Joint Pose Based Sign Language Translation Using Graph Convolutional Network

by

Md. Safirur Rashid

Bhuiyan Sanjid Shafique

Tauhid Mostahid

Supervisor

Dr. Md. Hasanul Kabir

Professor, Department of CSE

Co-Supervisor

Md. Bakhtiar Hasan

Lecturer, Department of CSE

BACHELOR OF SCIENCE

IN

COMPUTER SCIENCE AND ENGINEERING

Department of Computer Science and Engineering

Islamic University of Technology (IUT)

Board Bazar, Gazipur-1704, Bangladesh.

April, 2022.

Declaration of Authorship

This is to certify that the work presented in this thesis is the outcome of the analysis and simulations carried out by **Md. Safirur Rashid**, **Bhuiyan Sanjid Shafique**, and **Tauhid Mostahid** under the supervision of **Dr. Md. Hasanul Kabir** and co-supervision of **Md. Bakhtiar Hasan**, Department of Computer Science and Engineering (CSE), Islamic University of Technology (IUT), Dhaka, Bangladesh. It is also declared that neither of this thesis nor any part of this thesis has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Supervisor

Dr. Md. Hasanul Kabir
Professor,
Dept. of Computer Science and Engineering,
Islamic University of Technology (IUT)

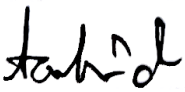
Co-Supervisor

Md. Bakhtiar Hasan
Lecturer,
Dept. of Computer Science and Engineering,
Islamic University of Technology (IUT)

Authors

Md. Safirur Rashid
Student No: 170041020
Academic Year: 2020-21

Bhuiyan Sanjid Shafique
Student No: 170041029
Academic Year: 2020-21



Tauhid Mostahid
Student No: 170041059
Academic Year: 2020-21

Table of Contents

List of Figures

List of Tables

Acknowledgment

All praise belongs to Allah, the Almighty and may peace and blessings be upon Prophet Muhammad(Sm). It would not have been possible to complete this thesis without the decree of Allah. Many people contributed to this thesis through their encouragement, support, and advice. Our heartfelt gratitude to them for that.

We are very grateful to our supervisor, **Dr. Md. Hasanul Kabir** and co-supervisor, **Md. Bakhtiar Hasan**, Department of Computer Science and Engineering, Islamic University of Technology (IUT), for their supervision, knowledge and support, which have been invaluable for us.

Finally, we offer our heartfelt thanks to our parents for their love and support.

Abstract

Sign Language Translation (SLT) is defined as the task of generating meaningful spoken sentence from a sequence of signs. Prior work on Sign Language Translation (SLT) focuses on generating sequence of spoken words using different RGB based and some skeletal based architectures. Extracting robust spatiotemporal features is very important in generating meaningful sign translation. Feature extraction based on skeletal methods has gained attraction in recent times, as they are able to capture complex human dynamics, ignore noisy data efficiently and extract more distinct spatiotemporal attributes. In this work, we explore the feasibility of a skeletal-based feature extraction system in the domain of Sign Language Translation (SLT) in order to generate more distinctive features. To accomplish that, we improve the estimation accuracy of existing particular pose estimation models, which are a key component of any skeletal-based feature extraction system. We also explore a sign video segment representation to enhance detection accuracy of signs within a video and generate more accurate translation. We assess the performance of our proposed pipeline on the *PHOENIX14T* dataset, which is the benchmark dataset in this field. Although our model performs better than some prior works, it fails to achieve state-of-the-art results. We also share some performance analysis regarding sign-segmentation size and the number of keypoints taken into consideration.

Chapter 1

Introduction

Sign language is a dynamic system of communication used by deaf or mute people around the world as their primary medium of transmission. It consists of intricate hand movements, upper-body postures, head shakes or nods, and even gaze. It is a challenging skill to master for the general public, as we are accustomed to auditory mode of communication, while sign languages operate on visual mode of communication. It is performed by physical movements and perceived by vision or eyes. On top of that, sign language is different from spoken language in the fact that it has its own grammatical rules and independent words. [5] This creates a notable barrier between signers and non-signers, although deaf people comprise a significant part of our societies. Hence, it is really important to bridge this gap, although this comes with significant challenges.

A sign gesture consists of mainly two kinds of attributes - manual attributes and non-manual attributes. The manual attributes are based on physical movement, shape, and coordination of the human body, while the non-manual attributes mainly depend on facial shape, nodding, and position of the shoulder. [6] Even though manual attributes are the key components in sign language, information is not transmitted in full by using these features only. Non-manual attributes play a part in this regard by providing more precision. Combination of these features give the meaning of a sign, provide necessary context and helps to identify between objects.

Sign languages are multi-dynamic. Sign gestures can be static or dynamic. In the case of static gestures, there is no presence of hand movements. And in the case

of dynamic gestures, hand movements are crucial to convey information properly. The execution speed of a sign also play role in determining the meaning of as sign. All these intricate components make sign language a challenging skill to execute, and makes the development of efficient sign analyzer tools more important. However, similar to spoken languages, where thousands of words can be produced using some finite amount of root words, the same thing can be noticed in sign languages also. A finite number of physical movements or gestures can produce a large number of signs. This makes analysis, recognition, and generation of sign languages more feasible and easier.

There have been considerable developments in the field of weakly supervised learning [7] in current times. In addition to that, innovations in the field of human body pose estimation [8] have made it possible to work on lingual data and sign language interpretation from video footage in a much more feasible fashion. Work has also been done to develop pose estimation techniques specialized for sign language videos [9]. However, Convolutional Neural Network (CNN)-based approaches have been more common.

Before deep learning, statistical and rule-based approaches were popular in various forms of works in the field of sign language. But it was not possible to achieve great results using these methods as sign language is a complex system with delicate upper-body movements and facial expressions, as mentioned earlier. Also, there is the case of different signers signing differently, which may make it difficult for translation systems to infer signs accurately.

Analysis of sign language in terms of recognition can be done in three separate tracks: isolated sign recognition, continuous sign recognition, and sign translation. In case of isolated sign recognition, the task is to recognize a single sign from a footage. Here, the source will not contain more than one sign gesture. This makes this task significantly easier than the other two tasks. In case of continuous recognition, the task is to predict sequences of gloss, which will be explained in detail in subsequent sections. Finally, in case of translation, the task is to generate meaningful spoken sentence, which is elaborated in the next section.

In recent times, deep learning has come into existence, and computer vision has seen innovation by leaps and bounds. Systems based on handcrafted features and sequence modeling using classic algorithms are not good enough anymore. Recent progress has made it possible to use different deep architectures like convolutional neural networks and graph-based networks to extract more robust and discriminative features from sequences of sign frames. Hence, it has been possible to encode these intricate aspects of sign language systems and to produce useful works in this particular field in different forms. One aspect of such work is Sign Language Translation (SLT), where the generation of spoken language is key.

1.1 Sign Language Translation

Sign language translation is the task of generating meaningful spoken sentences from a given sequence of signs. It is different from Continuous Sign Language Recognition (CSLR) in the fact that, in the case of continuous sign language recognition, the goal is to generate glosses from a sequence of signs. A gloss is just a label. It is the fundamental block of sign language. Each gloss is associated with the meaning of a particular sign. [10] As sign language has its own set of grammatical rules and ordering, sequences of glosses don't form any appropriate spoken sentence. They only merely indicate what each sign in a sequence refers to. Hence, this task has less practical contributions in the real world. It is important to differentiate between Continuous Sign Language Recognition (CSLR) and Sign Language Translation (SLT) as the relationships are different. The mapping between signs and gloss and the mapping between signs and texts or gloss and texts can be very different depending on the context. They can also pose different levels of difficulties. However, Sign language translation can be thought of as an extension of the continuous recognition task. It aims to generate meaningful spoken words in an appropriate order that we use regularly in our day-to-day life. An example of sign translation is illustrated in Figure 1.1.

From the illustration, it can be noticed that although the mapping between signs

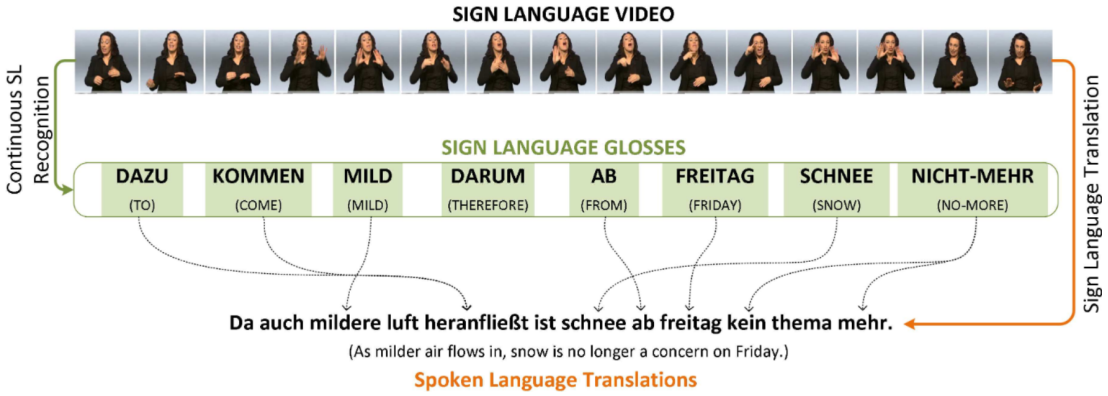


Figure 1.1: Example of Sign Language Translation. (courtesy [1])

and gloss follow a linear order, the mapping between sign and text and gloss and text is not so straightforward. There is a presence of different ordering of words, which makes SLT significantly different and inherently more difficult than CSLR.

1.2 Application of Sign Language Translation

Sign language translation has great potential as a practical real-life application task. Over 5% of the total population of the world are deaf. Expertise in sign language is very rare in comparison. This creates a communication barrier between signers and non-signers, resulting in a significant portion of the deaf people living in society to lag behind in various aspects of life. The advent of deep learning and computer vision has opened up a door of opportunities, where any real-time sign language translation application can mitigate the communication gap between signers and normal people. Applications like Google Translate have reduced the communication barrier between people of different mother tongues in a significant way. It has been possible due to its real-time translation capabilities and robust translation. More research is needed on Sign Language Translation to introduce better-performing translation models and lightweight models so that a Google Translation equivalent of Sign Language Translation (SLT) system can be developed. These types of applications will influence greatly to mitigate the communication gap existing between sign language users and non-users. On the other hand, as sign language consists of intricate movements of human body parts along a significant temporal dimension, any types of improvement done in

this domain can also contribute to other research domains like human-action recognition [11], facial expression recognition [12], gesture recognition [13]and so on.

Real-time Sign Language Translation (SLT) system has a lot of applications in recent times with all the innovations going around in mobile and web application development. Previously, in addition to less amount research in this domain, poor hardware capabilities and limitation of software development hindered the progress of real-time sign translation systems. But recent innovations has enabled organizations like Evalk [14], MotionSavvy [15] to develop real-time software applications with the capability to generate text from sign, speech from sign, text from speech, and so on. Figure 1.2 shows the application developed by MotionSavvy. All these innovations have been possible due to the research in the field of Sign Language Translation (SLT) recently. Exploring better ways of capturing spatiotemporal features in real-time will enable more developments of similar kinds.

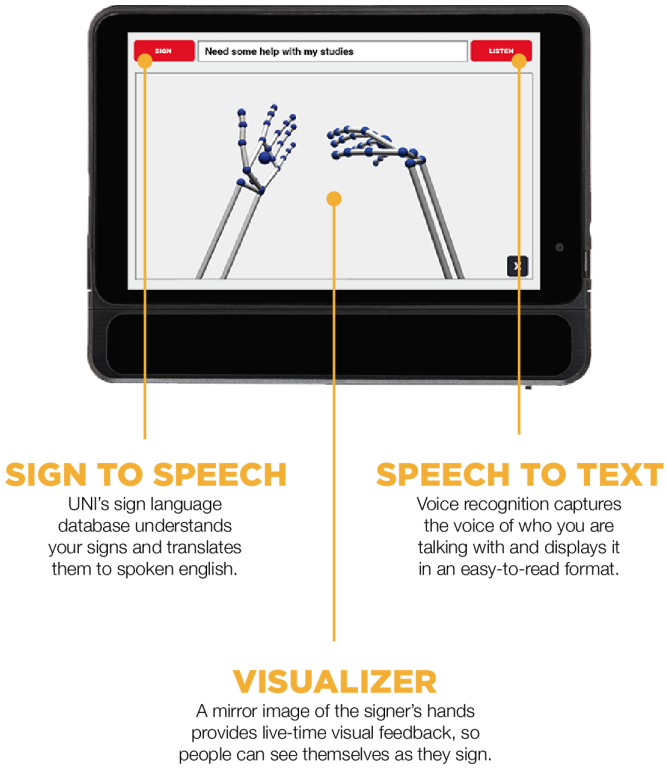


Figure 1.2: MotionSavvy: Sign Language Translation Application.(courtesy [16])

1.3 Problem Statement

Extracting robust features from sign streams and their temporal modeling is a very challenging piece of work as there are a lot of aspects to pay attention to. Hence, it has been our main focus to try to build a pipeline which can extract robust and distinct features from sequences of signs and generate more accurate spoken sentences. Joint-pose estimation has been our focal point in this regard. Skeletal-based joint-pose estimation systems have proven to be a robust feature extraction tool in case of different action recognition tasks recently due to their ability to encode complex human dynamics well and strong adaptability to different types of backgrounds. Sign language recognition or translation is also a form of action recognition tasks, albeit a more challenging one. Many recent works have adopted this type of architecture to achieve better results, which inspired us to work with the same. More precisely, our goal can be expressed as:

"To build a Sign Language Translation pipeline, which will extract glosses from sign videos and generate appropriate spoken sentences from those glosses."

1.4 Research Challenges

Complex human motion and dynamics pose a challenge on that task of sign language recognition and translation. In addition to this, there are many other challenges in this domain. These challenges are described below:

- **Occlusion:** Occlusion is the obstruction of background body parts from view by any other parts in the foreground. This phenomenon is very common during the signing process, as some parts of fingers or palms may get obscured from view by some other parts while performing complex movements.

- **Variance of Signers:** All signers don't sign in a completely identical way. There might be subtle differences in their execution of the same sign. This issue needs to be taken into account.

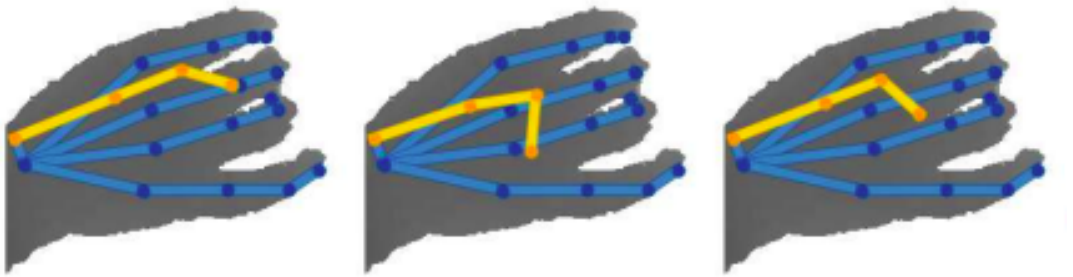


Figure 1.3: Example of Occlusion. Here, the thumb is being obstructed by other fingers.(courtesy [17])

- **Movement Epenthesis:** The number of frames with no sign information, which exists between the transition of two consecutive frames is known as movement epenthesis. Not all frames contain sign information. Some frames may capture the adjustments done by a signer after completing a signing and before starting another. Identification of these frames is a challenging task.

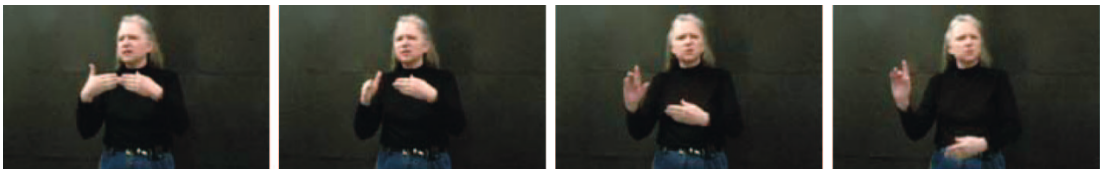


Figure 1.4: Example of Movement Epenthesis segments. The first frame is the end of sign "Gate", the last frame is the start of sign "Where". (courtesy [18])

The first frame in the example above marks the end of a gloss, while the last frame marks the start of a new one. There are multiple transition frames in between these frames that have no meaning. These frames are known to be **ME** (Movement Epenthesis) segments.

- **Subtle Facial Expressions:** Non-manual attributes like facial expressions play an important role in execution of some signs. [19] These expressions are sometimes so subtle that it is difficult to notice these fine characteristics, leading to misunderstanding any signs.

- **Temporal Boundaries and Alignment:** Signs may span over different number of frames depending on execution speed, the inherent nature of the sign, and

other factors. Some signs may span over less number of frames, and some may span over a larger number. Hence, identifying the starting and ending frame of each sign gesture is a challenge. This in turn also helps in the alignment of the frames and glosses.

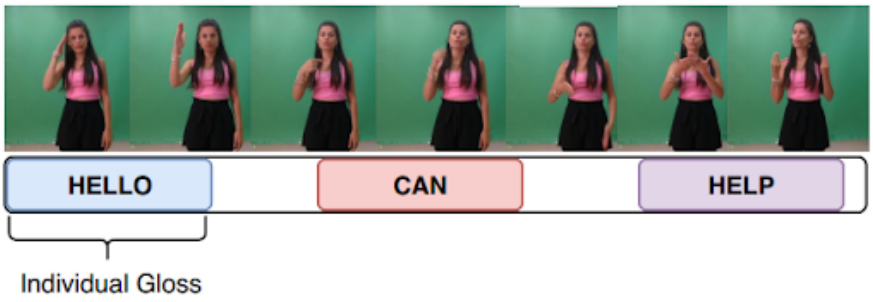


Figure 1.5: Example of Temporal Boundaries and Alignment. Here, the glosses 'Hello', 'Can', 'Help' have their own boundaries which should be identified as accurately as possible.(courtesy [5])

- **Varying Backgrounds:** Signers present in real-life scenarios may introduce complex background information, which acts as a noise. As background information is irrelevant to the task of sign translation, these information may introduce uncertainty and confusion within a sign translation system, leading to poor recognition of signs and substandard level of translation. As a result, it is necessary to reduce the level of noise in sign footage, which introduces a different level of challenge.

Figure 1.6: Example of various backgrounds of signers.(courtesy [20])

- **Nature of Gestures:** Some sign gestures require movement of a single hand, others require movement of both hands. Context and meaning may differ de-

pending upon this single-double movement. Accurate identification of this distinction is a challenge, as the translation system may wrongly put more emphasis on a hand which is taking no part in the execution of sign.

There are many other challenges that exist in this field, such as, execution speed difference of the signers, subtle orientation of pose, etc. In addition to these, lack of annotated datasets, weakly aligned gloss annotations, lack of expert guidance while preparing annotations, standardization of sign language vocabularies etc. also pose a considerable challenge to the field of Sign Language Translation (SLT).

1.5 Contributions

We can encapsulate our significant contributions of our work in the following points:

1. Utilization of a better pose estimation model with the help of Darkpose [4] employed on top of HRNet [3], so that human joints can be estimated more accurately.

2. Introduction of Graph Convolutional Networks to Sign Language Translation via SL-GCN.

3. Generation of bone and two types of motion features from joint information for robust spatial and temporal modelling respectively and demonstration of their effectiveness in Sign Language Translation.

4. Demonstration of results found using different segment size of the input to find optimal sign segment size.

1.6 Organization of Thesis

The rest of the paper is organized as follows: in Chapter 2, we describe some of the relevant works related to our domain, as part of our literature review. In Chapter 3, the overall pipeline of our work including a detailed explanation of each module of the framework is described. In Chapter 4, we analyze the results obtained by our pipeline under different hyperparameters and show necessary comparisons. Finally, we draw some conclusions and highlight potential future research directions in Chapter 5.

Chapter 2

Background Study

Sign language has been studied for a long time in the field of computer vision, although, most of the works focused on Isolated Sign Language Recognition. However, in recent years, with the advent of deep learning approaches, significant research has been conducted on Continuous Sign Language Recognition (CSLR) [21, 22] and subsequently on Sign Language Translation (SLT). Before the advancements in these fields, recognition models were built using traditional methods and handcrafted features [23,24]. On the other hand, for time-related feature processing, different classical approaches like Hidden Markov Model(HMM) [25], Random fields [26], DTW [27], and others were used. However, with the advent of vision research, researchers working on sign language recognition adopted CNNs [28] for visual feature representation [29] and RNNs for temporal modeling. However, the recent introduction of transformers [30] has replaced other recurrent-based models for temporal modeling tasks, leading to significant improvement of results in CSLR and SLT research [31]. Another significant contribution to CSLR and SLT research has been the development of sequence-to-sequence learning approach, which learns mapping between sign and gloss or spoken sentences. Recent SLT works [1, 31] employ the same method using an encoder-decoder architecture for time-related modeling. In summary, the current works on SLT all follow a standard design of framework:

- A visual feature extractor module to extract features from input frames of videos of signs. This module can be based on CNN architecture [1,31], Keypoints [32],

etc. Recently, significant work has been done on Isolated SLR [20, 33] by introducing skeletal-based methods and graph convolution. These kinds of methods require the help of different pose estimation models.

- A Temporal Alignment Module to learn mapping between the input frames and sign gloss and sentences. Recurrent models like RNN, LSTM and non-recurrent models like Transformers serve this purpose of translation. A CTC loss [34] layer is also incorporated within it to calculate sequence generation loss effectively.

2.1 Pose estimation methods

Employing skeletal-based visual feature extraction method instead of RGB-based methods in the field of CSLR and SLT requires human keypoint or joint estimation from RGB frames. Different pose estimation models come in handy for this purpose. In recent times, much research has been conducted in order to develop better pose estimation systems. HRNet [3] presented a new pose estimation model with a focus on a dependable high-resolution output of human pose. Even though most of the current models use a high-to-low resolution model, their method proposed a first-of-its-kind model that keeps high-resolution representations throughout the whole process.

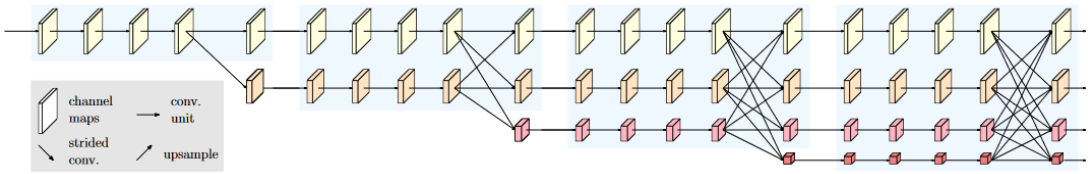


Figure 2.1: An example of a high-resolution network. Only the main body is illustrated, and the stem is not included. There are four stages. The 1st stage consists of high-resolution convolutions. The 2nd (3rd, 4th) stage repeats two-resolution blocks.(courtesy [3])

Figure 2.1 displays the deep high resolution model employed by the model. It starts with its input at the high-resolution subnetwork as the first layer. The model then steadily adds high-to-low resolution subnetworks one by one to form more layers and then connect the multi-resolution subnetwork layers in parallel. Then these

subnetwork layers are repeatedly fused together using multi-scale fusions such that each of the high-to-low resolution representations gets information from other parallel representations over and over, leading to rich high-resolution representations.

Different promising research has also been conducted in developing robust algorithms and methods, which can be incorporated into pose estimation models in order to enhance their estimation performance. One such method is DarkPose [4], which aims to enhance the accuracy of already existing methods by proposing an improved way of performing coordinate encoding and coordinate decoding. Coordinate encoding and Coordinate decoding both together is called Coordinate Representation. The standard coordinate decoding involves transforming the predicted heatmap of the joint into a coordinate position in the original image space. However, the heatmap is not always of the same spatial size as the original image. Thus the heatmaps need to be upsampled for accurate coordinate prediction. This upsampling causes the sub-pixel localization problem, resulting in an inaccurate prediction of joint coordinates. Therefore, to alleviate the problem it is suggested to use the predicted heatmap equation which assumes that the predicted heatmap is a 2D Gaussian distribution. Figure 2.2 shows this coordination decoding method.

$$\mathcal{G}(x;\, \mu,\, \Sigma) = \frac{1}{(2\pi)|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \tag{2.1}$$

Then Taylor's theorem is used to find the value of μ which in turn helps to calculate the coordinate position of the heatmap.

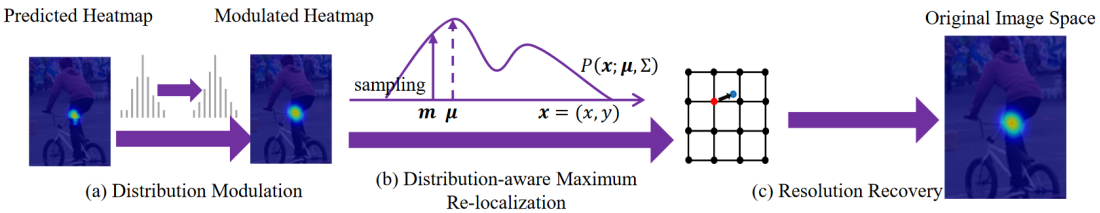


Figure 2.2: Overview of the proposed distribution aware coordinate decoding method.(courtesy [4])

Moreover, a modulation technique called Heat Distribution Modulation is also introduced. As stated before, it is assumed that the heatmaps follow Gaussian distribu-

tion however, the output heatmap of the model gives out multiple peaks close to the maximum activation. As a result, this may cause adverse effects upon the performance of the decoding algorithm. To smooth out the heatmap distribution the authors propose to perform convolution with heatmap and Gaussian Kernel which has the same variation as the training data.

