AN ITERATIVE APPROACH TO KEYWORD SEARCH IN SIGN LANGUAGE

by

Mansur Yeşilbursa

B.S., Electrical and Electronics Engineering, Ihsan Dogramaci Bilkent University, 2019

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical and Electronics Engineering
Boğaziçi University
2022

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor Prof. Murat Saraçlar for his invaluable advice and patience during my MSc study. His passion and dedication to his job inspired me and kept me motivated during challenging times. Needless to say, this thesis would not be possible without his immense knowledge and guidance. I would like to extend my sincere thanks to Prof. Lale Akarun and Prof. Engin Erzin for their valuable contributions.

I would like to also thank my family and friends. Their belief in me kept me going during times when I doubted myself. Lastly, I am very grateful to my cat for her companionship.

The work in this thesis was supported in part by the Scientific and Technological Research Council of Turkey (TUBITAK) under the program 2210-A.

ABSTRACT

AN ITERATIVE APPROACH TO KEYWORD SEARCH IN SIGN LANGUAGE

Sign languages are the main medium of communication for the Deaf. However, insufficient retrieval tools for sign languages restrict the Deaf's access to information. To address this issue, we tackle the problem of keyword search in sign language. Although keyword search is a well-studied task for domains like speech processing, it has not been extensively studied in the context of sign language. To this end, we introduce improvements to an existing keyword search system for sign language and a new iterative training approach. We adapt Graph Attention Networks (GAT) to the sign language domain and extend its capabilities by employing a learnable mask and a separate temporal attention mechanism. Moreover, we investigate the effectiveness of the Pseudo-Relevance Feedback (PRF) technique in improving retrieval accuracy. Additionally, it is demonstrated that the existing model can also be trained with similarity-based methods using cosine and triplet losses, which can later be fused with other models to boost performance. Finally, we introduce an iterative training method similar to Expectation-Maximization (EM) that gradually improves its predictions. This method employs a cross-modal attention mechanism and a query encoder to discover subtle video-query interactions. The experiments are carried out on the RWTH-Phoenix2014T dataset, where the effectiveness of the proposed methods is verified. The results show that the pose models trained with a GAT-based encoder and in an iterative way significantly improve the retrieval performance.

ÖZET

İŞARET DİLİNDE ANAHTAR SÖZCÜK ARAMAYA YİNELEMELİ BİR YAKLAŞIM

İşaret dilleri, Sağırlar için ana iletişim aracıdır. Ancak, Sağırların bilgiye erişimi için gerekli geri getirme sistemleri henüz geliştirilmemiştir. Bu sorunun çözümüne yönelik, bu tezde işaret dilinde anahtar sözcük arama sorunu ele alınmıştır. Anahtar sözcük arama, konuşma işleme gibi alanlar için detaylıca çalışılmış bir problem olsa da, işaret dili bağlamında kapsamlı bir şekilde çalışılmamıştır. Bu tezde, mevcut bir işaret dilinde anahtar sözcük arama sisteminde yapılan iyileştirmelerin yanı sıra yeni bir yinelemeli eğitim yaklaşımı önerilmiştir. Çizge Dikkatli Sinir Ağları (GAT) işaret dili alanına uyarlanıp, öğrenilebilir maske ve ayrık bir zamansal dikkat mekanizması kullanılarak geliştirilmiştir. