Vision-Based Automatic Translation for South African Sign Language (SASL)

Research on the automatic translation from SASL to English

Prepared by:

Mokgadi Setshekgamollo

Student Number: STSMOK001

Supervised by:

Robyn Verrinder

DEPARTMENT OF ELECTRICAL ENGINEERING

UNIVERSITY OF CAPE TOWN

Assoc Prof. Mohohlo S. Tšoeu

DEPARTMENT OF ELECTRONIC AND COMPUTER ENGINEERING,

DURBAN UNIVERSITY OF TECHNOLOGY

Submitted to the Department of Electrical Engineering at the University of Cape Town in fulfilment of the academic requirements for a Master of Science degree in Electrical Engineering.

Declaration

I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.

I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report/project from the work(s) of other people has been attributed, and has been cited and referenced.

This report/project is my own work.

I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as his or her own work.

Signature: Signed by candidate

Mokgadi Setshekgamollo, STSMOK001

December 01, 2023

Acknowledgements

I would like to acknowledge Prof. Mohohlo Samuel Tšoeu and Ms. Robyn Verrinder for their kindness, patience, support, and guidance as my co-supervisors throughout this research.

I would also like to thank all my friends and family for all the different ways that they supported me and showed their love during this journey.

Abstract

There are more than four million South Africans who are deaf and hard of hearing (DHH). However, most people with hearing abilities neither understand sign language nor know how to sign. This creates a communication barrier between the deaf and hard of hearing and people with hearing, to the disadvantage of the DHH. In 2018, South African Sign Language (SASL) became an official subject in South African schools and in 2023 it became South Africa's 12th official language. However, these implementations do not impose it on institutions and service providers. Although some provisions are made to cater to the needs of DHH people in the form of sign language interpreters, such interpreters are not always readily available. Sign language interpreters are also costly, charging in excess of R500.00 per hour.

In this research, we developed the first vision-based Neural Sign Language Translation model for SASL as a first step towards bridging the communication gap. To this end, we recorded a sizeable parallel SASL and English corpus with the help of six sign language interpreters, three of which are native signers and constitute around 90% of the dataset. The dataset comprises 5047 sentences in the domain of government and politics recorded in a studio setting with a uniform green background. At an average of 3.83 seconds per segment, this equates to around five hours of sign language data.

We conducted comprehensive experiments using various visual feature extraction architectures as well as translation architectures. We found that recurrent translation models outperform transformer models. We also investigated the impact of pretraining our feature extractor on a Continuous Sign Language Recognition task and fine-tuning on the SASL dataset and found that this is effective for improving feature extraction. Our best models achieved a BLEU-4 score of 1.35 on the SASL test set, a comparable

performance to the How2Sign dataset with a best BLEU-4 score of 1.73 but much lower than on the RWTH-PHOENIX-Weather 2014T dataset which produced a BLEU-4 score of 13.23 without gloss supervision. Our experiments also showed that annotating fingerspelled words as individual letters improves the performance of the model. Our model might benefit from the collection of more data and the addition of gloss annotation.

Our results on the SASL dataset are very poor and still very far from practical, indicating that more resources and experiments are required before we can remove the language barrier between the hearing and the Deaf. This would be most effectively achieved by working in collaboration with the Deaf Community to produce high quality datasets, annotations, and models.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

This chapter gives a brief background, and some literature and related works on Sign Language Recognition and Machine Translation, followed by a problem statement, research aim, questions, and objectives. The significance of the research, as well as the scope and limitations of the research, are then given followed by an outline of the report.

1.1 Background to the Study

According to the South African National Deaf Association (SANDA) website, more than 4 million South Africans were deaf and hard of hearing (DHH) in 2018 [1]. Most people with hearing abilities neither understand nor know how to use sign language for communication. This creates a communication barrier between the Deaf and people with hearing, to the disadvantage of the Deaf. This is especially true in South African public service facilities where provisions are largely not made for the Deaf. For instance, transport interchange facilities such as airports and train stations do not have announcements in sign language. The recognition of South African Sign Language (SASL) as an examinable subject in schools and as South Africa's 12th official language does not impose it on institutions and service providers. Despite its many advantages, such as adaptability, quality of translation, and effectiveness, human translation is expensive and not always readily available. For instance, in February 2018, when President Jacob Zuma gave his resignation speech, there was no translator or auto-generated subtitles [2].

1.1.1 Machine Translation

Machine Translation (MT) is the automatic translation from one language to another and is a field of study in Natural Language Processing (NLP). Natural Language Processing is a field in artificial intelligence that aims to model and process natural languages. Work on MT began in the late 1940s with the main approach then being dictionary-based word-for-word processing [3]. With advancements in machine learning and deep learning in particular, it has since grown into a major research area with multilingual translations and performance comparable to that of human translators in text-based spoken language Machine Translation applications. Other fields in NLP include language modelling, text summarisation, automatic speech recognition (ASR), text-to-speech synthesis (TTS), sign generation, text-to-sign synthesis, sentiment analysis, speaker identification, language identification and the recent large language model (LLM) automatic response and reasoning.

Some of the advantages of Machine Translation systems are that they are accessible, fast, and user-friendly. Although research into Sign Language Translation has increased over the years, DHH people are still being excluded as more research is being done on spoken language translation. In fact, Sign Language Translation datasets are far behind their spoken language counterparts in size and domain. One of the largest spoken language Machine Translation datasets comprises 4.5 billion parallel sentences in 576 language pairs [4]. In comparison, one of the larger-scale Sign Language Translation datasets only consists of around 36 thousand parallel sentences in one language pair [5].

1.1.2 Sign Language Recognition and Translation

Work on the translation of signed languages to spoken languages can be categorised into two tasks, Sign Language Recognition (SLR) and Sign Language Translation

(SLT). The difference between SLR and SLT lies in the approach to the problem. Sign Language Recognition approaches the problem similar to a gesture recognition problem where each sign is mapped to some word or words (referred to as glosses). Sign Language Translation on the other hand approaches the problem as a Machine Translation problem which recognizes the differences in grammar and other linguistic aspects between spoken language and the sign language. Most of the research done on sign languages in the field of artificial intelligence has been on SLR as an intermediary step towards SLT. The Neural Sign Language Translation (NSLT) task was only formalized in 2018 by Camgoz *et al.* [6] when they released the first publicly available NSLT dataset and proposed an NSLT model pipeline that is analogous to spoken language Neural Machine Translation and used it to translate German Sign Language into German.

1.1.3 Challenges in Sign Language Recognition and Translation

Signs can be divided into two categories, static signs and dynamic signs. In addition to non-manual features, static signs only consist of handshape, location, and orientation whereas dynamic signs also include a movement component. The movement poses a challenge to Sign Language Recognition and Translation because signers have varying speeds of movement in their signing, and this can affect predictability. Additionally, capturing the transition from one sign to the next is also a challenge due to the varying durations of signs and coarticulation (the beginning of the next sign is affected by the ending of the previous one). This is further complicated by hand occlusions during signing and the use of sign space for referent localization and referents that are later referenced without full articulation.

Another challenge is that few readily available datasets exist for sign language processing and the collection and annotation of continuous sign language data is time-

consuming and expensive. Sign language data collection is also physically strenuous for the participants, compared to text and speech which require no physical activity.

For vision-based recognition and translation, there is a further requirement for the system to be invariant to complex backgrounds, and differences in lighting and the physical appearance of the signer. The storage and processing of video data is also more computationally expensive than processing text and audio for spoken language processing.

1.1.4 Research on Sign Language Recognition and Translation

Research on SLR and SLT has been conducted mainly in two different ways, sensor-based, and vision-based. Sensor-based methods use gloves with sensors that extract the dynamic information of the signs. On the other hand, vision-based methods use data collected from digital images or video frames and computer vision techniques to extract useful features of the signs. Other research integrates the two methods by using cameras that have sensors that can provide skeletal and depth information (such as the Microsoft Kinect) to extract additional orientation and motion information.

Classical visual Sign Language Recognition methods typically require feature extraction from images and the classification of signs from the extracted features. Some common feature extraction techniques are Speeded Up Robust Features (SURF) [7], Principal Component Analysis (PCA), Histogram of Oriented Gradient (HoG) [8] and artificial neural networks (ANNs). For classification, static gestures use techniques such as Support Vector Machines (SVMs) [9], K-nearest neighbours (KNN) [10] and Artificial Neural Networks. Jin *et al.* [11] employed SURF for feature extraction and KMeans clustering [12] and a multi-class SVM for the classification of 16 static American Sign Language alphabets and achieved an overall accuracy of 97.13% using a Nokia Lumia 1520 smartphone with the Windows Phone 8.1 operating system.

Although they obtained high accuracy, their solution is not applicable to real-life situations because Deaf people do not communicate using fingerspelling and only use fingerspelling when referring to named entities for which they do not have signs.

For dynamic gestures, Dynamic Time Warping (DTW) [13] or Hidden Markov Models (HMMs) may be used. For instance, Jangyodsuk *et al.* [14] used HoG for handshape feature extraction and representation and DTW based on sign trajectories to classify signs in American Sign Language (ASL). They achieved a top-10 accuracy (the correct class is in the top 10 probabilities) of 86% on a dataset of 1113 isolated signs. This solution is also not applicable to real life as it recognizes individual signs instead of whole sentences.

Neural Machine Translation (NMT) has recently been proposed for Sign Language Translation. This method uses deep learning models for the task of end-to-end translation of sign language. Kumar *et al.* [15] proposed a multi-stage Sign Language Translation system that uses neural networks to translate American Sign Language into English. Stage 1 uses a Long Short Term Memory (LSTM) network for sign recognition while stage 2 uses a Recurrent Neural Network (RNN) encoder and LSTM decoder to translate the gloss into English. Stage 1 achieved a Gloss Error Rate (ratio of wrong glosses predicted to the total number of glosses predicted) of 23% and a Gloss Recognition Rate (ratio of correct glosses predicted to the total number of glosses in the target set) of 86%. Stage 2 achieved a BLEU [16] score of 49.7 when using a two-layer encoder-decoder with attention with a vocabulary size of 31 signs.

Camgoz *et al.* [6] formalized Neural Machine Translation for sign language (NSLT) in 2018 and developed an end-to-end sequence-to-sequence model for the translation of German sign language into German. Their model uses a 2D Convolutional Neural Network (CNN) for the spatial embedding of video frames and an RNN-based

sequence-to-sequence model with attention to jointly learn to align, recognize, and translate sign videos to German text. The model achieved a BLEU-4 score of 9.58 on the RWTH-PHOENIX-Weather 2014T dataset. The BLEU-4 score increased to 18.13 when intermediary gloss-level supervision was introduced.

Guo *et al.* [17] tackled the task of translating Chinese Sign Language with a hierarchical LSTM-based sequence-to-sequence model. They used a 3D-CNN to extract visual features and to capture spatiotemporal context in the input videos and used online key clip mining to compact less important clips to minimise their negative effects on the learning of the model. Their model, with attention, achieved a BLEU-4 score of 9.28 on the signer independence test with seen sentences and 0.605 for unseen sentences on a dataset with a vocabulary size of 179 signs.

In 2020, Camgoz *et al.* [18] proposed an architecture that jointly performs the recognition and translation of German Sign Language using Transformer networks and achieved a BLEU-4 score of 21.80 on the RWTH-PHOENIX-Weather 2014T. They showed that their new approach to adding gloss-level supervision outperforms the multistage architecture that is usually used in SLT models.

Although some progress has been made in the field of Sign Language Translation, the solutions are still far from practical. Most datasets that are used to train translation models are limited to specific domains and limited in vocabulary and are thus not applicable to real-life situations. Also, most models tend to be signer-dependent. Furthermore, models still require gloss-level supervision to perform decently which adds to the manual labour required when creating sign language datasets.

1.1.5 Research on SASL Recognition

Research conducted on SASL includes the development of a rule-based Machine Translation system for translating English to SASL by van Zijl [19], the development of a low-cost glove for the development of an SASL dataset by McInnes [20], the glove-based recognition of 31 SASL static signs by Seymour and Tšoeu [21] and the upper body pose recognition towards the vision-based translation of SASL by Achmed [22]. To the best of our knowledge, no work in Neural Sign Language Translation has been done on South African Sign Language.

1.2 Problem Statements

As most of South Africa's hearing population does not know sign languages, there exists a communication barrier between Deaf South Africans and the hearing population, to the disadvantage of the Deaf. Human sign language translators are few in South Africa and are not always available. On the other hand, the legal recognition of SASL as the 12th official language does not impose it on institutions and service providers. The few automatic translation systems developed for South African Sign Language are still far from practical and none of them have employed deep learning methods. The problem is thus a societal problem as well as an engineering problem.

1.2.1 Societal Problem

There is a need to bridge the communication barrier between the South African Deaf community and the hearing community.

1.2.2 Engineering Problem

There is a need for a system that can automatically capture the linguistic properties of South African Sign Language and use these properties to automatically translate it to English.

1.3 Research Aim

The aim of the research is the design, development and evaluation of vision-based automatic translation software for South African Sign Language (SASL). The system must be capable of capturing SASL and translating it into English.

1.4 Research Questions

The research questions are posed as follows:

- What are the considerations to make when designing and developing a parallel SASL and English corpus so that it can be suitable for the automatic translation task and be useful for further research into the field?
- Which types of computer vision and Neural Machine Translation (NMT) architectures are most effective for translating SASL to English in terms of translation accuracy?
- What are some of the adaptations that can be made to the Neural Sign Language Translation (NSLT) model and dataset annotation to improve the quality of the translation of SASL to English?

1.5 Research Objectives

Given the above research questions, the objectives of this research are therefore to:

- Design and develop a parallel SASL and English dataset suitable for use in Neural Sign Language Translation.
- Consider existing computer vision and NMT architectures and develop a suitable machine learning model for the translation of SASL to English.
- Evaluate the model and investigate ways to improve its performance.

1.6 Significance of the Research

Our research is the first exploration of the use of deep learning methods in the automatic translation of South African Sign Language to English. The dataset collected in this research, which is the first Neural Sign Language Translation parallel corpus for SASL, was based on an existing larger text-based Machine Translation dataset. This, along with the simple environment in which the dataset was recorded (uniform, green background), allows for ease in extending the dataset for further studies and future works. The exploration of the impact of annotating finger spelling, and the effectiveness of using a pre-trained autoencoder as a handshape feature extractor had not been done prior to this work to the best of our knowledge and will provide insight for future work on Sign Language Translation.

1.7 Scope and Limitations

The scope of the research includes the design and development of a SASL to English dataset, investigation into suitable deep learning methods for Sign Language Translation, design, and development of translation software as well as its evaluation and refinement. The research was meant to be a preliminary study and therefore the design and development of the dataset was limited in size and domain to that necessary to test proposed methods and was not exhaustive. We did not experiment with any data augmentation methods. This research only considered deep learning approaches to the Sign Language Translation task, classical methods are not included in the scope. Although sensor-based methods of Sign Language Translation may be explored in the literature, this research was vision-based and thus sensor-based approaches to Sign Language Translation are also not included in the scope. Also out of scope is translation in the opposite direction (i.e., translation from English to SASL).

1.8 Methodology

To meet the objectives of our research, we needed to design and develop a suitable parallel SASL and English dataset. When making considerations for our dataset, we looked at existing publicly available benchmark datasets. We collected a dataset comprising 5047 video segments and their corresponding English translations with the help of six sign language interpreters, three of which are native signers. The videos were recorded in a studio setting with a uniform green background.

We used the standard Neural Sign Language Translation model pipeline that comprises a visual feature extractor (2DCNN, 3DCNN, and pose) followed by a translation model. We experimented with three visual feature extraction architectures and three translation architectures (recurrent sequence-to-sequence models with LSTMs, recurrent sequence-to-sequence models with Gated Recurrent Units (GRUs), and transformer models), resulting in nine models to experiment with. We evaluated our models using the BLEU-4 score.

After the initial experiments, we experimented with fine-tuning our visual feature extractor on a Sign Language Recognition task and explored the use of autoencoders for feature extraction of the hands. We also evaluated the effect of annotating fingerspelled tokens on the performance of our models.

1.9 Report Outline

The rest of this report is arranged as follows:

Chapter 2: takes a brief look at sign languages and their grammar, with a focus on South African Sign Language.

Chapter 3: looks at deep learning and some of the building blocks of deep learning models. It then discusses some of the architectures used in Machine Translation. It goes on to list some of the metrics employed in evaluating Machine Translation models and gives some examples of the software used to develop these models.

Chapter 4: discusses the literature in the field of Sign Language Recognition and Translation. It explains the difference between Sign Language Recognition and Sign Language Translation and mentions some of the datasets available for these tasks. The focus of the chapter is deep learning as applied to the translation of signed languages.

Chapter 5: takes a deeper look at the benchmark datasets for NSLT. It looks at the key statistics of the datasets and the format of the data. It then goes on to detail the process followed in developing a parallel English and SASL dataset for this research.

Chapter 6: provides a detailed description of the proposed pre-processing methods, model architectures and pipeline.

Chapter 7: details how our experiments were conducted in the research and the different models that were investigated. It also presents the results of the experiments and discusses the various factors that might have influenced the performance of the models.

Chapter 8: presents the conclusions drawn from the research and gives recommendations for future work.

Chapter 2: Review of Sign Languages

Sign languages are visual-gestural languages which are produced by using the hands, face, and upper torso and are processed visually. They have their own specific linguistic rules and do not translate to spoken languages word by word. They occur naturally as a means of communication between members of the Deaf community. As a result of this, there is no universal sign language and many different sign languages have developed throughout the world, with differing dialects even within a single country. Some examples of sign languages are American Sign Language (ASL), British Sign Language (BSL), German Sign Language (DSGS) and South African Sign Language (SASL). This chapter looks at the way grammar functions in sign language and gives a brief history of South African Sign Language.

2.1 Gloss Annotation of Sign Languages

Although sign languages do not have a formal written form, they are sometimes represented using gloss annotation. Gloss annotation for sign languages is generally a written representation of every sign in the sentence and is generally written in uppercase. For example, the sign for "want" would be annotated as "WANT". Some gloss annotations also include non-manual features. In the annotation below that translates to "Did John buy a book?", the line above the sentence represents non-manual features for yes or no questions which are expressed by raising the eyebrows.

y/n
JOHN BUY BOOK

2.2 Grammar in Sign Languages

Signs are made up of five basic components. These components are handshape, hand movement, hand location, palm orientation and non-manual features such as facial expressions. Any combination of these components can occur at the same time to make up a sign and changing one of these components in a sign may change its meaning completely. For example, as shown in Figure 2-1, in SASL the signs WANT and HAPPY differ only in movement. The fact that these components can occur simultaneously means that multiple pieces of linguistic information may be conveyed at a time. Signs are also made up of a series of movements and holds, which shows a level of sequential organisation as well [23].