Additionally, a superior coordinate encoding technique is also proposed. Coordinate encoding transforms the coordinate to heatmap. While performing coordinate encoding the original image is downsampled to the size of the model input. Traditionally, the downsampled image is quantised then heatmap is generated from the Gaussian equation, however this quantisation of coordinate causes the value to be inaccurate causing the generated heatmap to be erroneous. Thus, it is suggested that quantization is not done. Rather the raw coordinates are used as input to the Gaussian equation. This results in a better prediction of heatmap from the coordinate. An illustration of this heatmap modulation technique is shown in Figure 2.3.

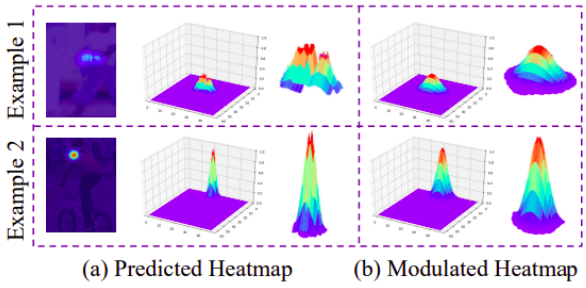


Figure 2.3: Illustration of heatmap distribution modulation. (a) Predicted heatmap; (b) Modulated heatmap distribution.(courtesy [4])

2.2 Feature extraction architecture

All general Sign Language Recognition and Translation pipelines use a visual feature extraction module at first to extract high-level spatiotemporal features from input images and videos. These features are generated using different methods, mainly RGB-based methods with CNN backbone [35], skeletal-based methods with Pose Estimation [36], depth-based methods [37], optical flow calculation [38], and so on. Recent

works on CSLR and SLT focus on two types of architecture predominantly:

- Keypoint based architectures
- Convolution based architectures

2.2.1 Keypoint based architectures

In these types of architecture, the keypoints or joints from the human body are estimated first using any pose estimation models. These keypoints are given as input to the temporal-alignment module to predict the sign. These points are passed through different types of deep learning models such as RNN [39], GRU [32], GCN [36] etc to extract the features and use them to predict the gloss sign.

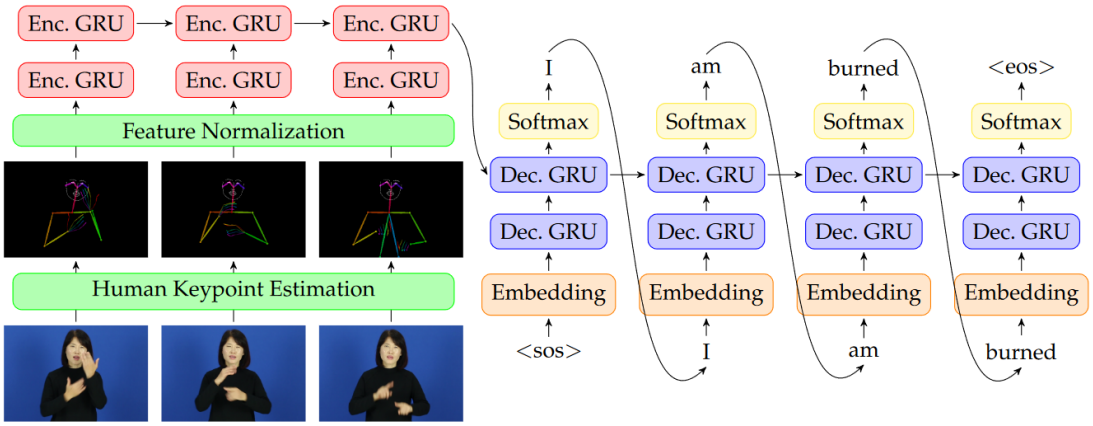


Figure 2.4: Architecture of Neural Sign Language Translation that translates a sign language video into a natural language sentence using sequence to sequence model based on GRU cells.(courtesy [32])

[32] first uses OpenPose [40], a pose estimation model, to extract necessary keypoints. These keypoints are normalized and reshaped and fed into an encoder-decoder structure consisting of layers of GRU [41]. The output of this structure is translation, that is, spoken sentences. The whole architecture is shown in Figure 2.4.

[20] used the key points to create skeleton graphs connecting the nodes that build up a bone-like structure to model the complex coordination between different joint of the body and applied Graph Convolutional Networks [42] for prediction. This work demonstrated the incorporation of three different architectures (SL-GCN, SSTCN and

3DCNN) and used RGB Video and Depth video in multiple pipelines and ensembled the result for their final output. Their whole pipeline is illustrated in Figure 2.5.

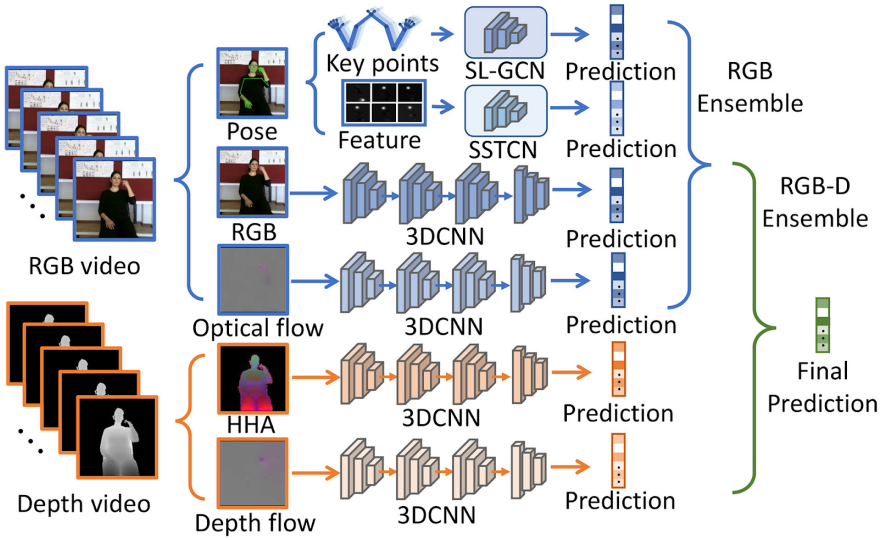


Figure 2.5: Skeleton Aware Multi-modal Sign Language Recognition Framework (SAM-SLR).(courtesy [20])

The pose keypoints were extracted from the RGB data using HRNet [3]. Now from the pose estimated data we get four different channels of data:

- Joint
- Joint Motion
- Bone
- Bone Motion

Here, the bone, joint motion and bone motion streams are generated from joint or keypoints estimated using the pose estimation model. Bone is a vector connecting the human body joints, which captures the coordination between different joints within the body. It helps to capture the spatial features within a frame. Joint Motion is a set of vectors connecting inter-frame joints, which helps to capture time-related features from a set of frames, capture complex dynamics of subtle movements of signs, and so on. Finally, Bone Motion is a set of vectors that tries to do the same job as Joint Motion, but for bone streams. That is, it tries to capture relations between inter-bones within a set of frames.

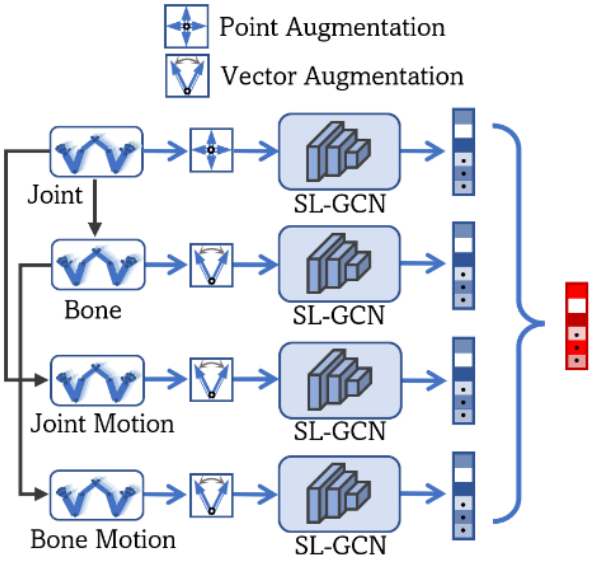


Figure 2.6: Multi-stream Workflow(courtesy [20])

These four data streams are used as input in four different SL-GCN modules to predict the gloss in the video. Finally the predictions are ensembled to generate a final output prediction, as shown in Figure 2.6.

Inside the SL-GCN architecture there is a decoupled spatial convolutional layer (Decouple SCN), a spatial, temporal and channel-wise attention module (STC) and a temporal convolutional layer (TCN), shown in Figure 2.7

These modules combined with Batch Normalization (BN), Drop Graph modules and ReLU functions form the SL-GCN architecture. The drop graphs work similar to skip connections in CNN architectures like ResNet [43]. In our pipeline, we use this SL-GCN architecture to extract useful features and feed them to the temporal-alignment module to get our final output.

2.2.2 Convolution based architectures

In these types of architectures, 2D or 3D convolutions are applied on input frames to extract spatial, temporal and spatio-temporal features. However, in recent times, some research has been conducted, which takes estimated keypoints as input to a CNN architecture and generate feature and prediction accordingly. [44] used this approach, which is illustrated in Figure 2.8.

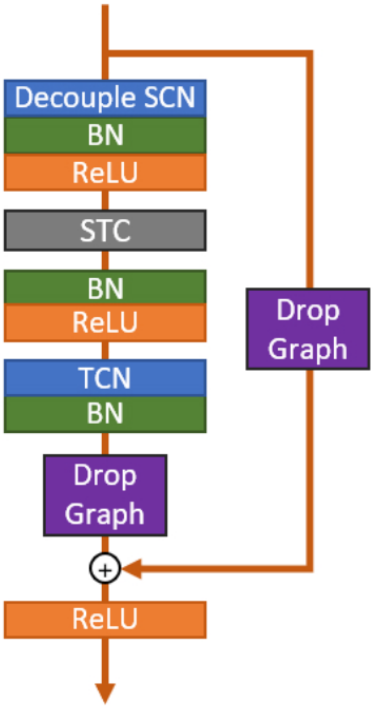


Figure 2.7: SL-GCN unit architecture.(courtesy [20])

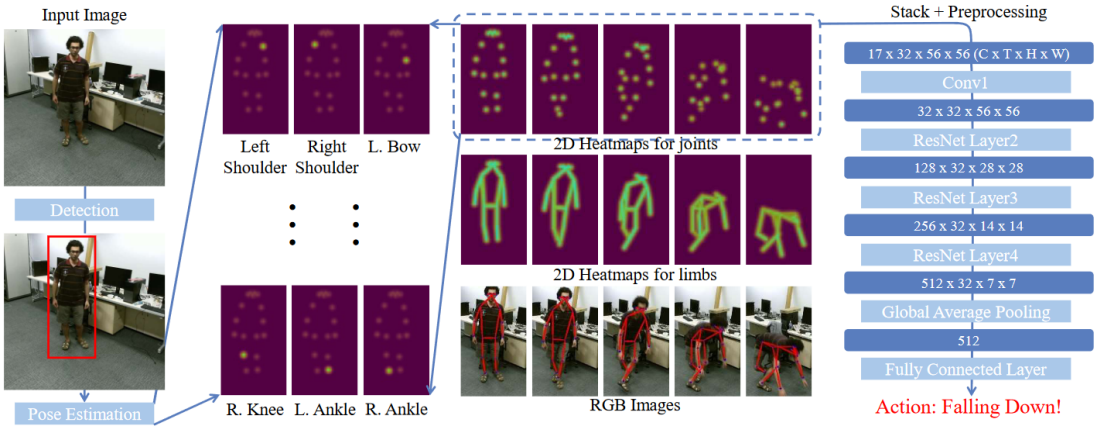


Figure 2.8: PoseConv3D Framework.(courtesy [44])

Here the RGB images are stacked and 3d convolution is used to recognise the action. The pose is only used to create a heatmap of the underlying graph but the graph itself is not used.

Similar to this model, the SSTCN module and 3DCNN from [20] also use the RGB images on these two architectures to do the predictions. A 3D CNN architecture is also used in [45].

The frames are convolved with multiple sized filters to extract hierarchical rep-

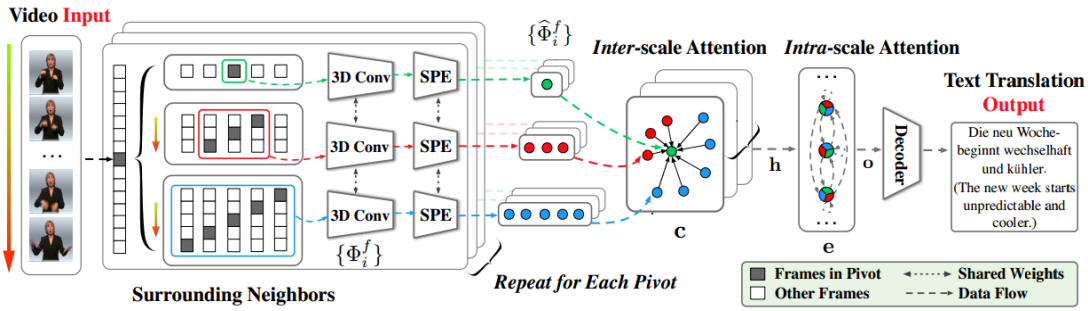


Figure 2.9: TSPNet Pipeline with Intra-Scale and Inter-Scale Attention.(courtesy [45])

resentation of segments, as shown in Figure 2.9. Here the large segments focus on global context and small segments focus on individual signs. In combination, focus is given on both local semantics and global context together, which is a crucial part of generating meaningful words in case of Sign Language Translation.

2.3 Translation methods

A common norm in Sign Language Translation in recent years has been the use of encoder-decoder based architecture [46] to replicate a Neural Machine Translation (NMT) [47] architecture. These encoder-decoder structures consist of layers of recurrent neural networks, like RNN, GRU, LSTM. Recently, mechanism of self-attention has flourished, resulting in the increase use of transformers instead of recurrent architectures. [48]

[1] was the first to formalize the task of sign language translation (SLT) in the framework of Neural Machine Translation, in an end to end fashion. The authors showed that by defining the translation task as a machine translation problem, it is possible to jointly learn spatial embeddings of sequence of frames, the language model and the mapping between sign and spoken language. This paper also presented the "RWTH-Phoenix-Weather 2014T" dataset, which has become the standard benchmark dataset in the field of sign language translation.

As illustrated in Figure 2.10, the pipeline proposed by the authors has AlexNet [49] as the spatial embedding layer, and an attention-based encoder-decoder network, consisting of two deep RNNs. The encoder takes the features extracted from the CNN as

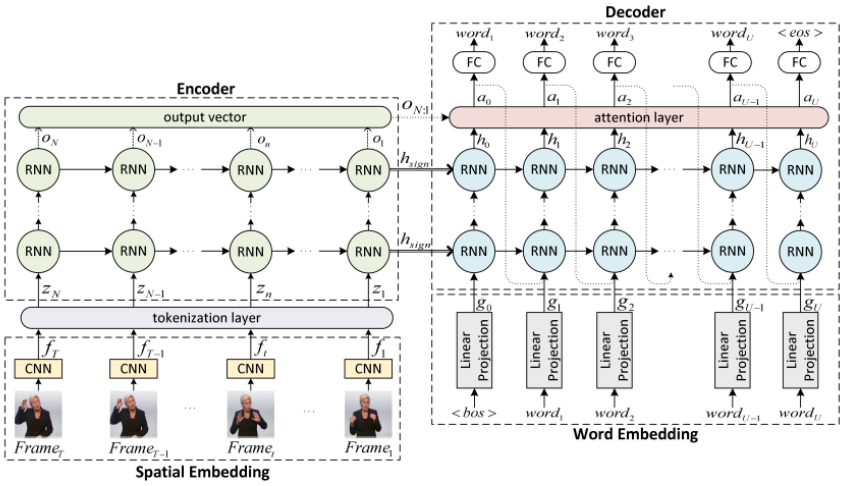


Figure 2.10: An overview of our SLT approach that generates spoken language translations of sign language videos.(courtesy [1])

input, while the decoder outputs spoken word sentences. In the decoder part, the target sequences are projected in a latent space using word embedding. The word embedding is a fully connected layer that learns a linear projection from the one-hot vectors of spoken words to a denser space.

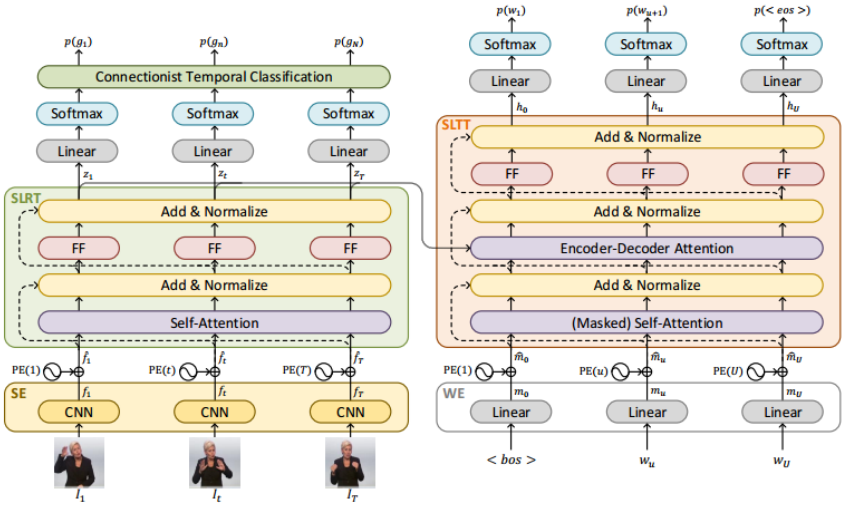


Figure 2.11: A detailed overview of a single layered Sign Language Transformer. (SE: Spatial Embedding, WE: Word Embedding , PE: Positional Encoding, FF: Feed Forward).(courtesy [31])

Next, the same authors in [31] incorporated transformers to jointly learn both sign language recognition and translation in an end-to-end fashion. No explicit form of gloss representation was generated. Rather, connectionist temporal classification (CTC) loss [34] was injected into the encoder of the transformer to facilitate recogni-

tion of gloss representation, keeping the translation as an end goal for the decoder to output. They showed that this kind of joint learning improves performance across both recognition and translation. The framework is shown in Figure 2.11.

The authors used Inception Net [50], EfficientNet-B0, EfficientNet-B4 and EfficientNet-B7 as the spatial embedding layer and showed that EfficientNets [51] give much better performance. The encoder-decoder network used transformers as their base model. The transformers had 512 hidden units with 8 heads in each layer. A CTC (Connectionist Temporal Classification) loss was incorporated in the encoder of the transformer in order to recognize continuous glosses. The transformers had 3 layers in both encoder and decoder for the gloss and spoken.

Chapter 3

Proposed Architecture

The objective of our research is to create a Sign Language Translation pipeline that extracts sign language glosses from videos and generates spoken language translations from them. Our general pipeline is divided into 2 modules, Visual Feature Extraction Module and Temporal - Alignment Module. For extracting sign glosses from the videos, the background does not contribute to give any meaning to the sign thus our sole focus is to extract features from the user while signing or performing gestures. Hence human joint keypoints can help us to mitigate the complex background information and also alleviate the occlusions caused while signing. In the background study section, we have seen that most of the visual feature extractor modules for SLT are CNN based, or non-pose based. However, in [19-Reference] the authors used human joint coordinates as visual feature, although the keypoint extraction method is not a contemporary one to present time. Thus we proposed to used HRNet [3] and on top of that introduce Darkpose [4] preprocessing steps to estimate the joint positions with greater accuracy. Although, joint pose is used, however as far as we know, there are no feature extraction process carried out which involves GCN for SLT. Hence we proposed to use SL-GCN [20] which performs graph convolution to extract the significant feature of relation between each joint and thus giving a higher level feature for our adjacency matrix made from joint and their corresponding edges. Moreover, in the background study we have observed that for SLT task many type of Temporal modules were used such as RNN, GRU, LSTM, and quite recently Transformers which out-

performs the previous temporal module. Therefore, we proposed to use Transformers for Alignment and Temporal Module. This section begins with a description of our general pipeline, followed by an overview of the various components therein.

3.1 Overview of Pipeline

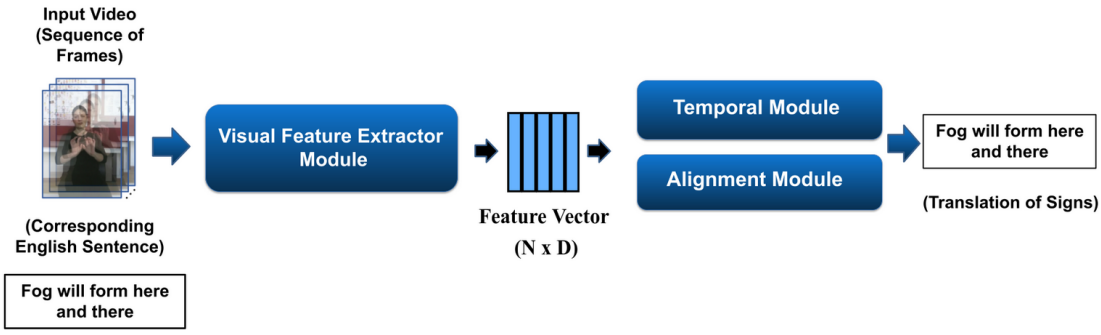


Figure 3.1: Block Diagram of Overall Pipeline.

Here first we give input a video or a collection of frames from which the Visual Feature Extractor gives output of feature maps. These feature maps are passed as input to the Temporal and Alignment module such that the Alignment module aligns the corresponding set of frames against corresponding set of gloss predicted by the Temporal module and thus a spoken sentence is generated which is a collection of gloss making a meaningful sentence. As it can be seen in Figure 3.1 the video consists of a signer making visual sign gestures to express “Fog will form here and there” in sign language. Hence the ground truth value for the dataset is the following sentence “Fog will form here and there”. We give input only the video to our model and after translation the output is a set of gloss labels as follows “Fog will form here and there”. Thus in this example the generated output matches with our ground truth value. We aim to generate an output spoken sentence which is as close as possible in similitude to the ground truth.

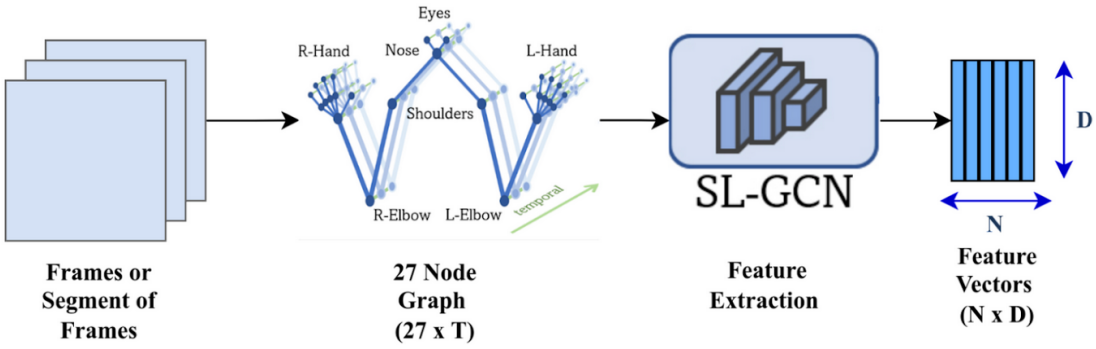


Figure 3.2: Visual Feature Extraction Module

3.2 Visual Feature Extraction Module

This module consists of extracting human joint positions and then extracting higher level features through the use of SL-GCN [20].

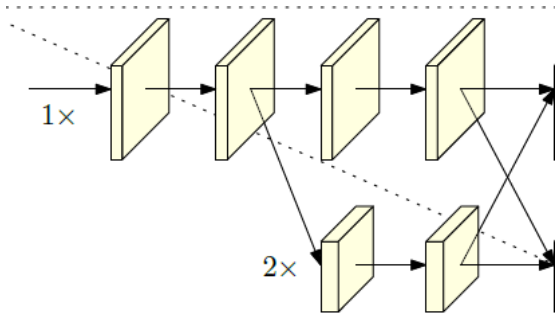


Figure 3.3: Network connecting high-to-low subnetworks in parallel (courtesy [3])

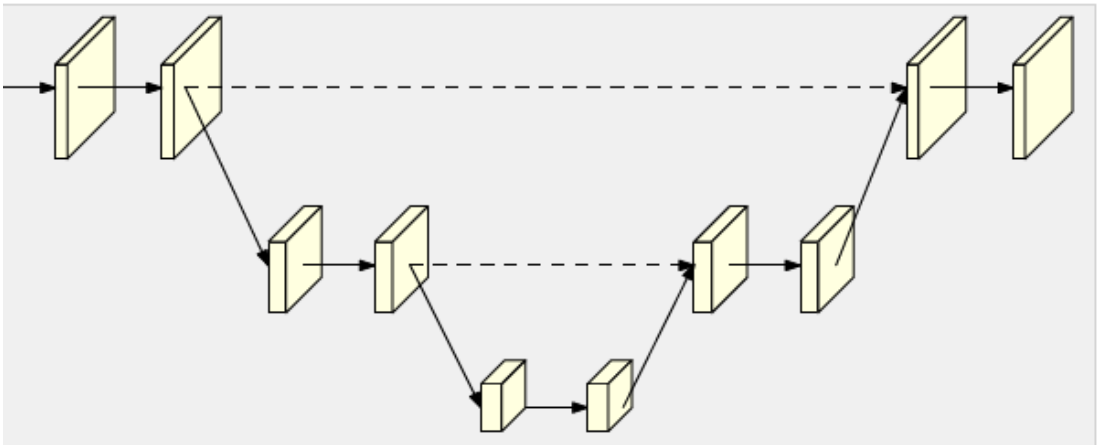


Figure 3.4: Traditional downsampling to upsampling network(courtesy [3])

For joint pose estimation we use Darkpose [4] since it essentially is a preprocessing technique which is used on top of HRNet [3]. It ensures both rich high-resolution

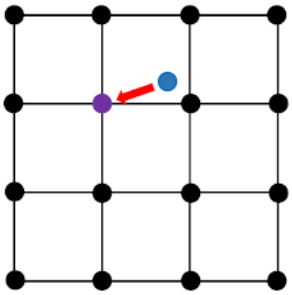


Figure 3.5: Traditional sub-pixel shifting(courtesy [4])

representations and better accuracy for Coordinate Representation and Heatmap Distribution Modulation. Coordinate Representation helps to predict better coordinate position of the heatmap and Heatmap Distribution Modulation helps to smooth a cluster of peaks into a single peak which gives a better estimation of coordinate position of the heatmap. Using joint position helps to mitigate occlusion which can occur in the RGB video. Figure:3.4 shows the traditional method where the network downsamples the features such that the feature size is decreased thus allowing faster computation and better memory management, and later after feature extraction the obtained features are upsampled, which causes significant contributing features to be lost due to loss of resolution. Hence we chose HRNet [3] which allows the higher resolution features, as shown in Figure:3.3, to be passed on to downsampled layer, thus allowing the high resolution features to contribute to the downsampled layers also, leading to recovery of resolution loss compared to traditional methods. Hence HRNet allows both efficient memory management and faster computation without compromising the contributing factors of the feature. After obtaining the high resolution features from HRNet [3], the preprocessing of Darkpose is used to further enhance the accuracy of joint position estimation. Darkpose utilises Coordinate Encoding, Decoding and Heatmap Modulation. Traditional Coordinate Decoding uses handcrafted feature of 0.25 sub-pixel shifting where the 0.25 is an arbitrary number, as shown in Figure: 3.5, without any intuition. Thus the proposed method in Darkpose includes using Taylor's Theorem to calculate the mean and using the mean we estimate the position of joint. Again, in traditional Encoding applies quantisation and uses the quantised values resulting in inaccurate

ground truth value. However, Darkpose alleviates this issue by using the unquantised value which results in accurate ground truth value. Additionally, the heatmap generated by tradition methods include many peaks thus Darkpose uses Heatmap Modulation to smooth out the peaks giving better estimate to joint positions. After extracting the joint position we can generate 3 streams of data from it as follows: joint, joint motion and bone. These 3 pose features are then passed as input to SL-GCN.

In Figure 3.2 the frames are given as input to HRNet [3] on top of which Darkpose [4] is applied as preprocessing step which further enhances the accuracy of joint estimation. While passing these 27 joint to SL-GCN data preprocessing is done to generate a graph from the 27 joints. The equation for graph formation are as follows:

Here A denotes the adjacency matrix and Equation 3.1 indicates the process to calculate the relation between each joints giving us the Adjacency matrix.

$$A_{i,j} = \begin{cases} 1 & \text{if } d(v_i, v_j) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

In Equation 3.2 x_{out} denotes the implementation of spatial GCN, where W denotes the trainable weight parameter.

$$x_{out} = D^{-\frac{1}{2}}(A + I)D^{-\frac{1}{2}}x_{in}W \tag{3.2}$$

SL GCN performs Graph Convolution which helps to extract features from the joint nodes only thus alleviating the background information and hence extracting better features for generating gloss, which does not depend upon any background details. This is achieved by passing a graph as input to SL-GCN and it performs convolution upon the graph thus generating higher level features of the joint nodes which are related to each other in a connected graph where each edge between the nodes infer a relation associated between the nodes.

This can be achieved by using aggregation function, as shown in Figure: 3.6, between all the neighbouring nodes from a target node and pass the aggregated feature

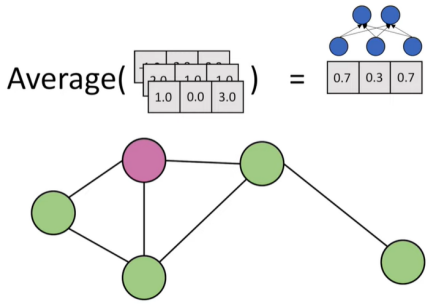


Figure 3.6: Feature Aggregation within neighbouring nodes

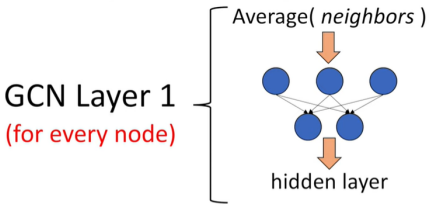


Figure 3.7: GNN hidden layer

to a dense neural network which gives a feature vector as output, as shown in Figure: 3.7, and this in turn again acts as input to another densely connected neural network. These feature vectors gives us the information of the relationship between each nodes. Thus at the end we get feature vectors of 3 data streams of joint, joint motion and bone as output of SL-GCN.

In Figure 3.8 we can observe that an rgb video of m frames is given as input to Darkpose. For each frame Darkpose extracts 133 keypoints thus our output is $m{*}133{*}3$ feature map. To avoid noisy data and inherent occlusion, we extract only 27 keypoints from the feature map resulting in a feature map of size $m * 27 * 3$. Next we choose a segment size of n and we are left with p number of n sized segments, hence the shape of the feature map is $(n{*}27{*}3){*}p$. These p number of segments are then passed on to SL-GCN to perform Graph Convolution giving us feature map of shape $(1{*}256{*}r){*}p$, where r varies upon the segment size n.

3.3 Temporal and Alignment Module

The main task of the temporal-alignment module is to take the features generated from the visual feature extraction module and perform a sequence-to-sequence learning to

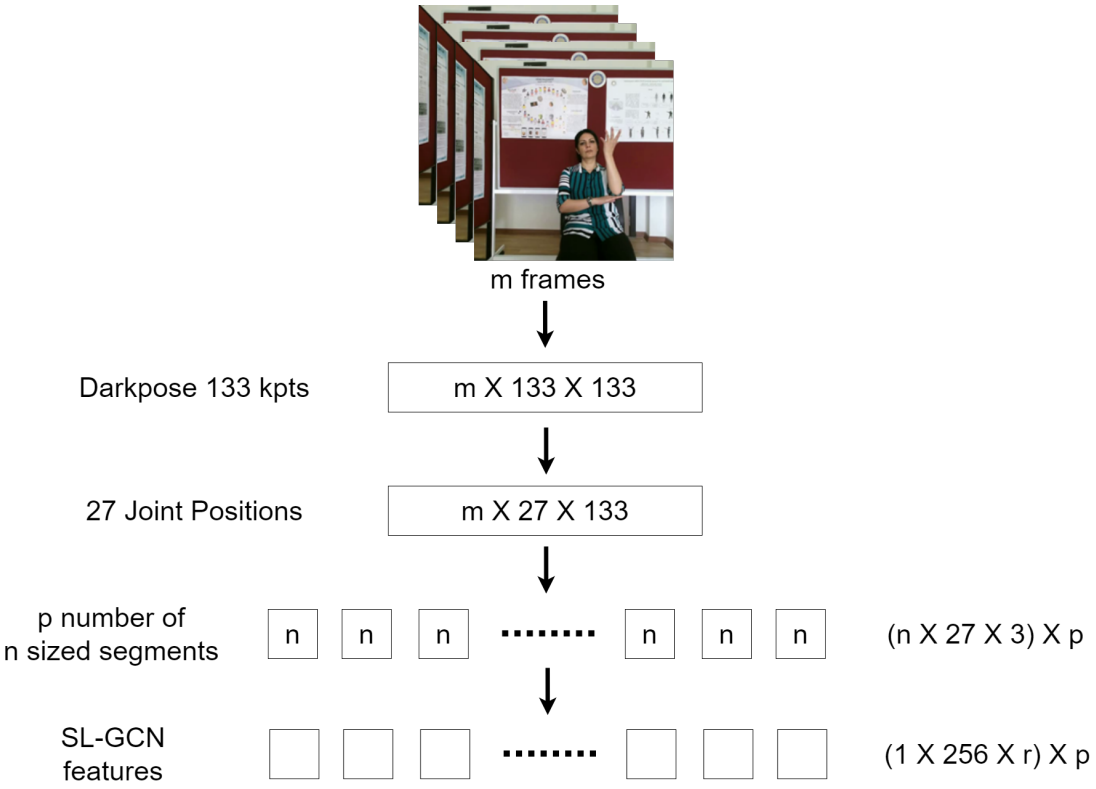


Figure 3.8: Feature map shape of SL-GCN.

learn the mapping between, firstly, sign features and sequence of gloss, and secondly, gloss sequence and sequence of spoken words. Use of an encoder-decoder network is common in these kinds of tasks. An encoder-decoder networks may consist of recurrent networks, like RNN, GRU, LSTM or a non-recurrent architecture like Transformers [30]. For our work, we use the architecture used in [31], where a transformer based encoder-decoder architecture is used to jointly learn sequence of gloss and spoken words from input features. An illustration of this module is shown in Figure 3.9.

The transformer-based temporal-alignment architecture works in two stages -

- Generation of gloss
- Generation of spoken words

The generation of gloss is handled by the encoder part of the transformer. It takes the input spatial and temporal information as features and tries to learn a mapping between these features and gloss. A robust mapping helps to predict a better sequence of gloss. In addition to this, the encoder also encapsulates the input features and gen-

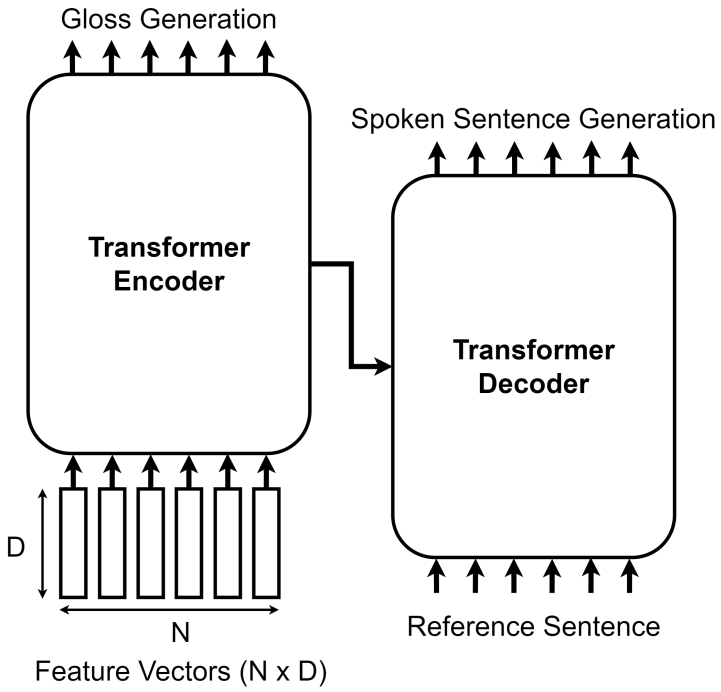


Figure 3.9: Temporal Alignment Module

erates a concise representation of the input features, keeping spatial and time-related information as concrete as possible. The encoder part is described in more detail in a subsequent section.

The generation of spoken words is handled by the decoder part of the transformer. It takes the concise representation of the input features generated by the encoder and the ground truth spoken sentence as input and generates a meaningful spoken sentence, one word at a time. The generated sentences are then evaluated using an appropriate evaluation standard. The decoder part is also described in more detail in a subsequent section.

In this way, the encoder and decoder work in a coordinated way and try to predict gloss and generate words by a process of collective learning.

3.3.1 Encoder Submodule

The encoder submodule of the transformer predicts sequence of glosses from input features, as we have already mentioned. This module is the same as the encoder architecture introduced in [31], illustrated in Figure 2.11. This part of the module is trained

using Temporal Classification (CTC) loss [34] to predict the glosses. It takes the spatial attributes produced by our visual feature extraction module from sign footage. In addition to predicting the glosses, the module also tries to learn spatio-temporal representations, keeping the end goal of translation in focus.

Given a sign video $V = (F_1, F_2, ..., F_M)$ with M number of frames as input, our visual feature extraction module generates features of shape NxD for each sample sign footage, where N is the number of segments produced and D is the feature dimension. Let's say the encoder takes as input a sequence of embeddings, $E = (S_1, S_2, \ldots, S_N)$. Using the positional encoding technique of transformers, a modified embedding, $\hat{E} - 1 : N$ is generated. This embedding gets passed into the core portion of the encoder submodule, illustrated in Figure 3.10.

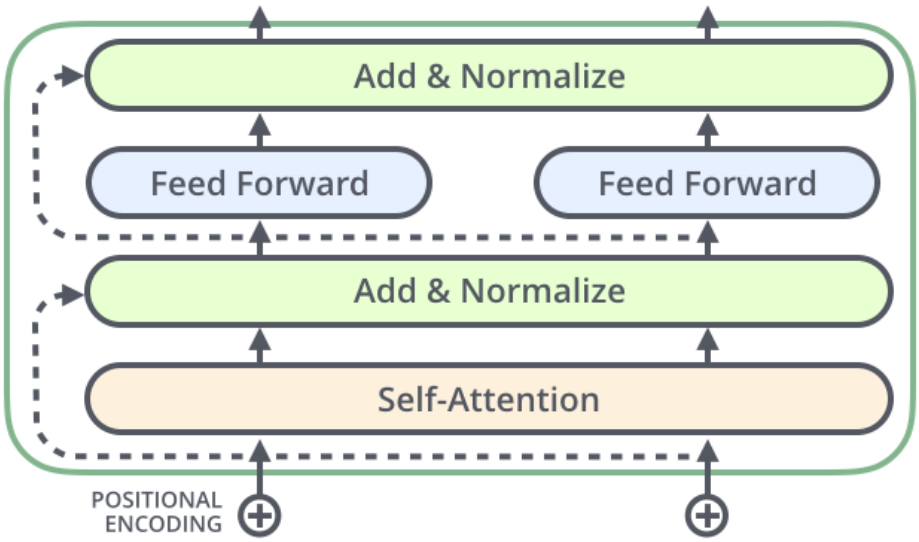


Figure 3.10: Block Diagram of The Encoder.(courtesy [52])

The modified embedding,$\hat{E}$ first gets passed through the self-attention layer, which learns to focus on the most relevant portions of the embeddings so that the performance of the overall module is as optimal as possible. This layer implements multi-headed attention, which is a process of generalizing the attention mechanism. Normally in simple attention, we have

$$Output = Attention(Q, K, V) \tag{3.3}$$

where, Q, K and V are the query, key and value vectors respectively. These are core components of self-attention mechanism. However, in multi-headed attention, the output is as follows:

$$Output = Attention(M_iQ, M_iK, M_iV) \tag{3.4}$$

where, each ith head focuses on certain aspects of the embedding. This ensures the model to solve any bottleneck issue in case of long sequence of features, mitigates the problem of vanishing gradient, and helps to learn alignment by itself in a more accurate manner. In our module, the number of multi-head attention is 8.

The attributes generated are then passed through a non-linear feed-forward layer. This layer helps in the regularization of the network and to learn non-linear patterns. In addition to these, there are several normalization layers and residual connections. Normalization layer simply standardizes the neural activations along the axis of features. This helps in the smooth training of the network. Residual connections, on the other hand, contribute in two key ways:

- Knowledge preservation
- Alleviate vanishing gradient

After passing through the core part of the encode submodule, the representation is then passed through a linear layer, followed by a Softmax layer to predict the gloss sequence. This is incorporated with CTC loss, which is commonly used as a sequence-to-sequence learning loss function. It has performed really well in Continuous Sign Language Recognition (CSLR) [53] in recent years. The loss is used onto $p(\mathcal{G}|\mathcal{V})$ by marginalizing over all possible $\mathcal{V}$ to $\mathcal{G}$ alignments as:

$$p(\mathcal{G}|\mathcal{V}) = \sum_{\pi \in \mathcal{B}} p(\pi|\mathcal{V}) \tag{3.5}$$

where, $p(\mathcal{G}|\mathcal{V})$ is the probability distribution of gloss.

Our model generates two diferent loss. One of them is recogntion loss, which evaluates the performance of the encoder submodule. The probability distribution $p(\mathcal{G}|\mathcal{V})$

is used to calculate the recognition loss, $\mathcal{L}_R$ as follows.

$$\mathcal{L}_R = 1 - p(\mathcal{G}^*|\mathcal{V}) \tag{3.6}$$

where, $\mathcal{G}^*$ is the reference gloss sequence, acting as the ground truth.

3.3.2 Decoder Submodule

The decoder submodule of the transformer generates sequence of spoken words, one at a time. This module is the same as the decoder architecture introduced in [31]. This part of the module depends upon the attributes learned by the encoder module. It also takes the reference spoken sentence as an input during training, and its generated sentence as input during evaluation. This decoder is autoregressive in nature, that is, it predicts future entities based on its past predictions. It is illustrated in Figure 3.11.

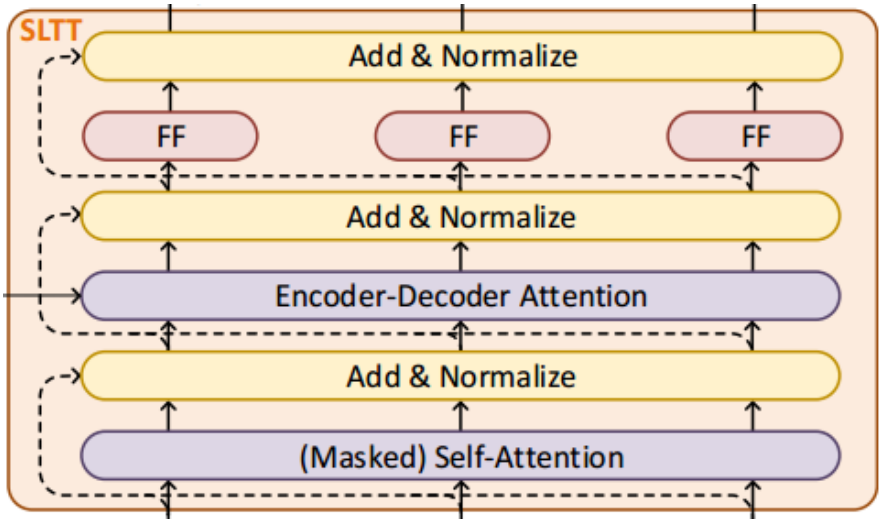


Figure 3.11: Block Diagram of The Decoder.(courtesy [31])

We first append the reference spoken sentence S with a special token <bos>, which indicates the beginning of a sentence. The words of the reference sentence are passed through a linear layer, which extracts word embeddings. These embeddings are passed through the decoder positional encoding layer, whose job is to remember the original ordering of the words within the sentence.

The positionally encoded embeddings are progressed through a masked self-attention

layer. The self-attention layer in the decoder submodule in itself is similar to the encoder one, except for the masking part. The decoder implements a mask over the self-attention layer inputs - so that the network can not see any future ground truth words while generating the current word. This ensures the autoregressive nature of the decoder, where it only gets the help of its own generated words so far to generate the next word. This makes the decoder more robust as it does not have access to any reference words during inference. The merging of the embeddings extracted from both the encoder and decoder self-attention layers takes place next. The output of the merge is then normalized and passed through an encoder-decoder attention module.

The goal of the encoder-decoder attention module is to find the mapping between the given embeddings and reference sequences. This is a crucial stage of translation, as the network learns to focus on the most relevant information in this stage. Attention to irrelevant information or noise is bound to generate poor translations. The results of the encoder-decoder attention are then passed through a non-linear feed-forward layer, followed by residual connections and normalization, similar to the encoder submodule.

As mentioned earlier, the decoder generates words, one at a time. This generation continues until the decoder generates the special token <eos>, which indicates the end of the sentence. The decoder is trained by breaking down $p(\mathcal{S}|\mathcal{V})$, which refers to the probability distribution of predicted spoken sequences S, given a sign footage V. The distribution is broken down in order to inject order into the conditional distribution using the following equation:

$$p(\mathcal{S}|\mathcal{V}) = \prod_{u=1}^{U} p(w_u|h_u) \tag{3.7}$$

Here, D is the target language and $p(w_u|h_u)$ represents ground-truth probability of word w_u^d at each decode-step u.The above equation also helps us to calculate the loss for each word. The translation loss $\mathcal{L}_T$ is calculated using the following equation:

$$\mathcal{L}_T = 1 - \prod_{u=1}^{U} \sum_{d=1}^{D} p(w_u^d) p(w_u^d|h_u) \tag{3.8}$$

Finally after obtaining these two losses, $\mathcal{L}_R$ and $\mathcal{L}_t$, the final loss $\mathcal{L}$ is calculated as follows:

$$\mathcal{L} = \lambda_R \mathcal{L}_R + \lambda_T \mathcal{L}_T \tag{3.9}$$

Here, λ_R and λ_T are recognition loss and translation loss respectively. The importance of recognition and translation loss can be varied during training accordingly by tweaking the weight values given in the above equation.

3.3.3 CTC Loss

Connectionist Temporal Classification (CTC) [34] loss is a really useful loss calculation technique for sequential tasks. Its practicality is based on the fact that, there is no need to know the alignment between input and reference to calculate this loss. Formally, given input mapping sequence, $I = (I_1, I_2, ..., I_p)$ and corresponding output mapping sequence $O = (O_1, O_2, ..., O_q)$, an accurate mapping from X to Y is to be found. An illustration of how CTC works is given in Figure 3.12. In the illustration, X is the input sequence. The predicted alignment is also shown, where the first two frames belong to *c*, the next three belong to *a* and the last two belong to *t*. The algorithm helps to generate a good output by smoothing out the alignments.

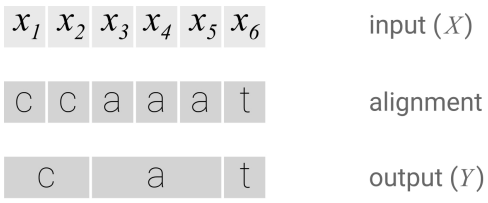


Figure 3.12: Simple CTC Alignment.(courtesy [54])

To obtain the output probability distribution given an input, CTC sums over the probability of all possible alignments between the two. These alignments give us a natural way to go from probabilities at each step of generation to the probability of an output sequence. The algorithm can attach any probability to Y, given an X. Its loss is calculated using dynamic programming to reduce the running cost.

Chapter 4

Result Analysis

4.1 Experimental Setup

All the experiments were done using cloud based systems, mainly Google Colaboratory. Primarily, three models of cloud GPUs were used: Tesla T4, Tesla K80 and Tesla P100. As a result, obtained inference and training time varied upon the performance of the corresponding GPU used.

Our transformer model was trained with an early stopping mechanism, which got triggered at around 40 epochs most of the time. The early stopping was applied based on the BLEU evaluation metric, with a patience of 8. The scheduling method used was plateau. We fixed a batch size of 32.

4.1.1 Hyperparameters

We experimented with two sets of hyperparameter values, one obtained from the literature and the other fixed by us. Initially, all of the experiments were done using given hyperparameters in [31]. After generating results using it, we tweaked hyperparameters to find an optimal set of values. The sets published in [31] and our set is shown below:

- Learning Rate: 0.001, Min Learning Rate: 1.0e-7, Weight Decay: 0.001, Decrease Factor: 0.7

- Learning Rate: 0.0001, Min Learning Rate: 1.0e-8, Weight Decay: 0.01, Decrease Factor: 0.25

The second set of hyperparameters, fixed by us, performed better. It can be attributed to the fact that, the learning rate in the first set is too high. This hinders the learning process of the transformer. Also, less weight decay value led to more chances of overfitting.

4.1.2 Decoding

Similar to [31], a greedy search technique is used during training to measure translation performance in valid set. However, during test time, beam search decoding is employed with the width ranging from 0 to 10. A length penalty is also implemented [55] with values ranging from 0 to 5. Valid set performance gives us the best width and penalty which are subsequently used for final results.

4.2 Datasets

For our experiments we mainly used two datasets. The two datasets are:

4.2.1 RWTH Phoenix Weather 2014T [1]

It consists of a collection of RGB frames of videos of different signers performing German sign language. It comes with ground truth glosses and spoken sentences for each video. There are 9 different signers, with more or less the same background and same outfit for the signers. The dataset is split as follows:

- Training - 7096 samples
- Valid - 519 samples
- Test - 642 samples

However, the dataset is based upon weather news, thus the signs and glosses cannot be ensured for generalization. An illustration of the dataset is given in Figure 4.1.

Figure 4.1: Phoenix Dataset has 9 different signers.(courtesy [1])

Table 4.1: Key statistics of the Phoenix Weather 2014T Dataset.

	Sign Gloss			German		
	Train	Dev	Test	Train	Dev	Test
segments	7,096	519	642	⟵ *same*		
frames	827,354	55,775	64,627	⟵ *same*		
vocab.	1,066	393	411	2,887	951	1,001
tot. words	67,781	3,745	4,257	99,081	6,820	7,816
tot. OOVS	-	19	22	-	57	60
singletons	337	-	-	1,077	-	-

4.2.2 AUTSL: Turkish Sign Language [2]

[2] introduced a new Large Scale Multi-modal Turkish Sign language dataset. Care was taken to preserve the system of communication using visual gestures, by incorporating simultaneous local and global articulations of multiple sources, i.e. hand shape and orientation, hand movements, body,posture, and facial expressions as used by deaf people.

Here the dataset consists of single gloss performed by a signer per video. In other words, it consists of a collection of isolated gloss. However, these glosses are performed among various different backgrounds including indoors and outdoors and by various different signers with different outfits and physical attributes, as shown in Figure 4.2. Thus the dataset is quite diverse which helps to form generalised data. Moreover, different signers have their own unique style to perform the same action of signing, as a result, the signs have a mix of variations. The dataset is split as follows:

- Training - 28142 samples

- Valid - 4418 samples
- Test - 3472 samples

Table 4.2: Key statistics of the AUTSL Dataset.(courtesy [2])

Property	Description
Number of signs	226
Number of signers	43
Total samples	38,336
Number of different backgrounds	20
RGB and depth resolution	512×512
FPS	30

AUTSL dataset offers challenges in the background; the data is collected from different indoor and outdoor environments as shown below. As a result, the videos contain high illumination changes and reflectance. It also contains variances in signer clothing, spatial positions of the signers and postural changes of signers in sitting and stand up positions.

Figure 4.2: Examples of different backgrounds from AUTSL.(courtesy [2])

A benchmark was created and shared for the evaluation of the models. In this context, a user independent test split was created using the samples from 6 signers. These signers were not included in the training or validation set.

4.3 Evaluation Metrics

Two commonly used metrics for measuring translation performance - BLEU scores and ROUGE-L scores.

4.3.1 Bilingual Evaluation Understudy (BLEU)

BLEU scores measure the closeness of the generated sentence to human reference sentence. It takes length, word choice, and word order into consideration. BLEU scores are within the range of 0 and 1. 1 means perfect translation, which is almost impossible, as even human translations are not foolproof.

BLEU-1, BLEU-2, BLEU-3, BLEU-4 refer to unigram, 2 grams, 3 grams and 4 grams calculation respectively. (Here, n-gram means sequence of n number of words)

Scores are calculated for individual translated segments—generally sentences—by comparing them with a set of good quality reference translations. Those scores are then averaged over the whole corpus to reach an estimate of the translation's overall quality. Intelligibility or grammatical correctness are not taken into account.

BLEU has frequently been reported as correlating well with human judgement, and remains a benchmark for the assessment of any new evaluation metric.

The following equations can be used to derive the BLEU scores as follows: p_n denotes the precision of the complete test corpus

$$p_n = \frac{\sum_{\mathcal{C} \in \{Candidates\}} \sum_{n-gram \in \mathcal{C}} Count_{clip}(n-gram)}{\sum_{\mathcal{C}' \in \{Candidates\}} \sum_{n-gram' \in \mathcal{C}'} Count_{clip}(n-gram')} \tag{4.1}$$

Here, BP denotes Brevity Penalty which is used for text normalisation. In Equation 4.2, r stands for length of reference corpus and c stands for length of candidate translation.

$$BP = \begin{cases} 1, & \text{if } c > r \\ e^{(1-r/c)}, & \text{if } c \leq r \end{cases} \tag{4.2}$$

For Equation 4.3, in standard case N = 4 and w_n = 1/N

$$BLEU = BP.exp(\sum_{n=1}^{N} w_n \log p_n) \tag{4.3}$$

4.3.2 Recall-Oriented Understudy for Gisting Evaluation (ROUGE)

ROUGE is also a metric which is used for evaluation of translation. Mainly there are five types of different ROUGE calculation. Among these, ROUGE-L is prominent in the literature of Sign Language.

ROUGE-L is a Longest Common Subsequence (LCS) based statistics. Longest common subsequence problem takes into account sentence level structure similarity naturally and identifies longest co-occurring in sequence n-grams automatically.

For the following equations p_{lcs} and r_{lcs} stands for precision and recall for LCS, respectively.

$$p_{lcs} = \frac{|LCS(S, \tilde{S})|}{|\tilde{S}|} \tag{4.4}$$

$$r_{lcs} = \frac{|LCS(S, \tilde{S})|}{|S|} \tag{4.5}$$

$$ROUGE - L = \frac{2 * p_{lcs} * r_{lcs}}{p_{lcs} + r_{lcs}} \tag{4.6}$$

4.3.3 Word Error Rate (WER)

It is the measure of erroneous glosses being generated by the predicted set of glosses. It does not account the similitude of meaning of glosses being used but rather it measures the score solely upon the exact gloss matching between predicted and ground truth set of glosses. The score ranges from 0 to 100% where 0 represents no occurrence of erroneous gloss and 100 represents all glosses being classified incorrectly. Thus the lower the score the better since 0 score infers perfect translation.

For Equation 4.7 $|\mathcal{G}^d|, |\mathcal{G}^i|, |\mathcal{G}^s|$ denotes the number of deletion, insertion and substitution required to transform the output sentence to ground truth sentence.

$$WER = \frac{|\mathcal{G}^d| + |\mathcal{G}^i| + |\mathcal{G}^s|}{|\mathcal{G}^*|} \tag{4.7}$$

4.4 Experimental Analysis

4.4.1 Implementation of Existing Literature

We reproduced the SL-GCN model from SAM-SLR [20] on our experiment environment and the results we got are shown below.

Table 4.3: SAM-SLR results reproduction

	Author's Result (%)	**Our Reproduced Result (%)**
Joint	95.02	**94.68**
Joint Motion	93.01	93.01
Bone	94.70	94.70
Bone Motion	92.49	92.49

As we can see almost all the stream's accuracy are reproduced by us, except the joint stream. We can positively infer that the authors have provided the weight of the joint stream for different epoch numbers rather than the one giving maximum accuracy. Hence the discrepancy in the results can be observed.

Next we tried to recreate the same experiment but using Darkpose [4] as the keypoint extraction module and the results are as follows.

Table 4.4: SAM-SLR experimental results using Darkpose

	Author's Result (%)	**Darkpose Result (%)**
Joint	95.02	92.30
Joint Motion	93.01	90.00
Bone	94.70	92.87
Bone Motion	92.49	90.15
Ensemble	95.45	93.86

As seen in Table 4.4, the results for Darkpose [4] are quite similar to the author's results. However, none of stream's accuracy for Darkpose surpasses the author's results. This can be due to the fact that our hyperparameters were not tuned, hence giving such results. Due to lack of GPU resources, hyperparameters could not be tuned leading to slightly less accuracy results with the authors'given one.

Then we tried to finetune the model from our previous experiment and compare it against the authors results.

Table 4.5: Comparison of results for Finetuned Darkpose Model

	Result by Authors	Our Pretrained_Normal	Our Pretrain_Finetune model			
Learning Rate	0.01	0.01	0.01	0.03	0.001	0.003
Joint	95.02	92.30	95.5	-	-	-
Joint Motion	93.01	90.00	-	-	-	-
Bone	94.70	92.87	95.37	93.28	96.04	96.08
Bone Motion	92.49	90.15	94.36	-	-	-

As we can see in Table 4.5, after finetuning the model we achieve better results than the author's results. We performed finetuning by varying the base learning rate upon our pretrained model. To achieve our pretrained model we trained the model using finetuned pretrained weights offered by the authors. Thus our pretrained finetuned trained model exceeds almost all the data streams. Other results could not be obtained due to limitation of lack of computing resources.

Next we tried to reproduce the TSPNet [45] architecture and our results are stated below.

Table 4.6: TSPNet results reproduction

	Author's Result	**Our Reproduced Result**
ROUGE-L	34.96	34.84
BLEU-1	36.10	35.99
BLEU-2	23.12	22.88
BLEU-3	16.88	16.55
BLEU-4	13.41	13.04

In the table 4.6, we show our reproduced result of TSPNet [45]. We achieved similar results like that of the authors. The slight discrepancies may be the result of difference of hyperparameters.

Finally we reproduced the Sign Language Transformers architecture and got the results shown below.

Table 4.7: Sign Language Transformers Results reproduction

Loss Weights ratio	***10:1***	
Metric	Paper Result	Reproduced
WER	***24.98***	29.38
BLEU-4	***22.38***	20.47

In Table 4.7, we show our reproduction of the results of the paper mentioned. The discrepancies between the results is due to difference in hyperparameter values as they were not not provided in the paper.

4.4.2 Results Using Our Extracted Features by Varying Different Hyperparameters

First we tried varying Segment size and Fixed Weight Ratio as 1:1.

Table 4.8: Comparison of result with segment size 16 and 1 as hyper parameter

	METRIC	OURS (JOINT STREAM)	
		segment size 16	**segment size 1**
Gloss Recognition	Word Error Rate	73.26	82.76
Translation to Text	BLEU-4	10.47	13.04
	ROUGE-L	31.89	35.99

The discrepancies between the results, as shown in the table, is due to the fact that segment size of 16 is too long. Many signs may exist within this segment, resulting in confusing our model. On the other hand, segment size 1 gives poor result because we do not explicitly tell our model to look for sign within a segment, we feed all the frames together.

We tuned and shifted the Fixed Weight Ratio up to 10:1 and again tried Varying Segment sizes and got the following results.

Table 4.9: Comparison of result with segment size 12 and 8 as hyper parameter

	METRIC	OURS (JOINT STREAM)	
		segment size 12	**segment size 8**
Gloss Recognition	Word Error Rate	67.87	59.86
Translation to Text	BLEU-4	12.86	13.66
	ROUGE-L	32.6	-

As shown in the table above, our model works better when the segment size in 8. We assume this is because most signs span around 8 frames in the dataset. However, in case of segment size 12, there may be cases when more than one sign exists within a segment, which confuses our model.

Next we tried to vary both the Loss Weights ratio while keeping number of frames per segment constant to get an idea of the best combination.

Table 4.10: Comparison of result for Segment size 8 with varying Loss Weights ratio as hyper parameter

Loss Weights ratio	8 Frame Per Segment	
1:1	WER	68.13
	BLEU-4	11.95
5:1	WER	58.79
	BLEU-4	13.38
10:1	***WER***	***56.47***
	BLEU-4	***14.07***
20:1	WER	59.83
	BLEU-4	12.78

Table 4.11: Comparison of result for All Frame segment with varying Loss Weights ratio as hyper parameter

Loss Weights ratio	All Frame Per Segment	
1:1	WER	82.76
	BLEU-4	13.04
5:1	WER	72.4
	BLEU-4	13.22
10:1	WER	61.04
	BLEU-4	13.72
20:1	WER	72.4
	BLEU-4	***14.11***

Table 4.12: Comparison of result for Segment size 12 with varying Loss Weights ratio as hyper parameter

Loss Weights ratio	12 Frame Per Segment	
1:1	WER	67.87
	BLEU-4	12.86
10:1	WER	62.88
	BLEU-4	12.91
20:1	WER	66.32
	BLEU-4	12.07

As seen in Table 4.12, the highest BLEU-4 score and the lowest WER is achieved for Loss Weight ratio of 10:1. We can observe that as lamda recognition parameter increases the WER decreases resulting better gloss generation thus higher BLEU-4 score. However this trend can be observed until lambda recognition value of 10.

For all the segment sizes of 8, 12 and all segment the same trend can be observed for respectively for table 4.13 and 4.14. Thus we can conclude that for lambda recognition value of 10 the lowest WER and highest BLEU-4 score can be observed.

For our next experiment we varied the Number of Keypoints and Segment size to get an understanding of any effect due to their count.

As seen in Table 4.13 we can see that the best result can be obtained when using 8 frames and 27 keypoints, thus giving the highest BLEU-4 score and lowest WER. It

Table 4.13: Comparison of result for varying Segment size and Number of Keypoints

Data Streams	Metric	
6 Frame - 27 Joints	WER	61.07
	BLEU-4	13.71
8 Frame - 27 Joints	***WER***	***56.47***
	BLEU-4	***14.07***
12 Frame - 27 Joints	WER	67.87
	BLEU-4	12.87
8 Frame - 133 Joints	WER	79.72
	BLEU-4	10.25

gives better result for 27 number of keypoints compared to 133 number of keypoints because 133 number of keypoints are more prone to occulusion and thus generating more noise compared to 27 number of keypoints.

On the other hand, 8 frame performs the best compared to 6 and 12 segment size because, for 6 frames the segment size is too low to consist a single gloss on average. Again, for 12 frames the segment size is too big thus can consist of more than one gloss, which can lead to erroneous gloss detection, thus decreasing the BLEU-4 score. However for 8 frame the segment size is better to be able to consist one sign on an average thus leading to higher BLEU-4 score and lower WER.

Finally we tried different channels from SAM-SLR [20] to see which channel works the best.

Table 4.14: Comparison of result for Three Data streams against 3DCNN

Data Streams	Metric	
JOINT	***WER***	***56.47***
	BLEU-4	***14.07***
BONE	WER	62.5
	BLEU-4	12.64
JOINT MOTION	WER	72.94
	BLEU-4	9.95
R2+1D (SAM-SLR)	WER	63.28
	BLEU-4	11.97

We can observe in Table 4.14 that compared to all data streams the joint data stream gives the best result, this is because the bone can give deteriorating result due to high distance between nodes and for joint motion the consecutive frames do not possess significant change to contribute to the result. Moreover, we can also observe that SLGCN extracts better higher level features than 3D CNN. Thus SL GCN joint data stream gives highest BLEU-4 score and lowest WER.

4.4.3 Ablation Study

At first we tried to see the effect of a Pretrained BERT Language Model as Encoder against a Normal Transformer Encoder. Our results are as follows.

Table 4.15: Comparison of result for Normal Transformer Encoder againstPretrained BERT Encoder

8 Frame Per Segment		
Normal Transformer Encoder	WER	58.79
	BLEU-4	13.38
Pretrained BERT Language Model as Encoder	WER	61.36
	BLEU-4	14.07

Inspired by [56], we tried to see whether tweaking around the transformer model improves the overall performance of our pipeline. As shown in the table, improvement occurs but not as much as expected, when we replace the encoder part with a pretrained BERT.

Next we tried to see if discarding the SL-GCN as a whole had any significant effect.

Table 4.16: Comparison of result for SL-GCN against Keypoint features

SLGCN		
METRIC	WITH	WITHOUT
WER	***56.47***	57.41
BLEU-4	***14.07***	12.98

So we tried to do an inquiry as to how our pipeline performs without the SL-GCN module. So we generated results by feeding the keypoint estimations directly to the

transformer. As shown in the table, SL-GCN does indeed improve performance, but not so much.

4.4.4 Qualitative Result

Table 4.17 shows some of the translations generated by our pipeline. The first one shows a comparatively better translation, in which the context is captured well, even though the model generated an extra expression. On the other hand, the second translation performed poorly. It is illegible and fails to generate some important proper nouns. This trend is actually seen throughout other translations of our model as well.

Table 4.17: Spoken language translations produced by our Model

Ground Truth:	Fog will form here and there
Model Hypothesis:	In the south, fog will form here and there
Remarks:	Generated an extra expression
Ground Truth:	Tonight temperatures from minus five degrees to plus three degrees in the northeast of the rhine
Model Hypothesis:	Tonight fifteen degrees by the sea five degrees in the north and in the eifel
Remarks:	Failed to generate correct numbers, name of the place, direction

4.4.5 Comparison against Other Models

Table 4.18: Comparison of BLEU-4 Results with Other Models

	BLEU-4
NSLT [1]	18.4
SL Transformer [31]	22.38
STMC Transformer [57]	22.23
TSPNet [45]	13.41
Our Model	***14.07***

It can be seen from the table that, NSLT [1], Sign Language Transformer [31] and STMC Transformer [57] perform better than our model. On the other hand, our model gives slightly better performance than TSPNet [45]. We assume the following points regarding this matter:

- SL Transformer [31] uses CNNs pretrained on datasets consisting of sequence of signs, resulting in a more fine-tuned model to sequential data.
- NSLT [1] has an end to end system which enables better fine-tuning of their model while generating translations.
- STMC-Transformer [57] decomposes the input into spatial features of different cues - face, palms, full-frame and pose. Then, a temporal multi-cue module processes those features, without any explicit segmentation. This leads to better performance.
- Neither of [1, 31] also performs any fixed segmentation of frames, leading to better results.

Chapter 5

Conclusion

In conclusion, we utilize a better keypoint estimation model to build a skeletal-based pipeline for Sign Language Translation to generate meaningful spoken sentences from sign videos. We take the help of Graph Convolutional Networks for this case. Some of our findings while conducting this work are given below as part of our concluding remarks. Our model could surpass performance of one of our studied models, TSP-Net [45] by a small margin, with segment size of 8 and 33 keypoints. Our pipeline generated better meaningful translations under these constraints.

5.1 Future Work

Our approach of using feature extractors, pretrained on single-sign dataset along with fixed size segmentation of frames performed unsatisfactorily against state-of-the-art models, which generally trained their pipelines on continuous datasets in an end-to-end way, which made their model more robust. One future direction to improve the current performance of our work may be to introduce an end-to-end translation system instead of a multi-stage pipeline. This will facilitate better fine-tuning of the visual feature extractor resulting in better feature generation. More robust features will clear the way for temporal modules like transformers to generate more meaningful translations of signs. Another approach for improvement can be to explore better ways of segmentation of frames. Choosing segment size dynamically may perform better than

fixed-size segmentation. Finally, another direction of work can be done to discard the segmentation of frames altogether and work with all frames of input.

REFERENCES

[1] N. C. Camgoz, S. Hadfield, O. Koller, H. Ney, and R. Bowden, "Neural sign language translation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7784–7793.

[2] O. M. Sincan and H. Y. Keles, "Autsl: A large scale multi-modal turkish sign language dataset and baseline methods," *IEEE Access*, vol. 8, pp. 181 340–181 355, 2020.

[3] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5693–5703.

[4] F. Zhang, X. Zhu, H. Dai, M. Ye, and C. Zhu, "Distribution-aware coordinate representation for human pose estimation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 7093–7102.

[5] N. M. Adaloglou, T. Chatzis, I. Papastratis, A. Stergioulas, G. T. Papadopoulos, V. Zacharopoulou, G. Xydopoulos, K. Antzakas, D. Papazachariou, and P. none Daras, "A comprehensive study on deep learning-based methods for sign language recognition," *IEEE Transactions on Multimedia*, 2021.

[6] U. Farooq, M. S. M. Rahim, N. Sabir, A. Hussain, and A. Abid, "Advances in machine translation for sign language: approaches, limitations, and challenges," *Neural Computing and Applications*, vol. 33, no. 21, pp. 14 357–14 399, 2021.

[7] P. Buehler, A. Zisserman, and M. Everingham, "Learning sign language by watching tv (using weakly aligned subtitles)," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 2961–2968.

[8] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016, pp. 4724–4732.

[9] J. Charles, T. Pfister, M. Everingham, and A. Zisserman, "Automatic and efficient human pose estimation for sign language videos," *International Journal of Computer Vision*, vol. 110, no. 1, pp. 70–90, 2014.

[10] Z. Yang, Z. Shi, X. Shen, and Y.-W. Tai, "Sf-net: Structured feature network for continuous sign language recognition," *arXiv preprint arXiv:1908.01341*, 2019.

[11] T. Ahmad, L. Jin, X. Zhang, S. Lai, G. Tang, and L. Lin, "Graph convolutional neural network for human action recognition: A comprehensive survey," *IEEE Transactions on Artificial Intelligence*, vol. 2, no. 2, pp. 128–145, 2021.

[12] S. Li and W. Deng, "Deep facial expression recognition: A survey," *IEEE transactions on affective computing*, 2020.

[13] P. Molchanov, S. Gupta, K. Kim, and J. Kautz, "Hand gesture recognition with 3d convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2015, pp. 1–7.

[14] "Evalk - crunchbase company profile amp; funding." [Online]. Available: https://www.crunchbase.com/organization/evalk

[15] "Motionsavvy - crunchbase company profile funding." [Online]. Available: https://www.crunchbase.com/organization/motionsavvy-llc

[16] J. Stemper, "Motionsavvy uni: 1st sign language to voice system," Oct 2014. [Online]. Available: https://www.indiegogo.com/projects/motionsavvy-uni-1st-sign-language-to-voice-system/

[17] Q. Ye and T.-K. Kim, "Occlusion-aware hand pose estimation using hierarchical mixture density network," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 801–817.

[18] R. Yang, S. Sarkar, and B. Loeding, "Handling movement epenthesis and hand segmentation ambiguities in continuous sign language recognition using nested dynamic programming," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 3, pp. 462–477, 2009.

[19] H. Cooper, B. Holt, and R. Bowden, "Sign language recognition, chapter in visual analysis of humans: Looking at people," 2011.

[20] S. Jiang, B. Sun, L. Wang, Y. Bai, K. Li, and Y. Fu, "Skeleton aware multi-modal sign language recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3413–3423.

[21] I. Papastratis, K. Dimitropoulos, D. Konstantinidis, and P. Daras, "Continuous sign language recognition through cross-modal alignment of video and text embeddings in a joint-latent space," *IEEE Access*, vol. 8, pp. 91 170–91 180, 2020.

[22] K. Koishybay, M. Mukushev, and A. Sandygulova, "Continuous sign language recognition with iterative spatiotemporal fine-tuning," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 10 211–10 218.

[23] H. Cooper, E.-J. Ong, N. Pugeault, and R. Bowden, "Sign language recognition using sub-units," *Journal of Machine Learning Research*, vol. 13, pp. 2205–2231, 2012.

[24] 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.

[25] C. Vogler and D. Metaxas, "Parallel hidden markov models for american sign language recognition," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 1. IEEE, 1999, pp. 116–122.

[26] L. C. ROUGE, "A package for automatic evaluation of summaries," in *Proceedings of Workshop on Text Summarization of ACL, Spain*, 2004.

[27] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE transactions on acoustics, speech, and signal processing*, vol. 26, no. 1, pp. 43–49, 1978.

[28] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[29] 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," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3793–3802.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[31] N. C. Camgoz, O. Koller, S. Hadfield, and R. Bowden, "Sign language transformers: Joint end-to-end sign language recognition and translation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 023–10 033.

[32] S.-K. Ko, C. J. Kim, H. Jung, and C. Cho, "Neural sign language translation based on human keypoint estimation," *Applied Sciences*, vol. 9, no. 13, p. 2683, 2019.

[33] M. Vázquez-Enríquez, J. L. Alba-Castro, L. Docío-Fernández, and E. Rodríguez-Banga, "Isolated sign language recognition with multi-scale spatial-temporal graph convolutional networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3462–3471.

[34] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 369–376.

[35] L. Pigou, S. Dieleman, P.-J. Kindermans, and B. Schrauwen, "Sign language recognition using convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2014, pp. 572–578.

[36] A. Tunga, S. V. Nuthalapati, and J. Wachs, "Pose-based sign language recognition using gcn and bert," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 31–40.

[37] L. Zheng and B. Liang, "Sign language recognition using depth images," in *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE, 2016, pp. 1–6.

[38] H. Luqman, S. A. Mahmoud *et al.*, "Arabic sign language recognition using optical flow-based features and hmm," in *International Conference of Reliable Information and Communication Technology*. Springer, 2017, pp. 297–305.

[39] S.-K. Ko, J. G. Son, and H. Jung, "Sign language recognition with recurrent neural network using human keypoint detection," in *Proceedings of the 2018 conference on research in adaptive and convergent systems*, 2018, pp. 326–328.

[40] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7291–7299.

[41] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[42] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[44] H. Duan, Y. Zhao, K. Chen, D. Shao, D. Lin, and B. Dai, "Revisiting skeleton-based action recognition," *arXiv preprint arXiv:2104.13586*, 2021.

[45] D. Li, C. Xu, X. Yu, K. Zhang, B. Swift, H. Suominen, and H. Li, "Tspnet: Hierarchical feature learning via temporal semantic pyramid for sign language translation," *arXiv preprint arXiv:2010.05468*, 2020.

[46] N. Kalchbrenner and P. Blunsom, "Recurrent continuous translation models," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1700–1709.

[47] G. Neubig, "Neural machine translation and sequence-to-sequence models: A tutorial," *arXiv preprint arXiv:1703.01619*, 2017.

[48] A. Yin, Z. Zhao, J. Liu, W. Jin, M. Zhang, X. Zeng, and X. He, "Simulslt: End-to-end simultaneous sign language translation," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 4118–4127.

[49] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[50] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[51] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[52] J. Alammar, "The illustrated transformer." [Online]. Available: http://jalammar.github.io/illustrated-transformer/

[53] H. Zhou, W. Zhou, and H. Li, "Dynamic pseudo label decoding for continuous sign language recognition," in *2019 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2019, pp. 1282–1287.

[54] A. Hannun, "Sequence modeling with ctc," Jan 2020. [Online]. Available: https://distill.pub/2017/ctc/

[55] M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado *et al.*, "Google's multilingual neural machine translation system: Enabling zero-shot translation," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 339–351, 2017.

[56] M. De Coster, K. D'Oosterlinck, M. Pizurica, P. Rabaey, M. Van Herreweghe, J. Dambre, and S. Verlinden, "Frozen pretrained transformers for neural sign language translation," in *18th Biennial Machine Translation Summit*, 2021, pp. 88–97.

[57] K. Yin and J. Read, "Better sign language translation with stmc-transformer," in *Proceedings of the 28th International Conference on Computational Linguistics*, 2020, pp. 5975–5989.