for research. - Working with LeTeX.
Additional section	

FIGURE C.2: Journal Entry 2

Journal Submission 3 (Week 9)

Student Name: Gauri Revankar

Student email address: dr2007@hw.ac.uk

Date of Submission: 10/11/2023

Title of dissertation and brief description	**American Sign Language Recognition using Deep Learning Networks** Evaluating different combinations of hybrid Deep Learning models to carry out Continuous American Sign Language Recognition from videos.
Communicating with supervisor	**Week 8 (31/10/2023)** Had a discussion on the following: - Reviewed the draft versions of Section 1,2 and 3 of Deliverable 1 and received feedback. - Given a task to formulate a Project plan.
References consulted	- IEEE - MDPI - Open Access - Springer Link - Google Scholar
Tools explored/used	- OverLeaf (LaTeX) - ChatGPT - Online Gantt Chart maker
Other work carried out	- Worked on the changes suggested during the meeting. - Started working on the LaTeX document for the D1 draft using Overleaf. - Created a draft version of the Project plan.
Plan the next 2 to 3 weeks	- Performing Requirement analysis. - Performing Risk Management. - Exploring how to work with the dataset found.
Overall Reflection	So far, the project is on track, but will need to speed up the work. Challenges faced: - Getting used to academic research paper writing. - Finding reliable sources for research. - Working with LeTeX.
Additional section	

FIGURE C.3: Journal Entry 3

Journal Submission 4 (Week 5)

Student Name: Gauri Revankar

Student email address: dr2007@hw.ac.uk

Date of Submission: 16/02/2024

Title of dissertation and brief description	**American Sign Language Recognition using Deep Learning Networks** Evaluating different combinations of hybrid Deep Learning models to carry out Continuous American Sign Language Recognition from videos.
Communicating with supervisor	**Week 2 (23/01/2024)** Had a discussion on the following: - Reviewed the report and was given feedback on the Deliverable 1 submission. Discussed the scope of improvement. **Week 4 (09/02/2024)** Had a discussion on the following: - Reviewed the progress as per the Gantt Chart. - Had a discussion on the code implementation of the project.
References consulted	- IEEE - MDPI - Open Access - Springer Link - Google Scholar
Tools explored/used	- Jupyter Notebook - VS Code - Google Collaboratory - ChatGPT - GitHub
Other work carried out	- Decided on subset of the total dataset to work on. - Performed frame extraction. - Performed video data augmentation. - Labelled and created a dataset out of videos.
Plan the next 2 to 3 weeks	- Write code for hybrid models. - Train and validate the models. - Fine tune the models. - Evaluate the performance of models.
Overall Reflection	So far, the project is on track, but will need to speed up the work. Challenges faced: - Choosing methods of frame extraction from videos. - Choosing methods of data augmentation.
Additional section	

FIGURE C.4: Journal Entry 4

Journal Submission 5 (Week 8)

Student Name: Gauri Revankar

Student email address: dr2007@hw.ac.uk

Date of Submission: 10/03/2024

Title of dissertation and brief description	**American Sign Language Recognition using Deep Learning Networks** Evaluating different combinations of hybrid Deep Learning models to carry out Continuous American Sign Language Recognition from videos.
Communicating with supervisor	**Week 6 (19/02/2024)** Had a discussion on the following: - Code check on the code implemented till Frame extraction and Data Augmentation. - Discussed challenges faced during model implementation. **Week 8 (08/03/2024)** Had a discussion on the following: - Reviewed the progress as per the Gantt Chart. - A quick review and feedback on the model implementation.
References consulted	- IEEE - MDPI - Open Access - Springer Link - Google Scholar
Tools explored/used	- Jupyter Notebook - VS Code - Google Collaboratory - ChatGPT - GitHub
Other work carried out	- Researched on how to create hybrid models. - Wrote code for ResNet50, BiLSTM, and the other models. - Working on fine tuning models.
Plan the next 2 to 3 weeks	- Fine tune the models. - Evaluate the performance of models. - Document progress.
Overall Reflection	So far, the project is on track, but will need to speed up the work. Challenges faced: - Building model architecture. - Fine tuning models.
Additional section	

FIGURE C.5: Journal Entry 5

Journal Submission 6 (Week 10)