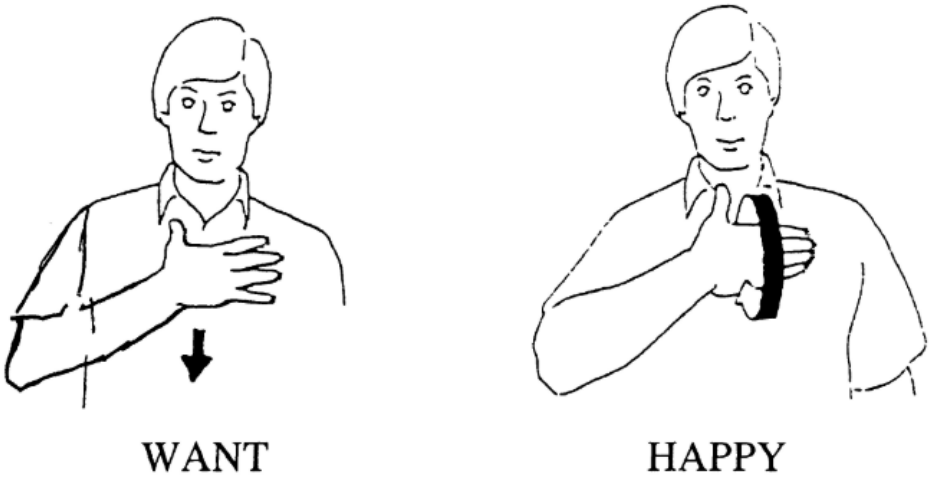


Figure 2-1: The SASL signs for WANT and HAPPY. The only difference between the two signs is in the movement. In WANT, the hand makes a downward brushing movement with the palm facing the body. In the case of HAPPY, the movement is circular; the other parameters remain the same [23].

Every sign language dialect has a finite set of handshapes. These can be common to other sign languages and dialects or unique to a particular sign language. Signs can be performed with one or both hands. Like spoken languages, sign language sentences comprise nouns, pronouns, verbs, adjectives, adverbs, and prepositions. Nouns are usually expressed lexically by a single sign. In SASL, proper nouns may be lexicalised

(e.g., sign names and places that feature frequently in the Deaf world such as SPRINGBOKS (the national rugby team), MANDELA, CAPETOWN, JOHANNESBURG), fingerspelled in full (for surnames) or with phonological deletion (e.g., P-M-B for Pietermaritzburg), or expressed using calques (e.g., Grahamstown = GREY-HAM-TOWN). Adverbs and adjectives are usually modulated by non-manual features [24].

Sign languages also make use of space to perform linguistic functions. For example, space is used for referent localization to perform pronominal functions. When the signer first mentions a proper noun, they can place it on a spot in the signing space (the space in front of and on the sides of the signer). Once they have done this, they can then point to this spot in subsequent sentences when referencing the proper noun [25]. This use of space for pronominal functions also presents a challenge in the automatic translation of signed languages as Machine Translation methods are typically trained at the sentence level and may not always have all of the context.

Similar to spoken languages, sign languages have their own syntactic rules that may be different to those of some spoken languages and even other sign languages. South African Sign Language has a different syntax from that of English. The examples below demonstrate these differences.

English: My name is [NAME]
SASL: NAME MINE [FINGER SPELLED/SIGN NAME]

English: What is your name?
wh-q
SASL: NAME YOURS WHAT

2.3 The History of South African Sign Language

The first school for the Deaf in South Africa was established by Irish Dominican nuns in Cape Town in 1863 and it was known as the Dominican Grimley Institute for the Deaf. This school enrolled all races and used sign language as a medium of instruction using Irish signs and the Irish one-handed alphabet. In 1877, the German Dominican nuns arrived in South Africa and brought with them German signs and the European two-handed alphabet [26], [27].

In 1880, at a conference in Milan attended by educators of the Deaf (who were not deaf themselves), it was decided that sign language should be abolished, and that the oral method should be the medium of instruction for the Deaf [23]. In response to this, The Worcester School for the Deaf and Blind was established in 1881 which combined oral and manual teaching methods. Other schools for the Deaf were later established that either combined manual and oral methods or strictly used oral methods. When the Nationalist Party came into power in 1948, more schools for the Deaf were established according to the spoken language of the ethnic groups, and in line with the Bantustan policy [27].

In the 1960s, the linguist William Stokoe published his study on American Sign Language (ASL) which showed that ASL is a language made up of phonological, morphological, and syntactic units just like spoken languages [28]. Following this, the philosophy of Total Communication was introduced, where sign language and spoken languages are used at the same time. This philosophy was adopted by some South African schools for the Deaf [29].

In 1996, South Africa officially recognized sign language as the medium of instruction for the Deaf [30]. In 2018, UMALUSI, the government authority responsible for monitoring standards for general and further education and training in South Africa,

officially recognized SASL as a home language in the education system which would be an examinable subject for the National Senior Certificate (NSC) [31]. In 2023, SASL was officially recognized as one of the South African official languages.

SASL is not the only active sign language in South Africa, but it is the one that is encouraged by the government and the one used in the interpretations of the news and parliament.

2.4 Linguistic Research on SASL

Penn and Reagan [32] conducted a study on the language usage of various groups of Deaf adults in South Africa. Below is a list of what they discovered about the lexical diversity of South African Sign Language.

- Of the 2500 lexical items collected for the dictionary of SASL, only 2% of all the words represented had a single, common sign across all the different deaf groups represented, and roughly 10% of the words had as few as one or two signed variants. On average, six variants per word were found and the range went as high as eleven variants, each informant having a different sign. It was observed, nonetheless, that inter-group communication among the different deaf groups was fluent and efficient. This presents a further challenge for SASL translation systems to be able to understand the different dialects and synonyms in the language.
- Due to the use of oral methods in much of South African deaf education, most deaf people employ an extensive amount of vocalization when they sign. This further contributes to variation in the sign language because vocalizations are based on different home languages (Afrikaans, English, Sesotho, IsiZulu, etc.). SASL translation systems must therefore be invariant to these variations and

the effect of oral communication in different languages on non-manual features.

- Although there is a high degree of lexical diversity in the South African deaf community, this diversity is not manifested in either the syntactic or morphological features of SASL observed.
- In those groups where there is a historical tradition of oral education, mouthing of words tends to take the place of, or perhaps mask, other non-manual linguistic behaviours.
- Although distinct sign variants have been observed and recorded, it should also be noted that often the variants are marked by minimal difference, e.g., range of motion, use of one or two hands, and minor handshape variation. These variants also often share broad common features like general formation.

The study was performed almost 30 years ago but the results can still be considered relevant as languages tend to evolve slowly.

Chapter 3: Deep Learning Methods

The field of Artificial Intelligence (AI) has been around since the 1900s with the development of computer science. It became dormant when machines were not advanced enough to process large information and large amounts of digital data were not readily accessible. As the 21st century dawned, machines became powerful enough to process large amounts of data efficiently and the internet grew, making the collection of copious amounts of digital data easier. In 2012, when AlexNet [33] significantly outperformed all the other models submitted to the ImageNet competition, AI emerged from its dormancy in the form of deep learning. Since then, deep learning has gained considerable popularity as it continued to yield better performance than feature engineering methods in many research areas.

Some of the common building blocks of deep learning architectures are linear layers, Convolutional Neural Networks, and recurrent networks. This chapter gives a brief overview of the inner workings of these building blocks. It also gives a history of Machine Translation and explores some of the deep learning architectures that have been developed in this field.

3.1 Neural Networks

Neural networks in deep learning take an input and the desired output and learn the relationships between them. They can be as simple as a single linear layer that performs an affine transformation followed by a non-linearity, or as complicated as multiple layers with chains of functions (e.g., convolutions, pooling etc.) that learn millions of parameters. They often require large amounts of data to learn these relationships so that they can generalize well on unseen data.

For an input vector x of shape $N \times D$, a linear layer performs an affine transformation

$$h = Wx + b, \tag{3-1}$$

where W is a matrix of learned weights of shape $D \times M$ and b is a bias or offset matrix of shape $N \times M$. Linear layers are typically followed by an activation function to introduce non-linearities such that the output

$$y = activation(Wx + b) = activation(h). \tag{3-2}$$

Activation functions help the model learn more complex representations. Figure 3-1 shows some of the common activation functions in deep learning applications. A neural network with several hidden layers is called a multi-layer perceptron (MLP).

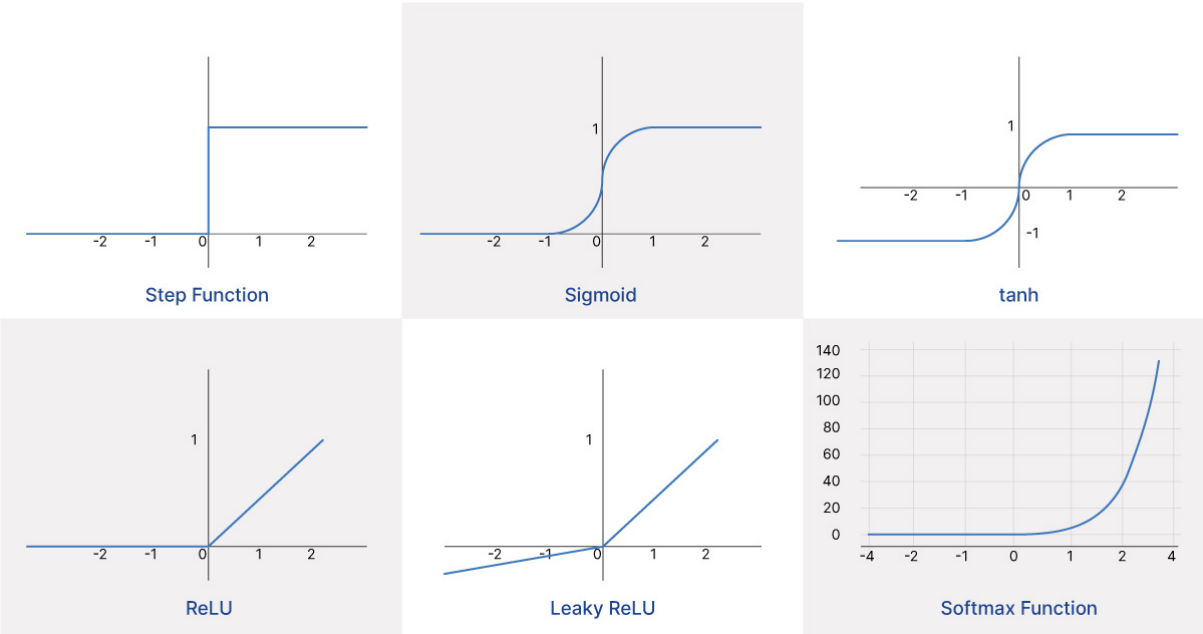


Figure 3-1: Graphical representations of some of the popular activation functions in deep learning [34].

3.2 Convolutional Neural Networks

Convolutional Neural Networks process arrays of data. They have become very popular in processing visual data such as images and video, although they can process data from other modalities as well. They gained their popularity in visual applications in 2012 when AlexNet significantly outperformed all other models proposed for the ImageNet challenge. CNNs are typically constructed by stacking several convolution blocks which are sometimes followed by a flatten layer, fully connected layers, and a softmax layer for classification. A convolution block is typically made up of a convolutional layer and an activation layer (typically ReLu) followed by a pooling layer.

A convolutional layer uses kernels to produce feature maps. The kernels are usually smaller in size than the input and use a sliding window approach throughout the input to produce the feature maps. Figure 3-2 shows an illustration of a typical CNN that would be used for image classification.

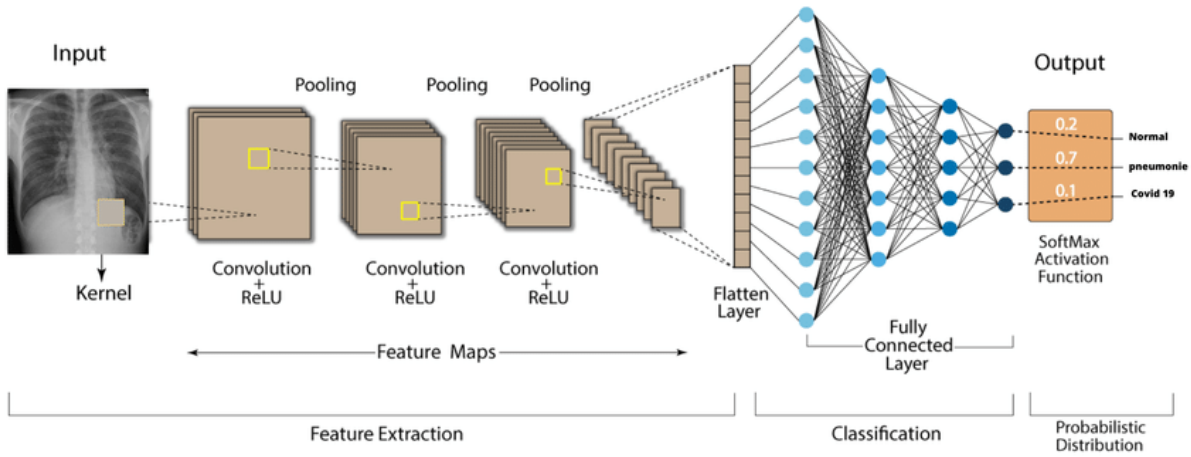


Figure 3-2: An illustration of a CNN used in image classification [35].

3.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) [36] model sequential data. Given a sequence of vectors as input, the RNN processes one timestep at a time computing the hidden state

vector h_t. The hidden state is initialized to a zero vector and then the hidden state of each timestep is computed based on the current input and the previous hidden state as

$$h_t = \tanh(W_{xh}x_t + W_{hh}h_{t-1} + b_h)\,,\ h_0 = 0. \tag{3-3}$$

In this way, the network always has access to information from previous timesteps and can therefore keep track of long-term dependencies such as gender or tense in an NLP task. The hidden state at the end of the sequence contains information about the whole sequence. For a classification task, this hidden state can be passed into a linear layer followed by a softmax layer to compute probabilities. The shortened version of the RNN equation is

$$h_t = RNN(x_t, h_{t-1}). \tag{3-4}$$

Since the weights (W) and biases (b) between timesteps are shared, the RNN can be thought of as a single block that processes its input and then feeds back its output as an input along with the input of the next timestep until the end of the sequence. Figure 3-3 shows a graphical representation of an RNN block in a single timestep.

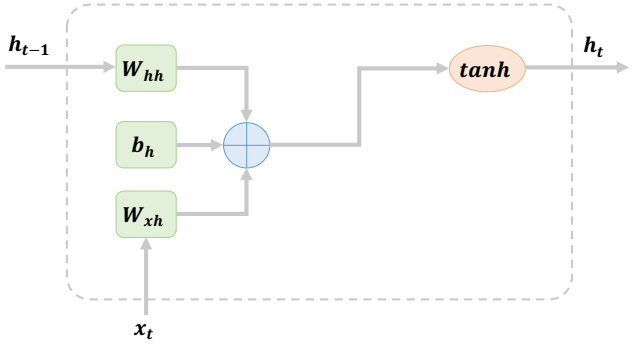


Figure 3-3: Graphical representation of an RNN block at a single timestep.

As much as standard RNNs can model sequential data well, they suffer from the diminishing gradient problem where gradients get smaller and smaller from the end of the sequence to the beginning. This leads to the gradients at the beginning of the

sequence being so small that they do not affect the updating of the parameters, effectively hindering the network from learning long-term dependencies. A solution to this was the introduction of the Long Short-Term Memory (LSTM) network [37], a variation of the RNN that ensures that the gradients at every timestep are always close to one. The LSTM network introduces a new vector called the context vector c_t and contains two internal gates (the input gate i_t and the output gate o_t) for better memory control within the network. Modern applications of the LSTM also include a forget gate (f_t) [38], which gives the network the option to discard information that is no longer relevant. A graphical representation of an LSTM block in a single timestep is shown in Figure 3-4.

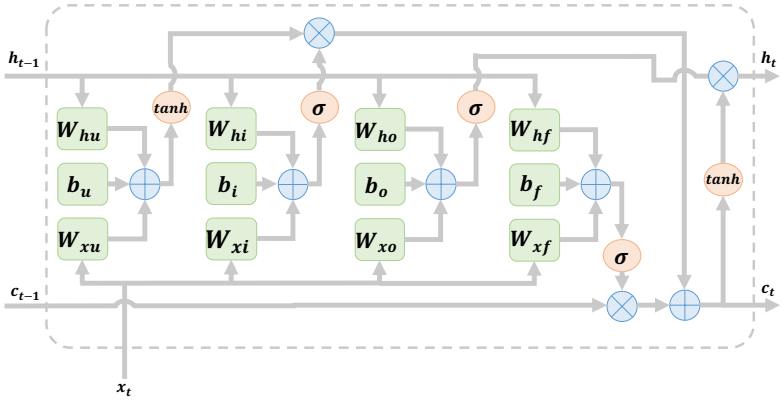


Figure 3-4: Graphical representation of an LSTM block at a single timestep.

The LSTM network is not the only variation that was designed to overcome the limitations of the standard RNN. One other variation is the Gated Recurrent Unit (GRU) [39] which also uses memory gates but is simpler than the LSTM in that it only uses two gates and does not introduce a context vector. In a GRU network, the reset gate r_t modulates the previous hidden state h_{t-1} before it is used to calculate the candidate hidden state similarly to the standard RNN. The current hidden state is then computed by taking a weighted sum of the previous hidden state and the candidate hidden state. The weighting is done by the update gate z_t. Figure 3-5 shows a graphical representation of the GRU block in a single timestep. The GRU network

requires fewer parameters compared to the LSTM but has been shown in [39] to have comparable performance and so becomes the better choice when memory footprint is a concern.

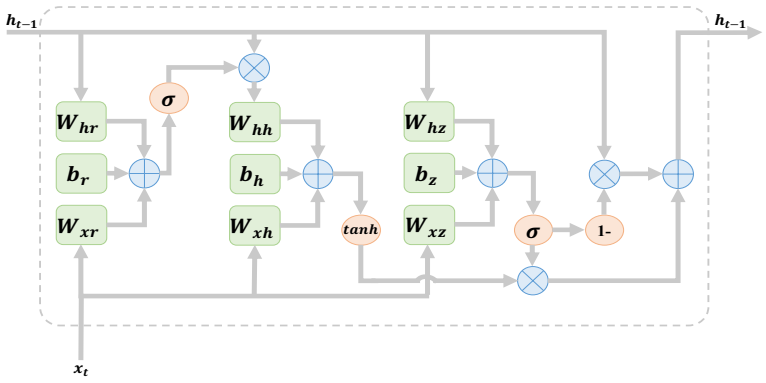


Figure 3-5: Graphical representation of a GRU block at a single timestep.

3.4 Training Neural Networks

When training a neural network, the objective is to find a set of parameters or weights such that the difference between the network output and the expected output is minimized. This is done by minimizing the objective function (also known as the loss function). The objective function calculates the difference (or the loss) between the expected output and the model's output. During training, the network does a forward pass on the input data and produces an output. The model's weights are then updated using backpropagation to decrease the loss. Backpropagation uses the chain rule to calculate the gradients of the loss function with respect to the model weights to find the direction of decreasing and updates them in this direction, iterating from the last layer to the first. Neural networks are trained on large amounts of data so that they can be able to generalize and perform well on data they have not seen.

During training, there are trainable parameters and hyperparameters. Trainable parameters are the model weights which get updated during backpropagation. Hyperparameters are not trainable and are chosen by the person training the models,

either from convention or experimentation. Examples of hyperparameters are the batch size, the learning rate and the number of training epochs or steps.

3.5 Sequence-to-Sequence Models

Sequence-to-sequence models are used to tackle tasks that map one sequence to another. For a source sequence $S = (s_1, s_2, \dots, s_n)$ with n tokens, the goal of the sequence-to-sequence model is to learn the conditional probability, $p(T|S)$, of generating a target sequence $T = (t_1, t_2, \dots, t_l)$ with l tokens. Tasks of this nature are common in Natural Language Processing (NLP). Examples of sequence-to-sequence tasks in NLP are automatic speech recognition, text-to-speech synthesis, text summarization, question answering, and Machine Translation. This research focuses on sequence-to-sequence models in the context of Machine Translation.

3.6 Machine Translation

Machine Translation (MT) is the automatic translation from one natural language to another. The language that is being translated is referred to as the source language and the language to which the source language is being translated is referred to as the target language. Initially, Machine Translation was dictionary-based based where sequences were translated word for word. This was not very ideal as different languages have different sentence structures and grammatical rules. There are two types of Machine Translation, rule-based Machine Translation and corpus-based Machine Translation.

3.6.1 Rule-Based Machine Translation (RBMT)

Rule-based Machine Translation requires linguistic knowledge of both the source and target languages. The grammar and other linguistic rules of the languages are used to analyse the source language and to generate the target language translation. One of

the challenges of rule-based translation is modelling the large number of complexities and exceptions within the rules of natural languages.

3.6.2 Corpus-Based Machine Translation

This form of Machine Translation relies on data rather than linguistic rules. Expert knowledge of the source and target languages is not required, only a parallel corpus of examples from both the source and the target language. Data from the source language is used in parallel with translations in target languages to generate a translation model. Two common ways of doing this are Statistical Machine Translation (SMT), which uses statistical models to learn translation from the source to the target language given aligned parallel corpora, and Neural Machine Translation (NMT), which uses deep learning to learn a statistical model for performing translations.

3.7 Statistical Machine Translation

Statistical Machine Translation (SMT) uses statistical models to learn how to translate between two languages given a parallel bilingual corpus. Implementations of SMT are generally phrase-based (PBMT) which means that they translate sequences of words or phrases. It was proposed by Brown *et al.* [40] in a task to translate between French and English using a corpus that is made up of sentence pairs from the Canadian parliament proceedings. In SMT, the goal is to find the sentence or phrase T in the target language that produces the highest probability given the sentence or phrase S in the source language. The probability associated with T can be calculated using Bayes' theorem as

$$Pr(T|S) = \frac{Pr(T)Pr(S|T)}{Pr(S)}. \tag{3-5}$$

Since the denominator of the equation is not dependent on T, $Pr(T|S)$ is maximized by maximizing $Pr(T) * Pr(S|T)$. $Pr(S|T)$ is referred to as the translation model and is

responsible for the translation of input phrases to the target phrase whereas $Pr(T)$, the language model, assesses whether the translation produced by the translation model is a suitable or natural phrase in the target language. This type of machine translation implies that for a given source phrase, there are several candidate translations or hypotheses and thus a need for some kind of searching algorithm (e.g., beam search) that will find the translation with the highest probability.

3.8 Neural Machine Translation

Neural Machine Translation (NMT) is the most recently proposed solution for machine translation. This type of approach makes use of deep learning methods to learn the statistical model for translating between the two languages. Some common models used for NMT are RNN-based sequence-to-sequence models [38], [41], [42], convolutional sequence-to-sequence models [43] and Transformers [44].

Sequence-to-sequence models in deep learning typically consist of an encoder network and a decoder network. The encoder encodes the source sequence into intermediary representations and the decoder decodes these representations to the target sequence. The decoder is usually auto-regressive, predicting the next token at each timestep from the tokens generated in the previous timesteps. This section details the workings of the recurrent sequence-to-sequence architecture and the transformer architecture, two popular architectures in Neural Machine Translation.

3.8.1 Token Representations

Before they can be processed by sequence-to-sequence models, the sequences must be converted into the appropriate vector form. Two ways of doing this are one-hot encoding and word embeddings. With one-hot encoding, for a dataset of vocabulary size V, each token is represented as a vector of dimension V, such that it has zeroes

everywhere except at the i_{th} position where it has a value of 1, with i being the position of the token in the vocabulary. As the number of tokens in the vocabulary increases, so does the number of dimensions in the input vectors, leading to larger models. An alternative to this is using word embeddings. Word embedding gives each word in the vocabulary a vector representation composed of float values. These can be learned such that words of similar context have corresponding vectors that are relatively close in space. The size of the word embeddings is a hyperparameter and does not have to grow with the vocabulary. Word embeddings are the most common form of word representation for translation models.

3.8.2 Recurrent Sequence-to-Sequence Models

The networks used in these models are usually varying types of RNNs such as standard RNNs, LSTMs, and GRUs. An RNN sequence-to-sequence model works as follows:

- At each time step in the encoder, the hidden state is h_t is calculated using the current source word embedding x_t and the previous timestamp's hidden vector h_{t-1}. In the initial time step, h_0 is typically initialized to a zero vector:

$$h_t = RNN_{enc}(x_t, h_{t-1}),\ h_0 = 0. \tag{3-6}$$

- The hidden state at the end of the source sentence contains information about the whole sentence (i.e., the source sentence has been encoded into the final hidden state) and is called the context vector c.
- In the decoder, the hidden state is initialized to the context vector c from the encoder. The hidden state s_t is subsequently calculated as a function of the previously predicted token y_{t-1} and the previous hidden state s_{t-1} as

$$s_t = RNN_{dec}(y_{t-1}, s_{t-1}),\ s_0 = c. \tag{3-7}$$

- This hidden state is then used to predict the next token

$$P(y_t | y_{t-1}, y_{t-2}, \dots, y_1, c) = g(s_t, y_{t-1}, c), \tag{3-8}$$

where $g()$ is some activation function that produces valid probabilities, usually a softmax.

One of the drawbacks of standard recurrent sequence-to-sequence models is that they encode varying-length input sentences into a fixed-length vector. This means that longer sentences may not be encoded with as much context information as shorter sentences. In fact, Cho *et al.* found in [45] that NMT performance deteriorates as the length of the input sentence and the number of unknown tokens increases.

In [42], Bahdanau *et al.* propose a method of allowing the model to automatically search for the parts of the source sentence that are relevant when at a particular timestep in the decoder. This is done by using a bidirectional RNN in the encoder to compute two hidden states for each token in the sequence. The decoder then predicts the target sequence using the previous hidden state, the previously predicted token, and the context vector for each target token. This model works as follows:

- The Bidirectional RNN reads the input sequence from the first token to the last with the forward RNN and calculates a series of hidden states $(\vec{h}_1, \dots, \vec{h}_T)$. The backwards RNN reads the input sequence from the last token to the first and calculates another series of hidden states $(\overleftarrow{h}_1, \dots, \overleftarrow{h}_T)$.
- The hidden state for each timestep is obtained by concatenating its forward and backward hidden states:

$$h_t = [\vec{h}_t^T; \overleftarrow{h}_t^T]^T. \tag{3-9}$$

This ensures that each token's hidden state has information about all the preceding and the following tokens. And since RNNs remember more recent information better, the annotation will be more focused on the tokens at and around time step t.

- In the decoder, the context vector is calculated as

$$c_t = \sum_{j=1}^{T_x} \alpha_{tj} h_j, \tag{3-10}$$

where $\alpha_{t,j}$ is the attention vector calculated at each time step. This attention vector is a function of the previous decoder hidden state and the encoder hidden states:

$$\alpha_{t,j} = f(s_{t-1}, h_j), \tag{3-11}$$

where $f()$ is some linear function.

- The decoder hidden state is calculated as

$$s_t = g(s_{t-1}, y_{t-1}, c_t), \tag{3-12}$$

and the probability of the next target token as

$$P(y_t | y_{t-1}, y_{t-2}, \dots, y_1, c) = k(s_t, y_{t-1}, c_t), \tag{3-13}$$

where $g()$ is some linear function followed by an activation function and $k()$ is some linear function followed a softmax.

3.8.3 Transformer Models

Although recurrent-based sequence-to-sequence models with attention perform well on NLP tasks, they take longer to train because of their recursive nature, and they still struggle with long-term dependencies. To tackle these issues, Vaswani *et al.* [44]

proposed the Transformer architecture, an architecture that is based on linear layers and makes use of no recurrent or convolutional layers. Similar to recurrent sequence-to-sequence models, the Transformer consists of an encoder and a decoder, each with its own layers. The encoder layers are made up of a multi-head attention (MHA) sub-layer followed by a position-wise feed-forward network (FFN). The layers in the decoder have an additional multi-head attention sub-layer. Each sub-layer has residual connections followed by layer normalization.

i. Positional Encodings

In sequence-to-sequence tasks, the order of the inputs is important. With recurrent architectures, the position information is automatically encoded into the model, but this is not the case for fully connected layers. The transformer adds positional information by using positional encodings. These encodings are added to the token embeddings in both the encoder and the decoder, before being passed to the layers. Positional encodings can be learned in the same way that word embeddings are learned but they can also be fixed. In the transformer paper, they use fixed sine and cosine functions with different frequencies for their positional encoding.

ii. Multi-Head Attention

The multi-head attention in the transformer computes attention independently on different parts of the inputs. Each of its three inputs, the query, the key, and the value, is split into N parts, called heads, where N is a hyperparameter. Each head, n, takes as input the n_{th} splits of the inputs and comprises three independent linear layers that correspond to the inputs. The inputs are passed through these layers to produce the Q, K, and V matrices that are then used to compute scaled dot-product attention according to

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V\ , \tag{3-14}$$

where d_k is the head dimension, obtained by dividing the model embedding dimension by N. The attention scores of each head are then concatenated and passed through a linear layer to produce the output of the multi-head attention sub-layer. Figure 3-6 shows a graphical representation of the multi-head attention.

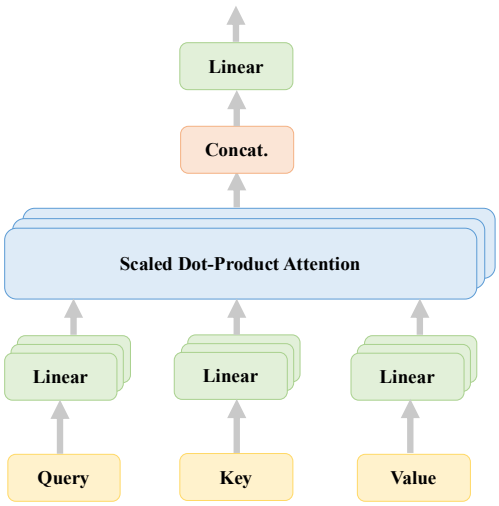


Figure 3-6: Graphical representation of multi-head attention.

Multi-head attention is used in the transformer as either self-attention, where the queries, keys, and values come from the same network or encoder-decoder attention, where the keys and the values are from the encoder, but the queries are from the decoder. Self-attention is present in both the encoder and decoder layers, and it allows the models to learn the relationships within the source sequences (in the encoder) and the relationships within the target sequences (in the decoder). Encoder-decoder attention is only present in the decoder and learns the relationships between the source and target sequences. Its keys and values are the outputs from the encoder layers and the queries are the outputs from the decoder self-attention.

When computing the attention scores, it is common to apply a mask to the scaled dot-product before applying the softmax. The type of mask is dependent on whether the

MHA is within the encoder or the decoder. In the encoder self-attention, masking is used to prevent the model from attending to paddings in the inputs. In the decoder self-attention, in addition to masking paddings, the mask also ensures that at each timestep, the model can only attend to previous tokens. In this way, the model is prevented from cheating by "looking into the future".

iii. Position-Wise Feed-Forward Networks

The feed-forward network consists of two linear layers with a ReLU activation between them such that

$$FFN(x) = W_2 \max(0, W_1 x + b_1) + b_2. \tag{3-15}$$

The layers in this network are applied to each position separately.

The transformer model improved the performance of machine translation models with less computational complexity and different variations of the model continue to produce state-of-the-art performance in other NLP tasks.

3.9 NMT Metrics

The most common evaluation metrics for Natural Machine Translation are discussed below. In the explanations below, the hypothesis refers to the translation produced by the model that is being evaluated and the reference refers to the ground truth translation.

3.9.1 BLEU

BiLingual Evaluation Understudy (BLEU) [16] is the most common NMT metric. It measures how well the hypothesis matches the reference translations by counting the percentage of n-grams in the hypothesis that also occurs in the references. BLEU is a

precision-oriented metric in that it measures how much of the system output is correct, rather than measuring whether the references are fully reproduced in the system output. [46]. It is usually calculated and stated as BLEU-n, where n is the n-gram and ranges from 1 to 4 (e.g. BLEU-1)

3.9.2 ROUGE

Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [47] is a set of measures designed to automatically determine the quality of summaries generated by a computer (hypotheses) by comparing them to those generated by a human (references). These measures are ROUGE-N, ROUGE-L, ROUGE-W and ROUGE-S. The most common in Sign Language Translation is ROUGE-L, which measures the longest matching sequence of words.

3.9.3 METEOR

Metric for Evaluation of Translation with Explicit ORdering (METEOR) [48] was designed to address some observed weaknesses in the BLEU metric. It is a recall-oriented metric, whereas BLEU is generally a precision-oriented metric. It evaluates a hypothesis by computing a score based on explicit word-to-word matches between the hypothesis and the reference. Unlike BLEU which only calculates precision, METEOR calculates both precision and recall, and combines the two [46].

3.9.4 WER

Word Error Rate (WER) measures the required numbers of deletion, insertion, and substitution operations required to transform the hypothesis into the reference sequence [49]. WER is the most common metric for evaluating Sign Language Recognition performance, but it is not typically used for Sign Language Translation performance.

3.10 Deep Learning Software

This section reviews three of the most common deep learning software.

3.10.1 TensorFlow

TensorFlow is an open-source software library developed by Google for numerical computation. It provides an interface for expressing machine learning algorithms and an application for executing these algorithms. A calculation expressed using TensorFlow can be carried out in a wide range of heterogeneous systems, from mobile devices such as phones and tablets to large-scale distributed systems of hundreds of machines and various computing devices such as graphic processing unit (GPU) cards [50]. It supports a variety of applications, with a focus on training and inference on deep neural networks [51]. Tensorflow can be interfaced with using programming languages like Python, Java, C/C++, Ruby and more. It has CUDA support which allows for parallel computation and has pre-trained models. Keras provides a Python interface for artificial neural networks and acts as an interface for the TensorFlow library.

3.10.2 PyTorch

PyTorch is an open-source Python library that performs immediate execution of dynamic tensor computations with automatic differentiation and GPU acceleration while maintaining performance comparable to the fastest current libraries for deep learning. It was designed to be Pythonic because data scientists are already familiar with the language and its tools. It was also designed to be easy to use and to produce compelling results [52]. It can be interfaced with using Python, C++ and Julia and has CUDA support and pre-trained models.

3.10.3 Caffe

Caffe is a deep learning framework that supports many different types of deep learning architectures, although its development was motivated by image recognition and segmentation tasks. It was designed to be as modular as possible, allowing easy extension to new data formats, network layers, and loss functions [53]. It is open-source and can be interfaced with using Python, C++, and Matlab. It has CUDA support and pre-trained models.

The Matlab Deep Learning Toolbox, Theano, and BigDL are also common deep learning software.

Chapter 4: Deep Learning in Sign Language Translation

Since the rise of deep learning in the early 2010s, deep learning methods have been applied in numerous fields, including the field of automatic Sign Language Translation. CNNs replaced classical feature extraction methods in statistical Sign Language Recognition models and RNNs were used in conjunction with Hidden Markov Models to improve the performance of statistical models. Although the use of deep learning methods in sign language related tasks has been common since the early 2010s, the development of end-to-end deep learning models for vision-based Sign Language Translation is fairly new and was only formalised in 2018 as a Neural Machine Translation task. This chapter reviews some of the literature involved in Sign Language Recognition and Translation.

4.1 Sign Language Recognition vs. Sign Language Translation

Most of the research that has been conducted on the automatic translation of sign language has been on Sign Language Recognition (SLR) rather than Sign Language Translation (SLT) [6]. Sign language recognition treats the problem as a gesture recognition problem where individual signs are recognised in the order that they appear in the videos. This is not an adequate solution since sign languages have different sentence structures from spoken languages. Even with all the recognised signs, a person who does not know sign language would not be able to make sense of what is being communicated. Figure 4-1 demonstrates the difference between SLR and SLT.