Student Name: Gauri Revankar

Student email address: dr2007@hw.ac.uk

Date of Submission: 23/03/2024

Title of dissertation and brief description	**American Sign Language Recognition using Deep Learning Networks** Evaluating different combinations of hybrid Deep Learning models to carry out Continuous American Sign Language Recognition from videos.
Communicating with supervisor	Meeting on 22/03/2024: - Revied model architecture and evaluation results. - Asked to submit a draft of Deliverable 2. - Given feedback on work done so far.
References consulted	- IEEE - MDPI - Open Access - Springer Link - Google Scholar
Tools explored/used	- Jupyter Notebook - VS Code - ChatGPT - GitHub
Other work carried out	- Trained the models created. - Fine-tuned models. - Evaluated models and generated evaluation graphs.
Plan the next 2 to 3 weeks	- Documentation of the results. - Creating a final draft of Deliverable 2. - Possibly training the model on 100 gloss subset of dataset.
Overall Reflection	So far, the project is on track, but will need to speed up the work. Challenges faced: - Building model architecture. - Fine tuning models.
Additional section	

FIGURE C.6: Journal Entry 6

Bibliography

[1] Abey Abraham and V Rohini. Real time conversion of sign language to speech and prediction of gestures using artificial neural network. *Procedia Computer Science*, 143:587–594, 2018.

[2] Aishwarya. Introduction to recurrent neural network. `https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/`, October 2018.

[3] Dosovitskiy Alexey, Beyer Lucas, Kolesnikov Alexander, Weissenborn Dirk, Zhai Xiaohua, Unterthiner Thomas, Dehghani Mostafa, Minderer Matthias, Heigold Georg, Gelly Sylvain, Uszkoreit Jakob, and Houlsby Neil. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv*, October 2020.

[4] Laith Alzubaidi, Mohammed A. Fadhel, Omran Al-Shamma, Jinglan Zhang, J. Santamaría, Ye Duan, and Sameer R. Oleiwi. Towards a better understanding of transfer learning for medical imaging: A case study. *Applied Sciences*, 10(13), 2020.

[5] Tunga Anirudh, Nuthalapati Sai Vidyaranya, and Wachs Juan. Pose-based sign language recognition using GCN and BERT. *arXiv*, December 2020.

[6] Natarajan B., Rajalakshmi E., Elakkiya R., Kotecha Ketan, Abraham Ajith, Gabralla Lubna Abdelkareim, and Subramaniyaswamy V. Development of an end-to-end

deep learning framework for sign language recognition, translation, and video generation. *IEEE Access*, 10:104358–104374, 2022.

[7] Mayank Banoula. Introduction to long short-term memory(LSTM). `https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/lstm`, April 2023.

[8] Risang Baskoro. WLASL (world level american sign language) video, September 2021.

[9] M. BELGODERE. The risk management process: 4 essential steps. `https://www.migso-pcubed.com/blog/risk-management/four-step-risk-management-process/.`, September 2021.

[10] Ghanem Bilal, Rosso Paolo, and Rangel Francisco. An emotional analysis of false information in social media and news articles. *ACM Trans. Internet Technol.*, 20(2), apr 2020.

[11] Necati Cihan Camgoz, Oscar Koller, Simon Hadfield, and Richard 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 (CVPR)*, June 2020.

[12] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1800–1807, Los Alamitos, CA, USA, jul 2017. IEEE Computer Society.

[13] François Chollet. Review: Xception with Depthwise Separable Convolution — Better than Inception-v3 ImageNet Transfer Learning with Python. *Towards Data Science*, 2017.

[14] N. Coen. A brief history of american sign language. `https://blog.cyracom.com/ciiblog/history-of-american-sign-language-0.`, 2022.

[15] Christian Cuxac. French sign language: proposition of a structural explanation by iconicity. In *International Gesture Workshop*, pages 165–184. Springer, 1999.

[16] Bragg Danielle, Koller Oscar, Bellard Mary, Berke Larwan, Boudrealt Patrick, Braffort Annelies, Caselli Naomi, Huenerfauth Matt, Kacorri Hernisa, Verhoef Tessa, Vogler Christian, and Morris Meredith Ringel. Sign language recognition, generation, and translation: An interdisciplinary perspective. *arXiv*, August 2019.

[17] Data preprocessing in machine learning. `https://www.javatpoint.com/data-preprocessing-machine-learning`.

[18] Long Short-Term memory networks (LSTM)- simply explained! `https://databasecamp.de/en/ml/lstms`.

[19] Kothadiya Deep, Bhatt Chintan, Sapariya Krenil, Patel Kevin, Gil-González Ana-Belén, and Corchado Juan M. Deepsign: Sign language detection and recognition using deep learning. *Electronics*, 11(11), 2022.

[20] Dongxu. WLASL: WACV 2020 "word-level deep sign language recognition from video: A new large-scale dataset and methods comparison".

[21] Li Dongxu, Opazo Cristian Rodriguez, Yu Xin, and Li Hongdong. Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1448–1458, 2020.

[22] Mael Fabien. Xception in Keras. `https://maelfabien.github.io/deeplearning/xception/#ii-in-keras`.

[23] Keith D Foote. A brief history of deep learning. `https://www.dataversity.net/brief-history-deep-learning/`, February 2022.

[24] Samaan Gerges H., Wadie Abanoub R., Attia Abanoub K., Asaad Abanoub M., Kamel Andrew E., Slim Salwa O., Abdallah Mohamed S., and Cho Young-Im. Mediapipe’s landmarks with rnn for dynamic sign language recognition. *Electronics*, 11(19), 2022.

[25] Convolutional neural networks (CNN) with deep learning. `https://www.happiestminds.com/insights/convolutional-neural-networks-cnns/`, April 2019.

[26] Cooper Helen, Ong Eng-Jon, Pugeault Nicolas, and Bowden Richard. Sign language recognition using sub-units. In *Gesture Recognition*, pages 89–118. Springer International Publishing, Cham, 2017.

[27] A. Hosna, E. Merry, and J. et al. Gyalmo. Transfer learning: a friendly introduction. *Journal of Big Data*, 9:102, 2022.

[28] Image enhancement. `https://www.mathworks.com/discovery/image-enhancement.html`.

[29] Dharani J. All about ml — part 4: Evaluation metrics in classification algorithms. `https://medium.com/all-about-ml/evaluation-metrics-in-classification-algorithms-79c036a131cb`, March 2020.

[30] Michelle Jay. History of sign language - deaf history. `https://www.startasl.com/history-of-sign-language/.`, July 2008.

[31] Lu Jia, Nguyen Minh, and Yan Wei Qi. Deep learning methods for human behavior recognition. In *2020 35th International Conference on Image and Vision Computing New Zealand (IVCNZ)*, pages 1–6, 2020.

[32] Shin Jungpil, Musa Miah Abu Saleh, Hasan Md. Al Mehedi, Hirooka Koki, Suzuki Kota, Lee Hyoun-Sup, and Jang Si-Woong. Korean sign language recognition using transformer-based deep neural network. *Applied Sciences*, 13(5), 2023.

[33] V. Kanade. What is machine learning? understanding types & applications. `https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-ml/`, 2022.

[34] Cormier Kearsy, Fox Neil, Woll Bencie, Zisserman Andrew, and Necati Cihan Camgoz Richard. ExTOL: Automatic recognition of british sign language using the BSL corpus. In *Proceedings of the Sign Language Translation and Avatar Technology (SLTAT)*. Universitat Hamburg, 2019.

[35] Karolina Kozik. Without sign language, deaf people are not equal. *Human Rights Watch*, 2020.

[36] Vihar Kurama. Introduction to capsule networks. `https://blog.paperspace.com/capsule-networks/`, July 2020.

[37] Alzubaidi Laith, Zhang Jinglanand Humaidi Amjadand Al-Dujaili Ayad, Duan Ye, Al-Shamma, Omran Santamaría, J., Fadhel Mohammed, Al-Amidie Muthana, and Farhan Laith. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. `https://www.researchgate.net/figure/An-example-of-CNN-architecture-for-image-classification_fig7_350527503`, 2021.

[38] Nicole Laskowski. Recurrent neural networks. `https://www.techtarget.com/searchenterpriseai/definition/recurrent-neural-networks`, July 2021.

[39] Xinci Liu and Chang Zhao. Research on image feature extraction algorithm of the egg and egg white protein thermal gelation based on PCA/ICA. *Comput. Intell. Neurosci.*, 2022:1266332, March 2022.

[40] Jia Lu, Nguyen Minh, and Yan Wei Qi. Sign language recognition from digital videos using deep learning methods. In Minh Nguyen, Wei Qi Yan, and Harvey Ho, editors, *Geometry and Vision*, pages 108–118, Cham, 2021. Springer International Publishing.

[41] Madhiarasan, Dr M, Roy, and Prof Partha Pratim. A comprehensive review of sign language recognition: Different types, modalities, and datasets. *arXiv*, April 2022.

[42] Bernard Marr. A short history of deep learning – everyone should read. `https://www.forbes.com/sites/bernardmarr/2016/03/22/a-short-history-of-deep-learning-everyone-should-read/?sh=3fc78c9d5561`, March 2016.

[43] Mediapipe holistic — simultaneous face, hand and pose prediction, on device. `https://blog.research.google/2020/12/mediapipe-holistic-simultaneous-face.html?m=1`, 2020.

[44] Hossin Mohammad and M.N Sulaiman. A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining & Knowledge Management Process*, 5:01–11, 03 2015.

[45] H. Mujtaba. Introduction to ResNet or residual network. `https://www.mygreatlearning.com/blog/resnet/`, September 2020.

[46] Sign language. `https://education.nationalgeographic.org/resource/sign-language/`.

[47] Fikri Nugraha and Esmeralda C. Djamal. Video recognition of american sign language using two-stream convolution neural networks. In *2019 International Conference on Electrical Engineering and Informatics (ICEEI)*, pages 400–405, 2019.

[48] Prem Oommen. Resnets — residual blocks & deep residual learning. `https://towardsdatascience.com/resnets-residual-blocks-deep-residual-learning-a231a0ee73d2`, November 2020.

[49] Papers with code - CapsNet explained. `https://paperswithcode.com/method/fixcaps`.

[50] Early stopping. `https://paperswithcode.com/method/early-stopping`.

[51] Selective kernel. `https://paperswithcode.com/method/selective-kernel`.

[52] Introduction to convolutional neural networks architecture. `https://www.projectpro.io/article/introduction-to-convolutional-neural-networks-algorithm-architecture/560`.

[53] Sabour Sara, Frosst Nicholas, and Hinton Geoffrey E. Dynamic routing between capsules. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[54] Alyami Sarah, Luqman Hamzah, and Hammoudeh Mohammad. Isolated arabic sign language recognition using a transformer-based model and landmark keypoints. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, feb 2023. Just Accepted.

[55] Noha Sarhan and Simone Frintrop. Unraveling a decade: A comprehensive survey on isolated sign language recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3210–3219, 2023.

[56] Sign language recognition - an overview — sciencedirect topics. https://www.sciencedirect.com/topics/computer-science/sign-language-recognition.

[57] Tamura Shinichi and Kawasaki Shingo. Recognition of sign language motion images. *Pattern Recognition*, 21(4):343–353, January 1988.

[58] What is computer vision: Applications, benefits and how to learn it. https://www.simplilearn.com/computer-vision-article, April 2021.

[59] Rautaray S.S. and Agrawal A. Vision based hand gesture recognition for human computer interaction: a survey. *SpringerLink*, 2012.

[60] Rachel Sutton-Spence and Bencie Woll. *The linguistics of British Sign Language: an introduction*. Cambridge University Press, 1999.

[61] Starner T, Weaver J, and Pentland A. Real-time american sign language recognition using desk and wearable computer based video. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(12):1371–1375, 1998.

[62] Keras Team. Keras documentation: Keras Applications — keras.io. https://keras.io/api/applications/.

[63] William Tomkins. *Indian sign language*, volume 92. Courier Corporation, 1969.

[64] The history of sign language. https://www.stillmantranslations.com/the-history-of-sign-language/, July 2020.

[65] What are capsule networks and how are they related to neural networks? https://www.turing.com/kb/what-are-capsule-networks-and-their-relation-to-neural-networks, May 2022.

[66] ur Rehman Atiq, Belhaouari Samir Brahim, Kabir Md Alamgir, and Khan Adnan. On the use of deep learning for video classification. *Applied Sciences*, 13(3), 2023.

[67] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[68] Yugesh Verma. Complete guide to bidirectional LSTM (with python codes). `https://analyticsindiamag.com/complete-guide-to-bidirectional-lstm-with-python-codes/`, July 2021. Accessed: 2023-11-25.

[69] Deafness and hearing loss. `https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss.`, 2018.

[70] Chung Wan-Young, Haokai Xu, and Boon Giin Lee. Chinese sign language recognition with batch sampling ResNet-Bi-LSTM. *SN Computer Science*, 3(5):414, August 2022.

[71] Residual neural network. `https://en.wikipedia.org/w/index.php?title=Residual_neural_network&oldid=1164824154`, July 2023.

[72] Luke T. Woods and Zeeshan A. Rana. Modelling sign language with encoder-only transformers and human pose estimation keypoint data. *Mathematics*, 11(9), 2023.

[73] Dong Yu-nan and Liang Guang-sheng. Research and discussion on image recognition and classification algorithm based on deep learning. In *2019 International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)*, pages 274–278, 2019.

[74] Zhou Yuanding, Li Baopu, Wang Zhihui, and Li Haojie. Integrating temporal and spatial attention for video action recognition. *Security and Communication Networks*, 2022:5094801, April 2022.

[75] Jiang Yufeng, Li Fengheng, Li Zongxi, Liu Ziwei, and Wang Zijian. Enhancing continuous sign language recognition with self-attention and mediapipe holistic. In *2023 8th International Conference on Instrumentation, Control, and Automation (ICA)*, pages 97–102, 2023.

[76] E. Zvornicanin. Differences between bidirectional and unidirectional lstm. `https://www.baeldung.com/cs/bidirectional-vs-unidirectional-lstm`, 2022.