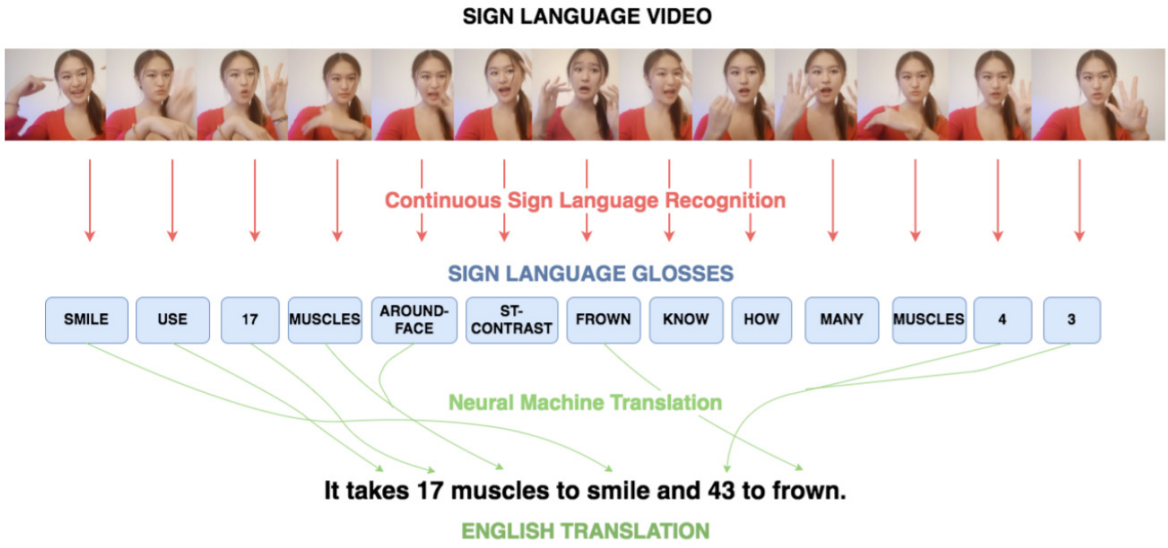


Figure 4-1: A demonstration of the difference between Sign Language Recognition and Sign Language Translation. The result of Sign Language Recognition does not produce a correct English sentence (SMILE USE 17 MUSCLES AROUND-FACE ST-CONTRAST FROWN KNOW HOW MANY MUSCLES 4 3) but the result of translation does (It takes 17 muscles to smile and 43 to frown) [54].

Sign language recognition involves complex processes such as motion modelling, motion analysis, pattern recognition and machine learning [55]. The methods used in sign language and gesture recognition can be classified into two groups: sensor-based methods and vision-based methods. Sensor-based methods use apparatus such as gloves fitted with sensors to capture motion and orientation data as features that are then used to classify the signs. For example, Escudeiro *et al.* [56] use the Microsoft Kinect and sensor gloves to perform Portuguese Sign Language recognition. They use SVMs for classifying handshapes and Dynamic Time Warping for classifying movements. Seymour and Tšoeu [21] use gloves for the recognition of the letters of the alphabet and digits in South African Sign Language. Although sensor-based methods may be better at modelling complex movements and handshapes for Sign Language Recognition, these methods are invasive, and the apparatus can be expensive. They also do not have access to non-manual features, one of the linguistic components of sign language.

Vision-based methods use cameras and image processing techniques to extract features that are used for sign classification. Some of the challenges that arise in vision-based SLR/SLT are:

- There is usually no sign-level supervision/temporal alignment between videos and glosses as it is not easy to predict where one sign ends and the next one begins.
- The execution speed of a given gloss may indicate a different meaning or the particular signer's attitude. For instance, signers would not use two glosses to express "run quickly", but they would simply speed up the execution of the involved sign [57].
- There are few publicly available datasets for Sign Language Recognition or translation and those that are available tend to be limited in vocabulary, size, or domain.
- The structure of the environment such as noise, background illumination, variation of viewpoint and speed of movement affects the predictive ability of the system [55].
- The collection and annotation of continuous sign language data is a laborious and time-consuming task and there are not a lot of guidelines on the design and development of sign language datasets.

The rest of this chapter discusses vision-based methods for SLR and SLT.

4.2 Sign Language Recognition

Sign language recognition can be divided into two tasks: Isolated Sign Language Recognition (ISLR), which focuses on recognizing individual signs, and Continuous Sign Language Recognition (CSLR) which recognizes multiple signs in a video in the order that they are signed. SLR is considered a weakly supervised sequence-to-sequence

problem because of the lack of gloss-level or frame-level annotations. SLR models are typically made up of two parts, a feature extractor for spatial embedding and a sequence model to model the long-term dependencies of the sequence. Since CNNs have shown great success in handling visual tasks, the feature extractor is usually either a 2D-CNN or a 3D-CNN. Some common sequence models that are used are Hidden Markov Models (HMMs) and RNNS with connectionist temporal classification (CTC) [58].

A few datasets have been released for work on both ISLR and CSLR. Most of the datasets are recorded in controlled environments and have similar backgrounds and clothing between the videos. This means that SLR is still a bit far from real-world application. Towards this, however, some of the datasets are designed to be signer-independent or have signer-independent subsets. Table 4-1 lists some of the publicly available datasets for SLR.

Table 4-1: Some publicly available datasets for Sign Language Recognition.

Dataset	**Language**	**Type**	**Signers**	**Videos**	**Classes**	**Modalities**	**Year**
WLASL2000 [59]	ASL / English	Isolated	119	21,083	2,000	RGB	2020
ASL1000 [60]	ASL / English	Isolated	222	25,513	1,000	RGB	2020
GSL [57]	GSL / Greek	Continuous	7	10,290	310	RGB+D	2020
CSL [61]	CSL / Chinese	Continuous	50	25,000	178	RGB+D	2018
PHOENIX-Weather 2014 [62]	DGS / German	Continuous	9	6,861	1,558	RGB	2014
Signum [63]	DGS / German	Continuous	25	21,060	585	RGB	2008

CSLR research is mostly conducted on the publicly available RWTH Phoenix Weather 2014 dataset [64]. It contains German Sign Language (DSGS) aired by the German public TV station PHOENIX in the context of weather forecasts. All videos were

recorded at 25 frames per second and a resolution of 210 × 260. Another popular dataset is the CSL dataset. It was recorded using the Microsoft Kinect and provides RGB, depth and body joints modalities in all videos, although most work only focuses on the RGB modality.

One of the first applications of deep learning in CSLR was implemented by Koller *et al.* [65] with the formulation of an Expectation-Maximization-based algorithm which integrates CNNs with Hidden-Markov-Models (HMMs) for handshape recognition. They used a 2D-CNN fed with cropped hand regions and trained their model iteratively on the RWTH Phoenix Weather 2014 and Signum datasets and achieved significant improvement in WER over the state of the art at the time (45.1% WER from 53.0% WER and 7.6% WER from 10.0% WER respectively). Later in [49] they applied this model for Continuous Sign Language Recognition but trained it in an end-to-end manner. They achieved 38.3% WER on the RWTH Phoenix Weather 2014 dataset and 7.4% WER on the Signum dataset, surpassing models that made use of multiple modalities even though their model only used right-hand features.

In [66], Koller *et al.* embedded a CNN and a bidirectional LSTM into an HMM. They introduced the BLSTM to the model because motion plays a significant role in sign languages and only relying on the HMM state sequence to capture temporal change may not have been enough. They also proposed an iterative re-alignment algorithm that treats the provided training labels as weak labels and refines the label-to-image alignment. The deep CNN-BLSTM network is trained end-to-end but once it is embedded into an HMM, the resulting model corrects the frame labels and continuously improves its performance in several realignments. This model also outperformed the state-of-the-art models at the time on both the RWTH Phoenix Weather 2014 dataset and the SIGNUM dataset by achieving WERs of 26.8% and 4.8% respectively. Their results showed that whole frames perform better than cropped

hand regions. This result proves the multimodal nature of sign languages and that other features such as facial expressions and pose information are important parts of sign languages.

Camgoz *et al.* [67] recognized this multimodal structure of sign languages and proposed a deep learning framework for explicitly modelling the subunits of sign languages during sequence-to-sequence learning. Each subunit consisted of a 2D-CNN for spatial modelling followed by a BLSTM for temporal modelling and a CTC decoder. Their results showed that having a handshape classifier as a subunit and adding it to a full frame subunit increases the performance of the model on the RWTH Phoenix Weather 2014 dataset. This is however impractical as handshape annotations are not always available in sign language datasets. However, adding a sign-level subunit that uses cropped hand regions to one that uses whole frames produces results that are close to their best model (42.6% WER vs. 42.1% WER). This may be because the model has a "zoomed in" view of the hands in the cropped hand regions which allows it access to some information that it does not have access to in the full frames.

Cui *et al.* [68] proposed a model that uses temporal CNNs and an LSTM for Sign Language Recognition. They used a pre-trained 2D-CNN [69] for feature extraction and then applied temporal convolution to these representations to obtain spatiotemporal representations of the input videos. In this way, they effectively captured spatio-temporal information without the computational cost of 3D-CNNs. A bidirectional LSTM with CTC loss was then applied to the extracted features. They trained their model iteratively by using the proposed alignments by the model to fine-tune the feature extractor and then retrain the BLSTM. This iterative process is similar to that of Pu *et al.* [70] which they claim is an effective way of training the lower layers in deep architectures. They also proposed a detection net that is parallel to the LSTM to implicitly locate sign glosses in the temporal sequences. Their results

on the RWTH Phoenix Weather 2014 dataset showed that their model with the 2D-CNN and temporal convolutions performs better than the same model using a 3D-CNN (47.3% WER vs. 77.6% WER). However, the 2D-CNN was first fine-tuned on hand images from the dataset, and they do not say whether this was the case for the 3D-CNN.

In [61], Huang *et al.* proposed a Hierarchical Attention Network with Latent Space (LS-HAN) for Chinese Sign Language Recognition. For feature extraction, they used a faster R-CNN with a compressive tracking model to track hands. They designed a two-stream 3D-CNN that takes both the entire video frames and cropped hand patches with a late fusion mechanism achieved through shared final fully connected layers. In this way, the CNN is allowed to "zoom" into the hands. The relevance between the projections of the sentences and videos into the latent space was measured using the Dynamic Time Warping (DTW) algorithm. This model contains two encoders and a decoder. Each encoder is a bidirectional LSTM with an attention layer and the decoder is a single LSTM. The clip encoder encodes the video clips aligning to a sign. They tested their model on the CSL dataset and the RWTH-PHOENIX-Weather dataset and obtained WERs of 17.3% and 38.4% respectively.

4.3 Sign Language Translation

Not as much research has been conducted on Sign Language Translation (SLT) as on Sign Language Recognition. Most of the research on Sign Language Recognition is conducted as an initial step towards Sign Language Translation. Because of this, some of the SLT models have two stages: a recognition stage (sign to gloss) and a translation stage (gloss to spoken language). SLT is considered a translation problem and therefore usually consists of a feature extractor for spatial embeddings and sequence-to-sequence models adopted from text-based Neural Machine Translation. SLT also has fewer

publicly available datasets. Table 4-2 shows some of the publicly available datasets for Sign Language Translation.

Table 4-2: Publicly available datasets for Sign Language Translation.

Dataset	**Language**	**Gloss**	**Signers**	**Videos**	**Vocab size**	**Modalities**	**Year**
How2Sign [5]	ASL / English	Not public	9	35,191	$> 60{,}000$	RGB (Multiview), Pose	2021
CSL-Daily [71]	CSL / Chinese	Yes	10	20,654	2,343	RGB	2021
CSL	CSL / Chinese	Yes	50	25,000	178	RGB+D	2018
RWTH-PHOENIX-Weather 2014T [6]	DGS / German	Yes	9	8,257	2,887	RGB	2018

SLT research is mostly conducted on the RWTH-PHOENIX-Weather 2014T as it was the dataset that was released along with the paper that formalized the task [6]. The creators of the CSL dataset claim that it was designed in such a way that it can be used for both SLR and SLT.

Kumar *et al.* [15] proposed a model that translates ASL to English in two stages, video to gloss and gloss to English using the National Center for Sign Language and Gesture Resources (NCSLGR) Corpus. Their first stage (recognition) uses Gaussian Blur filter and Otsu's Binarization to segment the hands and face from the video frames and then Angular Hashing for feature extraction. They then used an LSTM for the sequence modelling. Instead of using CTC like most other sequence models where the number of inputs is different to the number of outputs, they added a learnable parameter that acts as a threshold that signifies whether the neural network can provide a classification for a gloss at each time step. Their second stage uses an LSTM-based encoder-decoder model which takes a sequence of glosses as inputs and outputs an

English sentence. For translation, their best model achieved a BLEU score of 49.7. They did not say, however, whether this score was achieved from testing the entire two-stage system or just from testing the second stage. This is important information as the BLEU score for the entire system would be lower than that of the second stage due to possible errors from the first stage. They also did not specify which n-gram they used to calculate their BLEU score.

Guo *et al.* [17] introduced HLSTM, a hierarchical LSTM model for translating sign language. They fed video frames into a C3D [72] to extract features and then passed them through a key clip mining module that calculates the residual sum of square error between the current frame and previous frame to determine which frames can be linearly reconstructed by the previous ones. This distinguishes the important clips from the less important ones. The outputs of the C3D are also fed to $LSTM_1$, the first LSTM in the HLSTM. Before $LSTM_2$, they applied attention-aware pooling on the clips based on the results from the key clip mining. They then fed these outputs to an LSTM-based encoder($LSTM_2$)-decoder($LSTM_3$) network. Their encoder and decoder have shared weights. They evaluated their model on the CSL dataset and achieved BLEU-4, ROUGE-L and METEOR scores of 92.8, 95.1 and 70.3 respectively for the signer independent split and BLEU-3, ROUGE-L and METEOR scores of 20.7, 50.3 and 20.5 respectively on the unseen sentence split.

In [73], Guo *et al.* proposed HRF, an adaptation of HLSTM for two channels, RGB and skeleton. HRF is two HLSTMs whose $LSTM_3$ outputs are fused to predict the next word. This model achieved a BLEU-4 score of 99.0, a ROUGE-L score of 99.4 and a METEOR score of 81.7 on the signer independent split (an improvement on HLSTM) and BLEU-3, ROUGE, and METEOR scores of 12.7, 44.9 and 17.1 respectively on the unseen sentence split (diminished performance compared to HLSTM). This could be explained by the fact that pose information from skeleton data removes information

about the physical appearance of the signer, so the model benefits in the signer-independent split. In their paper, we also see that an HLSTM using skeleton information achieved a performance of 94.7 BLEU-4, 96.0 ROUGE-L and 73.8 METEOR on the signer independent split and 0 BLEU-3, 31.5 ROUGE-L and 10.3 METEOR on the unseen sentence split.

In 2018, Camgoz *et al.* [6] formalized the approach of the full SLT task as a Neural Machine Translation task. They also introduced the RWTH-PHOENIX-Weather 2014T dataset. They used a 2D-CNN for spatial embedding and a GRU-based encoder-decoder model with attention and compared the results of using the model in an end-to-end manner (from sign to spoken German) and using it in a two-stage system (sign to gloss to spoken German). Their results showed that the two-stage system (18.13 BLEU-4, 43.80 ROUGE-L) performed better than the end-to-end model (9.58 BLEU-4, 31.80 ROUGE-L) as the two-stage system allows for intermediate gloss-level supervision.

Ko *et al.* [74] proposed a Korean Sign Language translation system based on human keypoints estimated by OpenPose. They also constructed the first Korean Sign Language dataset called the KETI sign language dataset. They used the concatenations of the keypoints' z-scores as the feature vectors for each frame and sampled 50 frames from each sign video. For translation, they compared four sequence-to-sequence models: standard RNN-based, with Bhadanau attention [42], with Luong attention [75] and a transformer. The RNN-based encoder-decoder models that they used are based on two-layer Gated Recurrent Units (GRU). Their training set consisted of 9,432 videos, 1,048 videos for the validation set and 4,192 for the test set with signers that are not in the training or validation set. Their best-performing model was the transformer, and it achieved a BLEU score of 92.90 on the validation set and 66.58 on the test set. Such high BLEU scores could be a result of the fact that the dataset has

a large number of videos and a small vocabulary of 419 words, so the model had more examples to learn from.

In [18], Camgoz *et al.* proposed a transformer model, Sign2(Gloss+Text), that jointly learns CSLR and SLT in an end-to-end manner. They used features from a CNN-LSTM-HMM [76] hybrid model that was pre-trained on a CSLR task for spatial embedding. In their transformer model, the outputs from the encoder are not only passed to the decoder for translating but they are also passed to a linear and a softmax layer with CTC for gloss predictions. In this way, they added gloss-level supervision while also allowing the decoder direct access to information from the videos. This architecture is similar to that of Pu *et al.* [70] except that theirs was RNN-based and their goal was CSLR, not SLT. They (Camgoz *et al.*) evaluated their model on the RWTH-PHOENIX-Weather 2014T dataset and outperformed the state-of-the-art with a 20.17 BLEU-4 score.

Cabot *et al.* [77] adapted Sign2(Gloss+Text) to the How2Sign dataset. Due to the hybrid CNN-LSTM-HMM feature extractor implementation not being publicly available, they used the I3D from the How2Sign paper that was trained on an ASL recognition task for feature extraction. They also used the translations as the gloss because the How2Sign does not yet have gloss representations. They conducted various experiments to find the ideal parameters for the model and obtained a BLEU-4 score of 2.21 on their best-performing model.

Li *et al.* [78] proposed that extracting features in a frame-wise manner is insufficient for Sign Language Translation as it ignores the temporal dependencies between sign gestures. Motivated by this, they proposed TSPNet, a model that learns features from video segments instead of single frames. TSPNet uses a sliding window approach to segment input videos with multiple window sizes (8, 12 and 16) and pass each segment

through an I3D [79] that was finetuned on two Isolated Sign Language Recognition ASL datasets. The aim of the multi-scale segment representation was to alleviate the impact of imprecise video segmentation. They implemented inter-scale attention to ensure that the model can learn both the fine-grained movements (smaller scales) and transitional movements (from larger scales) and intra-scale attention to ensure that the model can use non-local information to learn the context when learning gesture semantics. They then fed the outputs of these attentions to a transformer decoder for predicting the spoken language translations. Their experiments on RWTH-PHOENIX-Weather 2014T produced a BLEU-4 score of 13.41 and a ROUGE-L score of 34.96. These results were achieved when the inter-scale and intra-scale attentions were applied jointly. When applied sequentially (inter-scale first then intra-scale), the BLEU-4 and ROUGE-L scores were 12.97 and 34.77 respectively, indicating that the model gains from joint intra-scale and inter-scale attention. They also showed that when using a single scale, the model performance decreases.

Narayanan *et al.* [80] proposed the Multi Context Transformer, which uses the same multi-scale pre-processing as Li *et al.* above. However, instead of the inter-scale and intra-scale attention, they used a multi-context encoder where each layer of the encoder uses inputs from a different scale. The outputs from the layers are concatenated and fused with a fully connected layer and then passed to a decoder layer for translation. One of their main aims was to reduce the number of model parameters to make the model suitable for real-world applications. Their use of video segments instead of individual frames was one of the ways of doing this as it reduces the number of parameters in the multi-head attention (MHA). Another way was making the feed-forward layer dimension less than the MHA dimension D instead of the standard 4D recommended in the transformers paper. On the RWTH-PHOENIX-Weather 2014T, with one multi-context encoder layer and 6 decoder layers, their model obtained a

BLEU-4 score of 11.62 and a ROUGE-L score of 34.33. This model used 30.88% less parameters than TSPNet. but only dropped the BLEU-4 score and ROUGE-L scores by 1.79 and 0.63, respectively. The model obtained a similar BLEU-4 score to the TSPNet when they used a single scale of 16 (11.61 BLEU-4).

Gan *et al.* [81] hypothesized that skeleton information is beneficial in Sign Language Translation. They proposed SANet for jointly learning skeleton extraction and SLT. Similar to Li *et al.* and Narayanan *et al.*, they segmented the videos into clips but used P3D [82] instead of an I3D to extract the clip-level features. They also extracted frame-wise features from the video using a compressed VGG [69] model. It is through the outputs of the VGG's third and fourth layers that they constructed their skeleton extractor. The skeletons extracted from the frames were added to the clips as a fourth channel before running them through the P3D network. They also used the skeleton features to construct a skeleton-based Graph Convolutional Network which produces a scale factor for the fused frame level and clip level features to determine the meaningful clips from the less meaningful ones. The resulting vector was then passed to an LSTM-based encoder-decoder model. In their three-layered BiLSTM, they designed MemoryCell to change the dimensions of hidden states. They evaluated their model on the RWTH-PHOENIX-Weather 2014T dataset as well as the CSL dataset. On RWTH-PHOENIX-Weather 2014T, their model achieved a BLEU-4 score of 24.8 and a ROUGE-L score of 54.8. On the CSL dataset, the BLEU-4 and ROUGE-L scores were 99.0 and 99.6, respectively. Their results on various experiments showed that their model benefits from all the different components of the model (skeleton extraction, the skeleton channel, frame level features, clip level features, skeleton-based GCN and MemoryCell).

Chen *et al.* [83] proposed to tackle the issue of a lack of data presenting a bottleneck for training effective Sign Language Translation models by using multi-modal transfer

learning. They proposed a baseline model that is made up of a visual network that converts signs to gloss and a translation network that transforms the gloss into text. They took advantage of larger datasets from related domains to progressively pretrain their model. They pre-trained the visual network on an action recognition dataset and then on the WLASL dataset before fine-tuning it on continuous SLR. They also pretrained their translation network using ground truth gloss annotations. They performed experiments on RWTH-PHOENIX-Weather 2014T and CSL-Daily and achieved BLEU-4 scores of 28.39 and 23.92 respectively.

Almost all of the research above is focused on performing well on the whole task but not much investigation has gone into comparing which elements perform best for specific stages of the task (e.g., 2D-CNN vs. 3D-CNN for feature extraction while keeping all other elements of the model the same). Ananthanarayana *et al.* conducted various experiments [84] to determine which combinations of deep learning methods perform better for Sign Language Translation without any gloss-level supervision. They conducted ablation studies comparing the performance of different input features (pose vs. CNN) and sequence-to-sequence models (RNN-based vs. RNN-based with attention vs. transformers) on three datasets (DSGS, CSL and ASL). From their experiments, they found that:

- On the DSGS dataset, pose information and pre-trained deep CNNs had similar performance with all the sequence-to-sequence models. Transformers performed better.
- On the CSL dataset, pose information led to better results than using CNNs, and the transformer model performed better.
- On the ASL dataset, pose information performed better with a transformer but CNNs performed better with RNN-based models. CNNs with RNN-based models outperformed pose with transformer.

These results give some useful guidelines on SLT. One shortfall of the paper, however, is the fact that experiments were not done on the effectiveness of 3D-CNNs in feature extraction although they have been used extensively in the literature [17], [77], [78], [80].

4.4 SASL Recognition and Translation

Research conducted on SASL includes the development of a rule-based machine translation for translating English to SASL by van Zijl [85], the development of a low-cost glove for the development of an SASL dataset by McInnes [20], the glove-based recognition of 31 SASL static signs by Seymour and Tšoeu [21] and the upper body pose recognition towards the visual translation of SASL by Achmed [22]. To the best of our knowledge, no research on Neural Sign Language Translation for South African Sign Language has been conducted.

Chapter 5: Dataset Design

Data is one of the biggest contributors to the performance of deep learning models. The size and quality of the dataset have a significant impact on the output and performance of the model. As such, the design and collection of a dataset for application in deep learning is a crucial process. To solve the engineering problem of building a system that can capture the linguistic properties of South African Sign Language and use these properties to automatically translate it to English, we needed a dataset comprising parallel SASL and English sentences. We opted to use vision-based methods instead of glove-based methods because the vision-based methods are less invasive on the signer and do not ignore non-manual features. Since there was no such publicly available dataset, we had to design and collect a new one. This chapter explores some of the available benchmark datasets for Neural Sign Language Translation. It then details the design and development process of the SASL dataset presented in this research.

5.1 Benchmark Datasets

Our first research question is shown below.

- What are the considerations to make when designing and developing a parallel SASL and English corpus so that it can be suitable for the automatic translation task and be useful for further research into the field?

To answer this question, we looked at other benchmark datasets for Neural Sign Language Translation. There are few publicly available datasets for Neural Sign Language Translation. Over the past few years, most of the research on NSLT was conducted on the RWTH-PHOENIX-Weather 2014T dataset as this was the only

public dataset suitable for the task. In 2021, the How2Sign dataset was also made public for SLT. The two datasets are studied below.

5.1.1 RWTH-PHOENIX-Weather 2014T

The RWTH-PHOENIX-Weather 2014T dataset was released in 2018 along with the paper that formalized NSLT as a Neural Machine Translation task. The dataset consists of video data, gloss and translations and can thus also be used for Sign Language Recognition. Similar to the PHOENIX-Weather 2014 dataset, it contains German Sign Language (DSGS) from weather forecasts. The authors ensured that the validation and test sets of PHOENIX-Weather 2014 do not overlap with the PHOENIX-Weather 2014T training set and vice versa. Table 5-1 shows some key statistics of the dataset.

Table 5-1: Statistics of the RWTH-PHOENIX-Weather 2014T dataset

	Train	**Validation**	**Test**
Sign Language	German Sign Language (DSGS)		
Modalities	RGB (210x260 25 fps)		
Number of Signers	9		
Number of Segments	7,096	519	642
Vocabulary Size	2,887	951	1,001
Out of Vocabulary Words	-	57	60
Singletons	1,077	-	-

i. Image Data

The dataset is made up of images of size 210x260 recorded at 25 fps. The format of the image data in the dataset is uniform. The signers are all wearing black, and the background is grey. There is no background noise and there is only ever one person in the frame. All the signers are centred in the image and standing at similar distances

from the camera such that their entire signing space is captured. Figure 5-1 shows a sample image from the training set of the dataset.

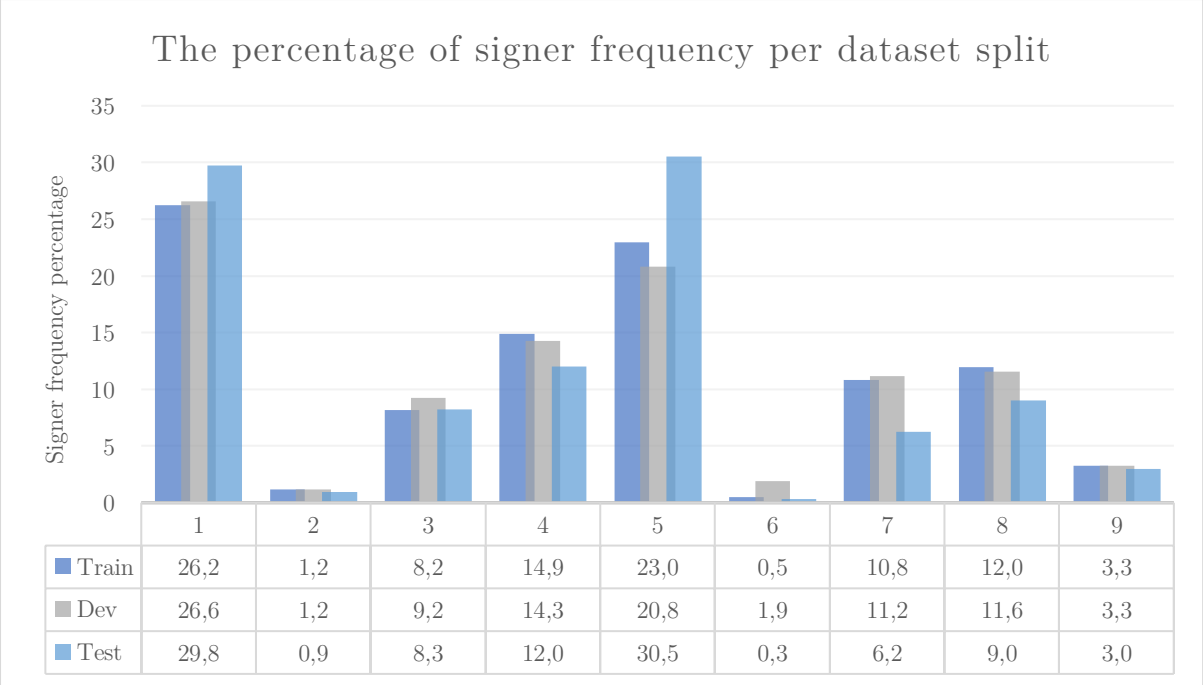
Figure 5-1: Sample image from the RWTH-PHOENIX-Weather 2014T training set [6].

ii. Signer Distribution

The dataset comprises data from nine professional sign language interpreters. Although the dataset consists of data from nine signers, the distribution of the video segments among the signers is not even. About 50% of the training set is made up of only two signers and the signer with the fewest segments only makes up 0.5% of the training set. This distribution is similar in the other sets as well. Figure 5-2 gives a more detailed picture regarding the distribution of signers in the training, validation, and test sets of the dataset.

iii. Frame Distribution

The dataset is available as a set of frames for each segment. The number of frames per segment ranges from 16 to 475 but the average number of frames per segment is 116. At 25 fps, this equates to an average of 4.66 seconds per segment. Figure 5-3 shows the distribution of frames per segment in the training set of the dataset.



	1	2	3	4	5	6	7	8	9
Train	26,2	1,2	8,2	14,9	23,0	0,5	10,8	12,0	3,3
Dev	26,6	1,2	9,2	14,3	20,8	1,9	11,2	11,6	3,3
Test	29,8	0,9	8,3	12,0	30,5	0,3	6,2	9,0	3,0

Figure 5-2: Distribution of signers in the train, validation, and test sets for the RWTH-PHOENIX-Weather 2014T dataset.

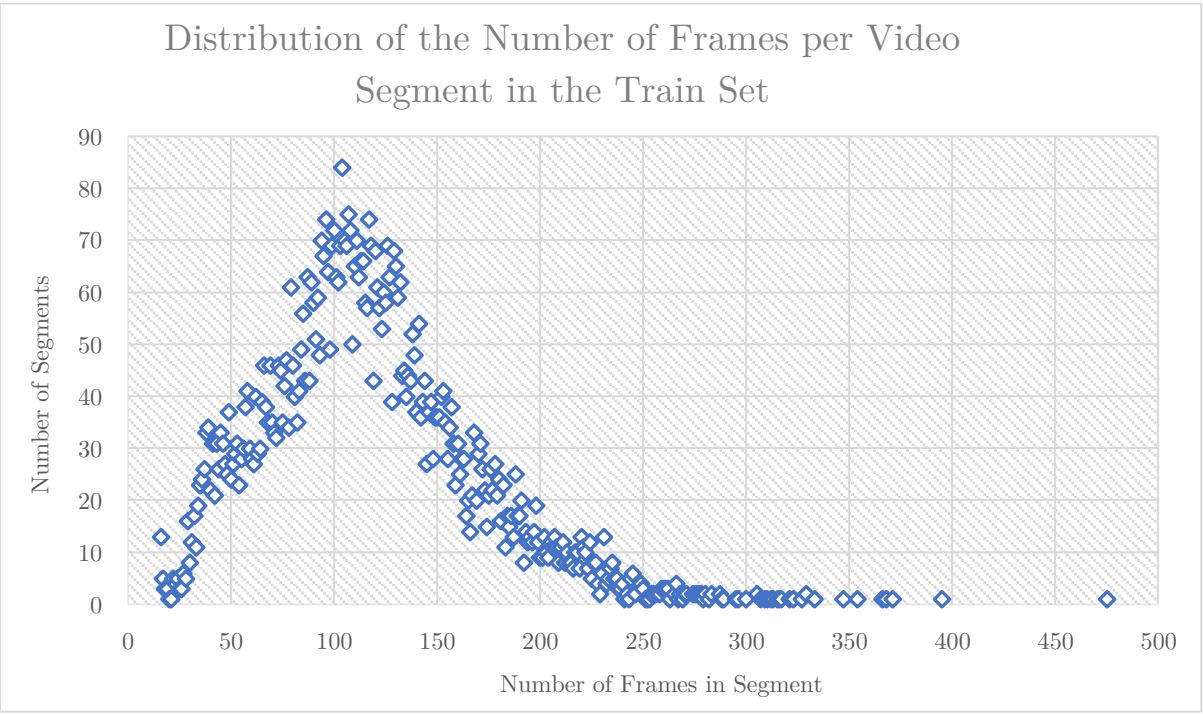


Figure 5-3: The distribution of the number of frames per segment in the RWTH-PHOENIX-Weather 2014T train set.

5.1.2 How2Sign Dataset

The How2Sign Dataset was released in 2021. It consists of a parallel corpus of speech and transcriptions of the How2 dataset instructional videos and their corresponding American Sign Language (ASL) translation videos. It has a total of 80 hours of sign language videos with multiple views. They recorded most of their videos in a green screen studio and the rest in a panoptic studio to extract 3D keypoints from them. This analysis only focuses on the green screen studio subset. Table 5-2 shows the key statistics of the subset of the How2Sign dataset that was recorded in the green studio.

i. Image Data

The green screen studio subset was recorded from two views (front and side) and includes automatically extracted 2D keypoints from the videos. The videos were recorded with a depth sensor and an HD camera in front of the signer and one other HD camera on the side of the signer. The HD cameras have a resolution of 1280x720, and the videos were recorded at 30 fps. In the videos, the signers are all seated and have a green background. The signers wear different coloured clothing but still maintain good contrast with the background. Similar to the RWTH-PHOENIX-Weather 2014T dataset, the signers are centred in the frame and are seated at similar distances from the camera. Figure 5-4 shows sample RGB images from the How2Sign dataset.

ii. Signer Distribution

The dataset comprises a total of 11 signers, six males and five females, but only nine signers are present in the green screen studio subset. The clip-level annotation of the dataset does not include signer IDs which makes it difficult to analyse the signer distribution in the dataset without manually annotating it ourselves.

Table 5-2: Key statistics of the green studio subset of the How2Sign dataset.

	Train	**Validation**	**Test**
Sign Language	American Sign Language		
Modalities	Multi-view HD RGB + D (1280x720 30 fps)		
Number of Signers	9		
Number of Segments	31,128	1,741	2,322
Vocabulary Size	15,686	3,218	3,670
Out of Vocabulary Words	-	413	510
Singletons	4,978	-	-

Figure 5-4: Sample RGB images from the How2Sign dataset [5].

iii. Video Clips and Sentences

Unlike the RWTH-PHOENIX-Weather 2014T dataset, the How2Sign dataset is available in either whole video form or sentence-level clips. According to their paper, some of the whole videos were recorded multiple times by a different signer. Each sentence-level clip contains an average of 162 frames, corresponding to 5.4 seconds at 30 fps.

From the analysis of the above benchmark datasets, we identified the following common attributes:

- The datasets consist of sentence-level clips.

- The lengths of the sentences in the datasets, and therefore the durations of the video segments, are not fixed.
- The video segments have a uniform background, where the signer is centred in the frame. The signers are all wearing clothes that contrast well with the background.
- Each video segment only consists of one signer.
- The datasets comprise multiple signers.
- The video segments are not full-body recordings but only cover the signer's signing space.

We considered all these attributes when developing our SASL dataset.

5.2 SASL Dataset

The rest of this chapter details the design and development of the SASL dataset presented in this research.

The dataset constitutes 5047 parallel South African Sign Language and English sentences in the domain of government and politics. It was recorded with six participants at the University of Cape Town (UCT) and North West University (NWU) over 11 non-consecutive days. The reference text is a subset of the Autshumato English and Setswana parallel machine translation corpus [86]. The corpus was collected as part of a government-funded project to develop, release and support open-source translation technologies for South Africans. They mainly collected their data from the South African government domain. We decided that this data would be suitable for our dataset because it was collected and developed for a machine translation task. For our dataset, the text was randomly selected such that the vocabulary size of the dataset would not exceed 3500 to ensure that most words in the dataset would be well represented since the dataset would be small. The dataset does

not yet include gloss annotations. Table 5-3 shows the key statistics of our SASL dataset.

Table 5-3: Key statistics of the SASL dataset

	Train	**Validation**	**Test**
Sign Language	South African Sign Language		
Modalities	RGB (1280x720 30 fps)		
Number of Signers	6		
Number of Segments	4 139	454	454
Vocabulary Size	3 263	1 407	1 437
Out of Vocabulary Words	-	264	277
Singletons	1 030	-	-

5.2.1 Participants

Developing a new SASL dataset meant that we had to find suitable participants who would translate our reference text into SASL. When selecting participants, we aimed for professional sign language interpreters who have experience translating English into SASL while putting a preference on native signers. All participants were provided with an information sheet and consent form explaining the purpose of the project and how the data will be used.

Our dataset comprises six participants. Three of the participants are native signers and professional sign language interpreters. The other three participants are trained sign language interpreters. The native signers make up the bulk of the dataset. The participants vary in physical appearance and complexion and may use different signs for the same word. The dataset is not balanced in terms of race and gender.

5.2.2 Image Data

The data was recorded using the Intel RealSense D435I with a resolution of 1280x720 at 30 fps. The background in the videos is green, except for the videos recorded outside and some of the videos recorded at NWU where the background is cream white. The participants wear dark clothes so that there's contrast between them and the background. In the recordings, the participants are positioned in the middle of the frame and far enough away from the camera that their entire signing space is visible in the frame. Figure 5-5 shows a sample image from the dataset.

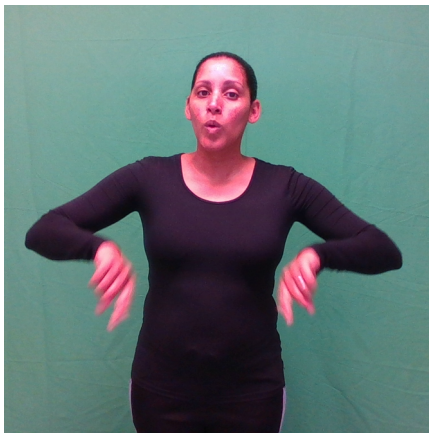


Figure 5-5: Sample image (cropped) from the SASL dataset.

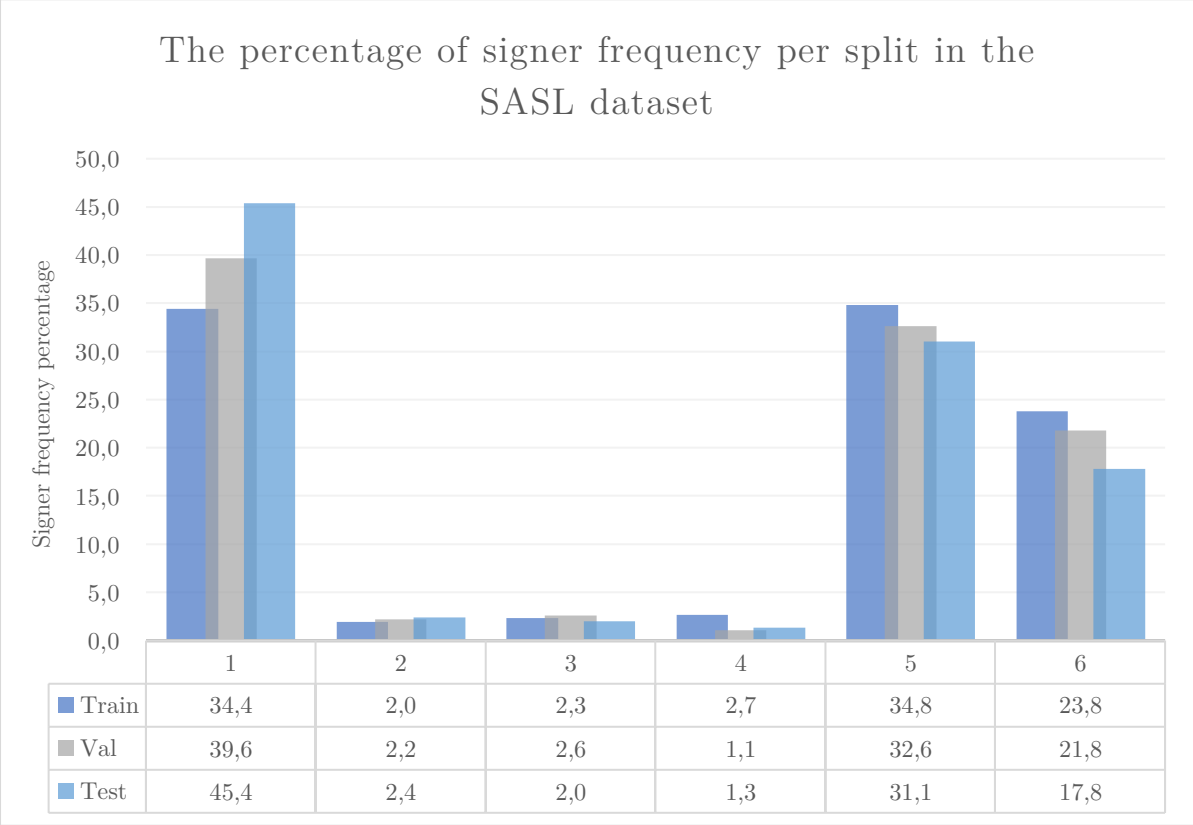
5.2.3 Recording Procedure

During the recording process, the sentence was read out twice for the signer. The first reading was so that they could familiarise themselves with the sentence and the second time was when they were expected to sign while being recorded. If after the first reading, the signer was not comfortable with the sentence, it would be either rephrased, reconstructed, or discarded.

5.2.4 Signer Distribution

Our dataset comprises data from six professional sign language interpreters. Although the dataset consists of data from nine signers, the distribution of the video segments

among the signers is not even. Around 90% of the training set is made up of three signers (the native signers) and the trained interpreters contribute around 2% each to the training set. The distribution is similar in the validation and test sets as well. Figure 5-6 gives a detailed representation of the distribution of signers in the training, validation, and test sets of the SASL dataset.



	1	2	3	4	5	6
Train	34,4	2,0	2,3	2,7	34,8	23,8
Val	39,6	2,2	2,6	1,1	32,6	21,8
Test	45,4	2,4	2,0	1,3	31,1	17,8

Figure 5-6: Distribution of signers in the train, validation, and test sets for the SASL dataset.

5.2.5 Frame Distribution

Similar to the RWTH-PHOENIX-Weather 2014T dataset, our dataset is available as a set of frames for each video segment. The number of frames per segment ranges from 19 to 413 but the average number of frames per segment is 115. At 30 fps, this equates to an average of 3.83 seconds per segment. Our video clips are generally shorter than the clips in the other two datasets. Figure 5-7 shows the distribution of frames per segment in the training set of the SASL dataset.

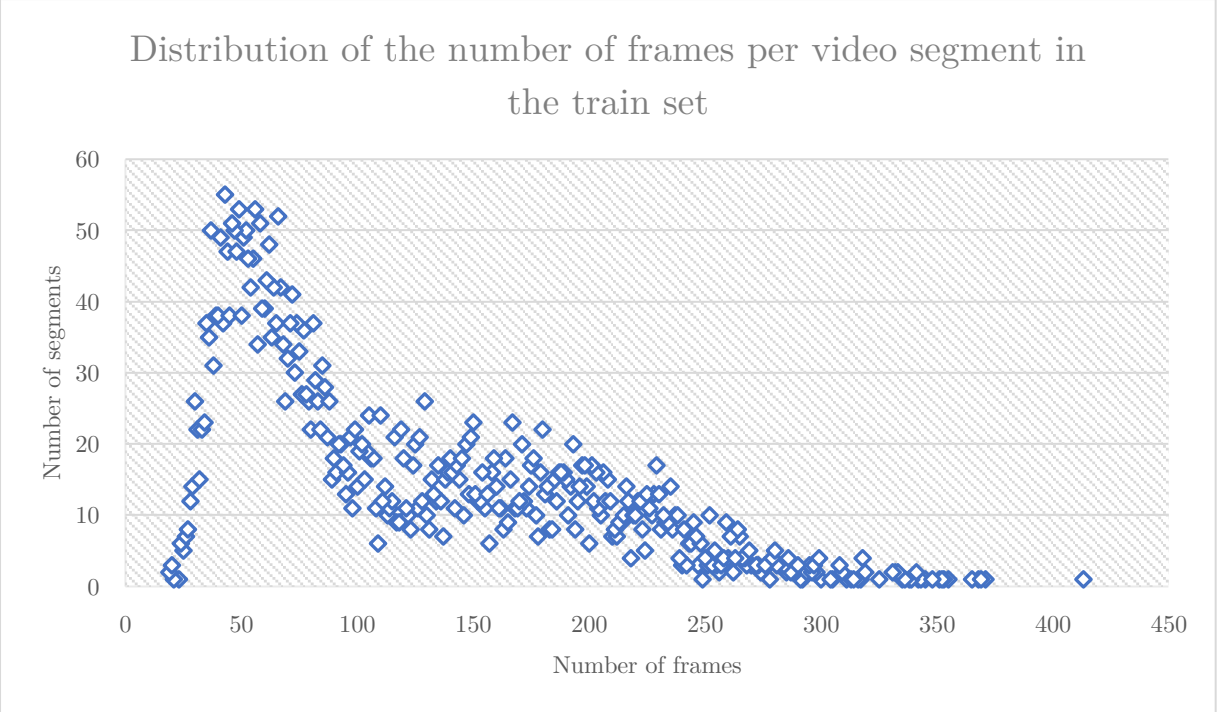


Figure 5-7: The distribution of the number of frames per segment in the SASL train set.

5.2.6 Annotation

The only annotation we did on the data was marking the start and end frames for each video clip. We built a Windows-based Flutter application to speed up the annotation of the data. Figure 5-8 shows the interface of the annotation tool. For every video clip, we selected the start and end frames as the frames where the signer started signing and the frames where the signer returned to their neutral position respectively. Only the frames between these marked frames were used in all our experiments.

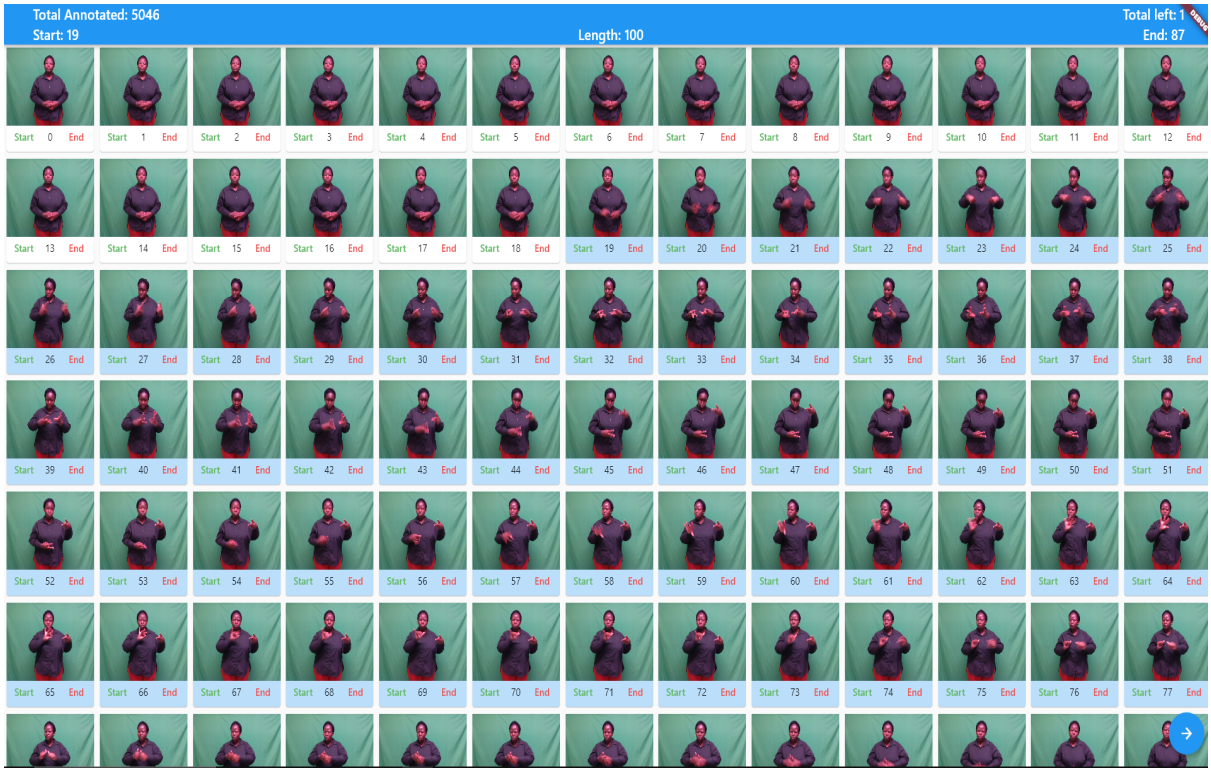


Figure 5-8: The annotation tool used to mark start and end frames for the SASL dataset.

5.2.7 Limitations of the SASL Dataset

Compared to the other datasets, our dataset is quite small and lacking in diversity. It is biased in terms of gender and race and is limited in domain. The dataset also does not contain an equal distribution of sentences between participants. Furthermore, the conditions within which the data was collected have implications for the use of any model developed with this dataset. For instance, the model would have to be tested and/or employed in situations where the background is green, and uniform and participants are wearing contrasting clothes. All of these factors render the dataset unsuitable for practical use due to the risk of creating a distribution shift. For our preliminary research, we assume no distribution shift in the data because the training, validation and testing sets were recorded in the same conditions and contain similar distributions of the participants (shown in Figure 5-6).

We left the extension of the dataset as future work as the current dataset was sufficient for initial experimentation on SASL.

Chapter 6: Proposed Method

Since deep learning gained its popularity, there have been many architectures proposed for solving different problems in different fields. Natural Language Processing and image processing were no exceptions to this. Over the years, deeper and wider models have been developed that continue to outperform the state of the art in language modelling, translation, image captioning, speech recognition, and many more applications. Although state-of-the-art architectures are improving performance in many tasks, the models tend to be very large and require large amounts of data and computation power to train effectively. This is very disadvantageous for under-resourced applications. Transfer learning, where models trained on large amounts of data are finetuned on smaller datasets, is effective when working with smaller datasets.

In this chapter and the following chapters, we answered our remaining two research questions:

- Which types of computer vision and Neural Machine Translation (NMT) architectures are most effective for translating SASL to English in terms of translation accuracy?
- What are some of the adaptations that can be made to the Neural Sign Language Translation (NSLT) model and dataset annotation to improve the quality of the translation of SASL to English?

Our proposed method uses some of the popular architectures in Sign Language Translation and video processing. We also made use of transfer learning in our experiments because of the size of our dataset. This chapter details the architectures proposed for our South African Sign Language translation model.

6.1 Model Pipeline

The model follows the standard Neural Sign Language Translation pipeline which includes image pre-processing, visual feature extraction, translation, and sometimes sign recognition. Since our dataset does not have gloss annotation, our pipeline does not include sign recognition.

6.1.1 Pre-Processing

We extracted keypoints from the frames using mediapose holistic from the MediaPipe framework [87] and used these keypoints to crop the frames to reduce the background information while ensuring that we do not crop out any part of the signer. Frames from the same sequence were cropped using the same dimensions to ensure that the aspect ratio remained the same between frames of the same sequence. Following the cropping, all frames were resized to 200x200.

6.1.2 Visual Feature Extractor

The visual feature extraction step of the model is sometimes called sign embedding, inspired by word embeddings from text-based translation. There are two common sign embedding methods in Neural Sign Language Translation. The first one uses CNNs to convert the visual information in the videos to feature vectors. This method processes all the visual information available in the frames such as the background and the colour of the clothes the participant is wearing. The advantage of this method is that the model has access to all visual information involved in sign language such as facial expressions, handshape, and palm orientation. The disadvantage is that a large variation in the appearances of the participants may hinder the model's performance.

The second method uses skeletal keypoints obtained from passing the frames through a keypoint estimator. This method only provides skeletal information to the model.

This means that the noisy backgrounds and large variations in the participants' appearances do not have a significant impact on the performance of the model. The limitation, however, is that the model cannot use facial expressions and fails to distinguish between some of the hand shapes and poses.

Since our dataset has a uniform background and has contrast between the participants and the background, it is reasonable to experiment with both these methods. In our experiments, we investigated the use of both methods to determine which one works best for SASL.

For the CNNs, we experimented with both 2D-CNNs and 3D-CNNS. As our 2D-CNN, we used the ResNet50 [88] model pre-trained on the ImageNet dataset. For our 3D-CNN, we finetuned an I3D network and an S3D network [82] (both pre-trained on the Kinetics human action recognition dataset) on WLASL2000 (an isolated SLR dataset) and used the best-performing architecture for our experiments. We followed the same experimental setup as specified in the WLASL2000 paper. As shown in Table 6-1, the S3D model significantly outperformed the I3D. We used the S3D in all our experiments.

Table 6-1: I3D vs. S3D performance on WLASL2000 dataset.

Model	**Top 1 Accuracy**
I3D	33.3 %
S3D	40.8 %

For pose estimation, we used normalized keypoints extracted using MediaPipe as features. We used mediapose holistic which extracts 33 pose landmarks, 468 face landmarks, and 21 hand landmarks per hand. Figure 6-1 shows plotted keypoints extracted with mediapose holistic from a sample image from the SASL dataset. From

the image, we can see that the model is fairly accurate at identifying the hands, face and general pose of the signer.

Figure 6-1: Mediapose holistic keypoints on a sample from the SASL dataset.

6.1.3 Translation Model

The translation model is a sequence-to-sequence architecture that uses the features from the visual feature extractor and ground truth translations to learn the mapping between the two languages. We looked at two translation architectures, the recurrent sequence-to-sequence architecture and the transformer architecture. In our recurrent sequence-to-sequence models, we experimented with using either GRUs or LSTMs.

Figure 6-2 shows a graphical representation of our model pipeline where f represents input frames from a sign language video and t represents the target translated text.

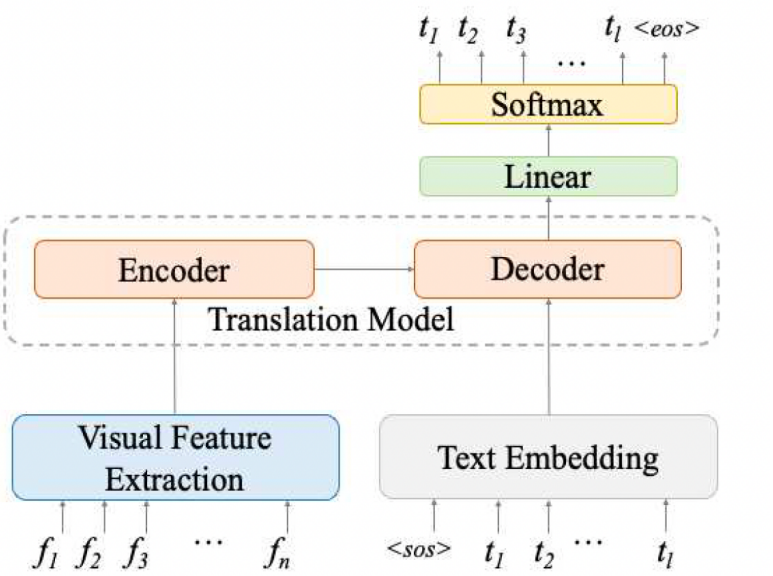


Figure 6-2: Graphical representation of our model pipeline.

6.2 Models

With the different combinations of the proposed architectures, we ended up experimenting with the following models:

- 2D-CNN + LSTM
- 2D-CNN + GRU
- 2D-CNN + Transformer
- 3D-CNN + LSTM
- 3D-CNN + GRU
- 3D-CNN + Transformer
- Pose + LSTM
- Pose + GRU
- Pose + Transformer

Chapter 7: Experiments, Results and Discussion

This chapter details the experiments conducted in this research and discusses the results obtained from them.

7.1 Experimental Setup

We made use of the signjoey framework developed in [18] to train our models. We trained all the models with a batch size of 32. We used Adam Optimizer with an initial learning rate of 0.001 that was reduced by a factor of 0.7 if the validation BLEU-4 score did not improve for 10 validation steps. We validated every 500 steps and stopped training if the validation BLEU-4 score had not improved for 10 validation steps since the learning rate was reduced. We trained all the LSTM and GRU-based models with two layers and a hidden size of 512. The transformers were all trained with three layers and four heads per layer, a hidden size of 512 and a feed-forward size of 2048.

We trained our models on an RTX3090 GPU with 24GB DDR6 RAM, running on an Intel i7 11^{th} Generation Ubuntu Linux machine with 64GB RAM. Due to GPU size constraints, the models were not trained end-to-end. The feature extraction process happened offline. This therefore meant that the visual feature extractors did not undergo any further training. Once feature extraction was completed, the translation model was trained with the extracted features as inputs.

7.2 Metrics

The metrics we used for the experiments are BLEU-1 to BLEU-4. The main evaluation metric is BLEU-4.

7.3 Results

Table 7-1 shows the performance of each of the models on the SASL test set. From the table, we see that 3D-CNN + GRU performs the best, but all our models are performing badly.

Table 7-1: Performance of the models on the SASL validation and test sets.

Model	**Validation**				**Test**			
	BLEU-1	BLEU-2	BLEU-3	BLEU-4	BLEU-1	BLEU-2	BLEU-3	BLEU-4
2D-CNN + LSTM	**14.24**	4.76	1.41	0.57	**15.78**	**5.53**	**1.55**	0.00
2D-CNN + GRU	12.24	3.34	1.11	0.47	8.45	2.09	0.00	0.00
2D-CNN + Transformer	10.16	2.76	0.98	0.43	10.42	2.55	0.73	0.00
3D-CNN + LSTM	12.97	4.42	1.32	0.53	13.19	3.84	0.76	0.00
3D-CNN + GRU	13.84	**5.06**	**1.92**	**0.84**	13.82	4.07	1.26	**0.51**
3D-CNN + Transformer	12.57	3.69	1.28	0.53	12.17	3.29	0.99	0.43
Pose + LSTM	10.44	3.08	1.19	0.55	10.13	3.21	0.21	0.00
Pose + GRU	10.13	3.31	0.15	0.63	10.02	2.54	0.12	0.00
Pose + Transformer	11.05	3.51	1.18	0.60	10.31	2.87	0.00	0.00

7.4 Additional Experiments

From Table 7-1, we see that all the models perform poorly. We investigated the possible causes by conducting additional experiments and comparing our best model's performance to the performance on other datasets.

7.4.1 Training on RWTH-PHOENIX-Weather 2014T and How2Sign:

We trained and evaluated the best-performing model (3D-CNN + GRU) on the RWTH-PHOENIX-Weather 2014T and the How2Sign dataset with the same hyperparameters. We did not use the gloss annotations in the RWTH-PHOENIX-Weather 2014T and on the How2Sign dataset, we only used the front RGB frames for our experiment. In Table 7-2, we see that the model performs significantly better on the RWTH-PHOENIX-Weather 2014T dataset but still performs poorly on the How2Sign dataset. This may be due to the smaller domain of the PHOENIX-Weather 2014T compared to the other two.

Table 7-2: Performance of 3D-CNN + GRU on RWTH-PHOENIX-Weather 2014T and How2Sign test sets.

Dataset	**BLEU-4**
RWTH-PHOENIX-Weather 2014T	11.73
How2Sign	1.73

7.4.2 Text Translation

After our initial results, we suspected that the dataset may be too small for the model to learn from. To validate this, we trained a text-based translation model using the recorded sentences and their Setswana counterparts from the Autshumato dataset. We trained the model with a GRU-based recurrent sequence-to-sequence translation model with the same hyperparameters as before. Table 7-3 shows that spoken language translation produces significantly better results than SLT, indicating that the translation model can learn from our dataset and suggesting that the bottleneck may be within feature extraction.

Table 7-3: Text translation results using recorded sentences and their Setswana counterparts.

Set	**BLEU-4**
Validation	9.53
Test	8.52

7.4.3 Finetuning 3D-CNN

Since we have access to gloss annotations on RWTH-PHOENIX-Weather 2014T, we finetuned the S3D network on gloss recognition and used it in the 3D-CNN + GRU model as an attempt to improve our feature extraction. In our finetuning experiment, we obtained a word error rate (WER) of 37.8% on the SLR task. Table 7-4 shows that finetuning the 3D-CNN improves the performance of the model on the RWTH-PHOENIX-Weather 2014T dataset. There is a slight improvement on the SASL dataset and the How2Sign dataset performs worse with the finetuned 3D-CNN.

Table 7-4: The performance of the 3D-CNN + GRU model with 3D-CNN pre-trained on SLR task.

Dataset	**Test (BLEU-4)**
RWTH-PHOENIX-Weather 2014T	13.23
How2Sign	1.31
SASL	1.35

7.4.4 Hand Autoencoder

Inspired by Nkwentsha [89] we trained an autoencoder using cropped hand images from the SASL dataset in another attempt to improve our feature extraction. We cropped the left and right hands of all the images in the dataset and randomly selected 100,000 samples to train on and 20,000 each for the test and validation sets. We resized the images to 128x128 and trained our autoencoder for 500 epochs with a batch size

of 318 and a latent dimension of 256. Figure 7-1 shows a detailed description of our autoencoder model and its components.

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
Autoencoder                              [318, 3, 128, 128]        --
├─Encoder: 1-1                           [318, 256]                --
│    └─Sequential: 2-1                   [318, 256]                --
│    │    └─Conv2d: 3-1                  [318, 32, 64, 64]         896
│    │    └─GELU: 3-2                    [318, 32, 64, 64]         --
│    │    └─Conv2d: 3-3                  [318, 32, 64, 64]         9,248
│    │    └─GELU: 3-4                    [318, 32, 64, 64]         --
│    │    └─Conv2d: 3-5                  [318, 64, 32, 32]         18,496
│    │    └─GELU: 3-6                    [318, 64, 32, 32]         --
│    │    └─Conv2d: 3-7                  [318, 64, 32, 32]         36,928
│    │    └─GELU: 3-8                    [318, 64, 32, 32]         --
│    │    └─Conv2d: 3-9                  [318, 64, 16, 16]         36,928
│    │    └─GELU: 3-10                   [318, 64, 16, 16]         --
│    │    └─Flatten: 3-11                [318, 16384]              --
│    │    └─Linear: 3-12                 [318, 256]                4,194,560
├─Decoder: 1-2                           [318, 3, 128, 128]        --
│    └─Sequential: 2-2                   [318, 16384]              --
│    │    └─Linear: 3-13                 [318, 16384]              4,210,688
│    │    └─GELU: 3-14                   [318, 16384]              --
│    └─Sequential: 2-3                   [318, 3, 128, 128]        --
│    │    └─ConvTranspose2d: 3-15        [318, 64, 32, 32]         36,928
│    │    └─GELU: 3-16                   [318, 64, 32, 32]         --
│    │    └─Conv2d: 3-17                 [318, 64, 32, 32]         36,928
│    │    └─GELU: 3-18                   [318, 64, 32, 32]         --
│    │    └─ConvTranspose2d: 3-19        [318, 32, 64, 64]         18,464
│    │    └─GELU: 3-20                   [318, 32, 64, 64]         --
│    │    └─Conv2d: 3-21                 [318, 32, 64, 64]         9,248
│    │    └─GELU: 3-22                   [318, 32, 64, 64]         --
│    │    └─ConvTranspose2d: 3-23        [318, 3, 128, 128]        867
│    │    └─Tanh: 3-24                   [318, 3, 128, 128]        --
==========================================================================================
Total params: 8,610,179
Trainable params: 8,610,179
Non-trainable params: 0
Total mult-adds (G): 101.60
==========================================================================================
Input size (MB): 62.52
Forward/backward pass size (MB): 2209.74
Params size (MB): 34.44
Estimated Total Size (MB): 2306.70
==========================================================================================
```

Figure 7-1: Detailed description of our autoencoder model and its components.

We evaluated the performance of the autoencoder using the reconstruction loss.

Table 7-5 shows the validation and test loss of our trained hand autoencoder and Figure 7-2 shows the autoencoder reconstructions of some sample cropped hand images from the SASL dataset.

Table 7-5: Reconstruction loss of the hand autoencoder on the validation and test sets.

Set	**Reconstruction Loss**
Validation	195
Test	193

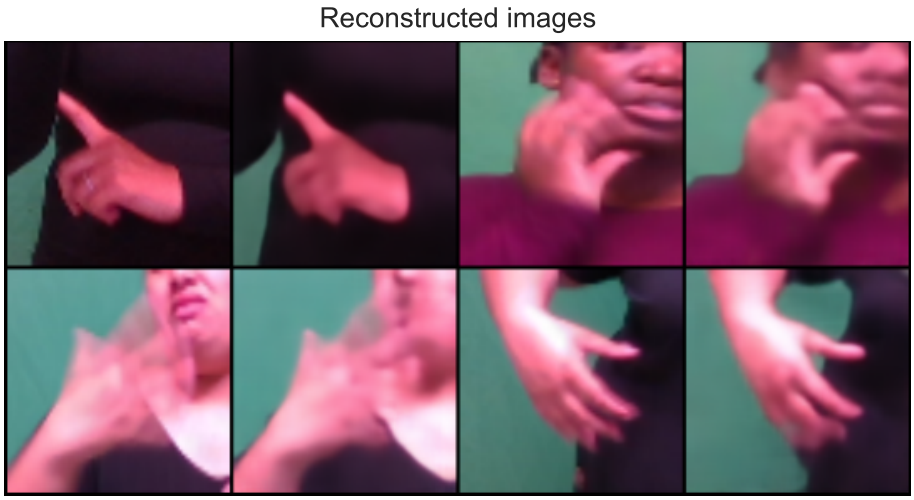


Figure 7-2: Hand image reconstructions by the autoencoder on the SASL dataset. The original images are on the left and the reconstructions are on the right.

In our next experiment, we used the hand autoencoder along with MediaPipe pose as feature extractors. For every image, we extracted the left- and right-hand features with the autoencoder and 33 pose keypoints and concatenated them to form feature vectors. We also experimented with using the hand autoencoder by itself as a feature extractor. We then fed the features into the translation models as we did with previous experiments. Table 7-6 shows the results of our experiments with the hand autoencoder and pose estimator. We see that Hand AE + Pose + GRU performs better than Hand AE + GRU, confirming that pose information is important when processing sign language.

Table 7-6: Performance of the hand autoencoder + pose model on the SASL dataset.

Model	**Validation**				**Test**			
	BLEU-1	BLEU-2	BLEU-3	BLEU-4	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Hand AE + Pose + LSTM	11.76	3.84	1.62	0.87	10.68	2.81	0.77	0.00
Hand AE + Pose + GRU	**12.38**	**4.43**	1.80	0.80	**12.07**	3.49	1.14	**0.57**
Hand AE + Pose + Transformer	11.22	4.29	**1.84**	**1.03**	10.30	3.03	0.82	0.00
Hand AE + LSTM	12.01	3.54	1.42	0.55	11.97	3.49	0.70	0.00
Hand AE + GRU	11.35	3.46	1.36	0.55	11.71	**3.71**	**1.28**	0.52
Hand AE + Transformer	10.10	3.00	1.08	0.45	10.41	2.80	0.87	0.38

7.4.5 Splitting Fingerspelled Words

Due to its domain, the SASL dataset contains a significant number of acronyms and proper nouns. In this last experiment, we investigated the effect of splitting all the fingerspelled words in the dataset into their individual characters. In this way, the model would have a better opportunity to learn fingerspelling and be able to identify it in the data. We built another simple Windows-based flutter application for annotating the text data. In our annotation, we split all words that were fingerspelled in the videos and all numbers greater than 20 into individual characters. Figure 7-3 shows an example of how we annotated the text.

We ran our experiment using Hand AE + Pose + GRU in our experiment because it has a more "zoomed-in" view into the hands than all the other feature extraction

methods that we have explored. Table 7-7 shows how the model performs when we split fingerspelled words into their individual characters. We see that this does indeed improve the performance of the model.

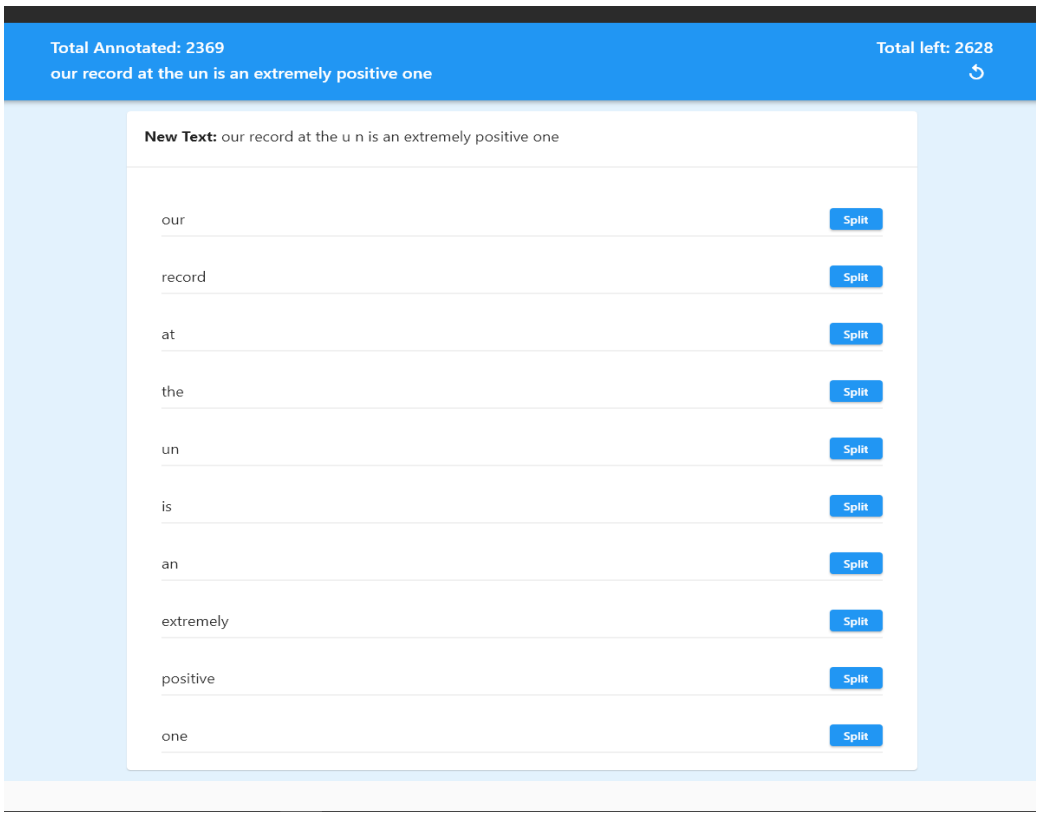


Figure 7-3: The text annotation platform interface. In this example, we split the acronym "un" into "u n" because it is fingerspelled in the corresponding video clip.

Table 7-7: Performance of Hand AE + Pose + GRU on the SASL dataset with split fingerspelled words.

Model	**Validation**				**Test**			
	BLEU-1	BLEU-2	BLEU-3	BLEU-4	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Hand AE + Pose + GRU	12.12	3.82	1.55	0.78	12.99	3.98	1.46	0.80

Chapter 8: Conclusion and Future Work

In this research, we implemented the first exploration of the use of Neural Sign Language Translation on SASL, as a preliminary study towards bridging the communication gap between the Deaf community and the hearing community in South Africa. As stated earlier, our objectives were to:

- Design and develop a parallel SASL and English dataset suitable for use in Neural Sign Language Translation.
- Consider existing computer vision and NMT architectures and develop a suitable machine learning model for the translation of SASL to English.
- Evaluate the model and investigate ways to improve its performance.

We met our first objective by collecting the first SASL dataset for Neural Sign Language Translation and met the last two objectives by conducting extensive experiments to find the best model given the size of the dataset and limited computational resources.

To meet our remaining two objectives, we used the standard Neural Sign Language Translation pipeline which comprises a feature extraction model followed by a translation model. In our experiments, we explored the use of three common feature extraction methods and two translation models. When the performance of the model suggested that better feature extraction methods are required for the SASL dataset, we experimented with transfer learning methods and self-supervised training. Our best model obtained a BLEU-4 score of 1.35 when the feature extractor was finetuned on a Sign Language Recognition task, indicating the effectiveness of transfer learning and

adding gloss-level supervision in Sign Language Translation tasks. We also investigated the impact of annotating fingerspelled words by splitting them into individual characters. To the best of our knowledge, this was the first investigation into this type of annotation for Neural Sign Language Translation.

Our models perform badly with our dataset and the How2Sign dataset but perform better on the RWTH-PHOENIX-Weather 2014T dataset. Although transfer learning resulted in some improvement, a BLEU-4 score of 1.35 is still significantly low.

Our research was the first exploration of the use of deep learning methods in the automatic translation of South African Sign Language to English. The dataset collected in this research is easy to extend for further studies and future works. The exploration of the annotation of fingerspelling, and the use of a pretrained autoencoder as a handshape feature extractor will provide insight for future work on Sign Language Translation.

8.1 Future Work and Recommendations

Our work was limited by time and computing resources and our model performance was poor and far from practical. We hereby make the following recommendations for future work:

- We recommend a deeper investigation into SASL and How2Sign datasets to determine the cause of the poor performance.
- We recommend adding gloss annotation to the developed SASL dataset as our experiments have shown that intermediate supervision has a positive impact on the performance of our translation models.

- We also recommend an extension of the SASL dataset with the collection of more sign language data in different domains with datasets designed in collaboration with members of the Deaf community.
- Communication is a two-way process. We recommend that research on the use of deep learning in the translation from English to SASL also be conducted as this would be the next step in the mission to bridge the language barrier.

Bibliography

[1] SANDA, "Welcome," South African National Deaf Association Website. Accessed: Jul. 06, 2022. [Online]. Available: https://www.sanda.org.za

[2] T. Ferreira, "South African TV slammed over its ongoing treatment of the deaf," *Channel24*, Feb. 23, 2018. [Online]. Available: https://www.news24.com/Channel/south-african-tv-slammed-over-its-ongoing-treatment-of-the-deaf-20180222

[3] K. S. Jones, "Natural Language Processing: A Historical Review," in *Current Issues in Computational Linguistics: In Honour of Don Walker*, A. Zampolli, N. Calzolari, and M. Palmer, Eds., Dordrecht: Springer Netherlands, 1994, pp. 3–16. doi: 10.1007/978-0-585-35958-8_1.

[4] H. Schwenk, G. Wenzek, S. Edunov, E. Grave, and A. Joulin, "CCMatrix: Mining Billions of High-Quality Parallel Sentences on the WEB," Nov. 2019, [Online]. Available: http://arxiv.org/abs/1911.04944

[5] A. Duarte *et al.*, "How2Sign: A Large-scale Multimodal Dataset for Continuous American Sign Language," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2735–2744, 2021, [Online]. Available: http://how2sign.github.io/

[6] N. C. Camgoz, S. Hadfield, O. Koller, H. Ney, and R. Bowden, "Neural Sign Language Translation," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 7784–7793, 2018, doi: 10.1109/CVPR.2018.00812.

[7] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded Up Robust Features," in *Computer Vision -- ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417.

[8] T. Surasak, I. Takahiro, C. H. Cheng, C. E. Wang, and P. Y. Sheng, "Histogram of oriented gradients for human detection in video," *Proceedings of 2018 5th International Conference on Business and Industrial Research: Smart Technology for Next Generation of Information, Engineering, Business and Social Science, ICBIR 2018*, pp. 172–176, 2018, doi: 10.1109/ICBIR.2018.8391187.

[9] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Leaming*, vol. 20, pp. 273–297, 1995, doi: 10.1007/BF00994018.

[10] T. M. Cover and P. E. Hart, "Nearest neighbour pattern classification," *IEEE Trans. Inf. Theory*, vol. 13, pp. 21–27, 1967.

[11] C. M. Jin, Z. Omar, and M. H. Jaward, "A mobile application of American sign language translation via image processing algorithms," *Proceedings - 2016 IEEE Region 10 Symposium, TENSYMP 2016*, pp. 104–109, 2016, doi: 10.1109/TENCONSpring.2016.7519386.

[12] J. MacQueen, “Some methods for classification and analysis of multivariate observations,” *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 5.1, pp. 281–297, 1967.

[13] H. Sakoe and S. Chiba, “Dynamic Programming Algorithm Optimization for Spoken Word Recognition,” *IEEE Trans Acoust*, vol. 26, pp. 43–49, 1978.

[14] P. Jangyodsuk, C. Conly, and V. Athitsos, “Sign language recognition using Dynamic Time Warping and hand shape distance based on Histogram of Oriented Gradient features,” *ACM International Conference Proceeding Series*, vol. 2014-May, pp. 3–8, 2014, doi: 10.1145/2674396.2674421.

[15] S. S. Kumar, T. Wangyal, V. Saboo, and R. Srinath, “Time Series Neural Networks for Real-Time Sign Language Translation,” *Proceedings - 17th IEEE International Conference on Machine Learning and Applications, ICMLA 2018*, pp. 243–248, 2019, doi: 10.1109/ICMLA.2018.00043.

[16] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “BLEU: a Method for Automatic Evaluation of Machine Translation,” in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, Philadelphia, 2002, pp. 311–318. doi: 10.1002/andp.19223712302.

[17] D. Guo, W. Zhou, H. Li, and M. Wang, “Hierarchical LSTM for sign language translation,” *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pp. 6845–6852, 2018.

[18] N. C. Camgoz, O. Koller, S. Hadfield, and R. Bowden, “Sign Language Transformers: Joint End-to-end Sign Language Recognition and Translation,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 10020–10030, 2020, [Online]. Available: http://arxiv.org/abs/2003.13830

[19] L. Van Zijl, “South African sign language machine translation project,” *Eighth International ACM SIGACCESS Conference on Computers and Accessibility, ASSETS 2006*, vol. 2006, no. July, pp. 233–234, 2006, doi: 10.1145/1168987.1169031.

[20] B. McInnes, “South African Sign Language Dataset Development and Translation: A Glove-based Approach,” 2014, [Online]. Available: https://open.uct.ac.za/bitstream/item/8977/thesis_sci_2014_kaaya_lt.pdf?sequence=1

[21] M. Seymour and M. Tšoeu, “A mobile application for South African Sign Language (SASL) recognition,” in *AFRICON 2015*, IEEE, 2015, pp. 1–5.

[22] I. Achmed and J. Connan, “Upper body pose estimation towards the translation of South African sign language,” in *Proceedings of the Southern Africa Telecommunication Networks and Applications Conference*, 2010, pp. 427–432.

[23] P. Akach, “The grammar of sign language,” *Language Matters*, vol. 28, no. 1, pp. 7–35, 1997, doi: 10.1080/10228199708566118.

[24] J. E. Wehrmeyer, "A Critical Investigation Of Deaf Comprehension Of Signed TV News Interpretation," 2013.

[25] L. Van Zijl and A. Combrink, "The South African sign language machine translation project: Issues on non-manual sign generation," *ACM International Conference Proceeding Series*, vol. 204, pp. 127–134, 2006, doi: 10.1145/1216262.1216276.

[26] S. C. Jael, "Looking Back: Dominican Education in SA," *The Southern Cross*, Jul. 10, 2016.

[27] C. Storbeck *et al.*, "South African Deaf Education and the Deaf Community," *Am Ann Deaf*, vol. 155, pp. 488–518, Sep. 2010, doi: 10.1353/aad.2010.0034.

[28] W. C. Stokoe and M. Marschark, "Sign language structure: An outline of the visual communication systems of the American deaf," *J Deaf Stud Deaf Educ*, vol. 10, no. 1, pp. 3–37, 2005, doi: 10.1093/deafed/eni001.

[29] H. G. Morgans, "Where did South African sign language originate?," *Language Matters*, vol. 30, no. 1, pp. 53–58, 1999, doi: 10.1080/10228199908566144.

[30] "South African Schools Act, Act 84." 1996.

[31] SANews, "Sign Language recognised as a home language," *SANews*, Mar. 04, 2018.

[32] C. Penn and T. Reagan, "The Properties Of South African Sign Language: Lexical Diversity & Syntactic Unity," *Sign Lang Stud*, vol. 1085, no. 1, pp. 319–327, 1994, doi: 10.1353/sls.1994.0011.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun ACM*, vol. 60, no. 6, pp. 84–90, 2017, doi: 10.1145/3065386.

[34] M. Narang, "Activation Functions: With Real-life Analogy and Python Code," Shiksha Online.

[35] Z. Rguibi *et al.*, "CXAI: Explaining Convolutional Neural Networks for Medical Imaging Diagnostic," Jun. 2022, doi: 10.3390/electronics11111775.

[36] J. L. Elman, "Finding Structure in Time," *Cogn Sci*, vol. 14, no. 2, pp. 179–211, 1990, doi: https://doi.org/10.1207/s15516709cog1402_1.

[37] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.

[38] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to Forget: Continual Prediction with LSTM," *Neural Comput*, vol. 12, no. 10, pp. 2451–2471, 2000, doi: 10.1162/089976600300015015.

[39] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," pp. 1–9, 2014, [Online]. Available: http://arxiv.org/abs/1412.3555

[40] P. F. Brown *et al.*, "A statistical approach to machine translation," *Computational Linguistics*, vol. 16, no. 2, pp. 79–85, 1990, doi: 10.3115/991365.991407.

[41] K. Cho *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pp. 1724–1734, 2014, doi: 10.3115/v1/d14-1179.

[42] D. Bahdanau, K. H. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–15, 2015.

[43] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," *34th International Conference on Machine Learning, ICML 2017*, vol. 3, pp. 2029–2042, 2017.

[44] A. Vaswani *et al.*, "Attention is all you need," *Adv Neural Inf Process Syst*, vol. 2017-Decem, pp. 5999–6009, 2017.

[45] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches," pp. 103–111, 2015, doi: 10.3115/v1/w14-4012.

[46] B. Dorr, M. Snover, and N. Madnami, "Part 5: Machine Translation Evaluation," *Handbook of Natural Language Processing and Machine Translation*, no. 1, pp. 801–894, 2010.

[47] C.-Y. Lin, "ROUGE: A Package for Automatic Evaluation of Summaries," in *proceedings of the workshop on text summarization branches out*, Barcelona, 2004. doi: 10.1253/jcj.34.1213.

[48] A. Lavie and A. Agarwal, "METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments," *Proceedings of the Second Workshop on Statistical Machine Translation*, no. June, pp. 228–231, 2007.

[49] O. Koller, S. Zargaran, H. Ney, and R. Bowden, "Deep sign: Hybrid CNN-HMM for continuous sign language recognition," *British Machine Vision Conference 2016, BMVC 2016*, vol. 2016-Septe, pp. 136.1-136.12, 2016, doi: 10.5244/C.30.136.

[50] F. Ertam, "Data classification with deep learning using tensorflow," *2nd International Conference on Computer Science and Engineering, UBMK 2017*, pp. 755–758, 2017, doi: 10.1109/UBMK.2017.8093521.

[51] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016*, 2016.

[52] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," *ArXiv*, no. NeurIPS, 2019.

[53] Y. Jia *et al.*, "Caffe: Convolutional Architecture for Fast Feature Embedding∗," pp. 675–678, 2014, doi: 10.1145/2647868.2654889.

[54] K. Yin, "Sign Language Translation with Transformers." 2020.

[55] M. J. Cheok, Z. Omar, and M. H. Jaward, "A review of hand gesture and sign language recognition techniques," *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 1, pp. 131–153, 2019, doi: 10.1007/s13042-017-0705-5.

[56] P. Escudeiro *et al.*, "Virtual Sign - A Real Time Bidirectional Translator of Portuguese Sign Language," *Procedia Comput Sci*, vol. 67, no. Dsai, pp. 252–262, 2015, doi: 10.1016/j.procs.2015.09.269.

[57] N. Adaloglou *et al.*, "A Comprehensive Study on Sign Language Recognition Methods," *arXiv*. pp. 1–12, 2020.

[58] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber, "Connectionist Temporal Classification : Labelling Unsegmented Sequences with Recurrent Neural Networks," in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, 2006, pp. 369–376.

[59] D. Li, C. R. Opazo, X. Yu, and H. Li, "Word-level Deep Sign Language Recognition from Video : A New Large-scale Dataset and Methods Comparison," *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 1459–1469, 2020.

[60] H. R. V. Joze and O. Koller, "MS-ASL: A large-scale data set and benchmark for understanding American sign language," *30th British Machine Vision Conference 2019, BMVC 2019*, 2020.

[61] J. Huang, W. Zhou, Q. Zhang, H. Li, and W. Li, "Video-based sign language recognition without temporal segmentation," *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pp. 2257–2264, 2018.

[62] J. Forster, C. Schmidt, O. Koller, M. Bellgardt, and H. Ney, "Extensions of the sign language recognition and translation corpus RWTH-PHOENIX-Weather," *Proceedings of the 9th International Conference on Language Resources and Evaluation, LREC 2014*, pp. 1911–1916, 2014.

[63] U. Von Agris, M. Knorr, and K. F. Kraiss, "The significance of facial features for automatic sign language recognition," *2008 8th IEEE International Conference on*

Automatic Face and Gesture Recognition, FG 2008, 2008, doi: 10.1109/AFGR.2008.4813472.

[64] O. Koller, J. Forster, and H. Ney, "Continuous sign language recognition: Towards large vocabulary statistical recognition systems handling multiple signers," *Computer Vision and Image Understanding*, vol. 141, pp. 108–125, 2015, doi: 10.1016/j.cviu.2015.09.013.

[65] O. Koller, H. Ney, and R. Bowden, "Deep Hand: How to Train a CNN on 1 Million Hand Images When Your Data is Continuous and Weakly Labelled," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 3793–3802, 2016, doi: 10.1109/CVPR.2016.412.

[66] O. Koller, S. Zargaran, and H. Ney, "Re-sign: Re-aligned end-to-end sequence modelling with deep recurrent CNN-HMMs," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 3416–3424, 2017, doi: 10.1109/CVPR.2017.364.

[67] N. C. Camgoz, S. Hadfield, O. Koller, and R. Bowden, "SubUNets: End-to-End Hand Shape and Continuous Sign Language Recognition," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3075–3084. doi: 10.1109/ICCV.2017.332.

[68] R. Cui, H. Liu, and C. Zhang, "Recurrent convolutional neural networks for continuous sign language recognition by staged optimization," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 1610–1618, 2017, doi: 10.1109/CVPR.2017.175.

[69] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–14, 2015, [Online]. Available: https://arxiv.org/pdf/1409.1556

[70] J. Pu, W. Zhou, and H. Li, "Iterative alignment network for continuous sign language recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 4160–4169, 2019, doi: 10.1109/CVPR.2019.00429.

[71] H. Zhou, W. Zhou, W. Qi, J. Pu, and H. Li, "Improving Sign Language Translation with Monolingual Data by Sign Back-Translation," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1316–1325, 2021, doi: 10.1109/CVPR46437.2021.00137.

[72] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2015 Inter, pp. 4489–4497, 2015, doi: 10.1109/ICCV.2015.510.

[73] D. Guo, W. Zhou, A. Li, H. Li, and M. Wang, "Hierarchical Recurrent Deep Fusion Using Adaptive Clip Summarization for Sign Language Translation," *IEEE Transactions on Image Processing*, vol. 29, pp. 1575–1590, 2020, doi: 10.1109/TIP.2019.2941267.

[74] S. K. Ko, C. J. Kim, H. Jung, and C. Cho, "Neural sign language translation based on human keypoint estimation," *Applied Sciences (Switzerland)*, vol. 9, no. 13, pp. 1–19, 2019, doi: 10.3390/app9132683.

[75] M. T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *Conference Proceedings - EMNLP 2015: Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421, 2015, doi: 10.18653/v1/d15-1166.

[76] O. Koller, N. C. Camgoz, H. Ney, and R. Bowden, "Weakly Supervised Learning with Multi-Stream CNN-LSTM-HMMs to Discover Sequential Parallelism in Sign Language Videos," *IEEE Trans Pattern Anal Mach Intell*, vol. 42, no. 9, pp. 2306–2320, 2020, doi: 10.1109/TPAMI.2019.2911077.

[77] P. Cabot´alvarez, C. Cabot´alvarez, X. G. Nieto, and L. Tarrés Benet, "Sign Language Translation based on Transformers for the How2Sign Dataset".

[78] D. Li *et al.*, "TSPNet: Hierarchical feature learning via temporal semantic pyramid for sign language translation," *Adv Neural Inf Process Syst*, vol. 2020-Decem, no. NeurIPS, pp. 1–12, 2020.

[79] J. Carreira and A. Zisserman, "Quo Vadis, action recognition? A new model and the kinetics dataset," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 4724–4733, 2017, doi: 10.1109/CVPR.2017.502.

[80] M. B. Narayanan, K. M. Bharadwaj, G. R. Nithin, D. R. R. Padamnoor, and V. Vijayaraghavan, *Sign Language Translation Using Multi Context Transformer*, vol. 13068 LNAI. Springer International Publishing, 2021. doi: 10.1007/978-3-030-89820-5_25.

[81] S. Gan, Y. Yin, Z. Jiang, L. Xie, and S. Lu, "Skeleton-Aware Neural Sign Language Translation," *MM 2021 - Proceedings of the 29th ACM International Conference on Multimedia*, pp. 4353–4361, 2021, doi: 10.1145/3474085.3475577.

[82] Z. Qui, T. Yao, and T. Mei, "Learning Spatio-Temporal Representation With Pseudo-3D Residual Networks," *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 5533–5541, 2017, doi: 10.23919/IConAC.2019.8894962.

[83] Y. Chen, F. Wei, X. Sun, Z. Wu, and S. Lin, "A Simple Multi-Modality Transfer Learning Baseline for Sign Language Translation," 2022, [Online]. Available: http://arxiv.org/abs/2203.04287

[84] T. Ananthanarayana *et al.*, "Deep Learning Methods for Sign Language Translation," *ACM Trans Access Comput*, vol. 14, no. 4, 2021, doi: 10.1145/3477498.

[85] L. Van Zijl, "South african sign language machine translation project," *Eighth International ACM SIGACCESS Conference on Computers and Accessibility, ASSETS 2006*, vol. 2006, pp. 233–234, 2006, doi: 10.1145/1168987.1169031.

[86] C. McKellar, "Autshumato English-Setswana Parallel Corpora." 2016. [Online]. Available: https://hdl.handle.net/20.500.12185/404

[87] C. Lugaresi *et al.*, "MediaPipe: A Framework for Building Perception Pipelines," Jun. 2019, [Online]. Available: http://arxiv.org/abs/1906.08172

[88] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.

[89] X. Nkwentsha, "Semi-Supervised Transfer Learning for medical images as an alternative to ImageNet Transfer Learning," University of Cape Town, 2022. [Online]. Available: http://hdl.handle.net/11427/37687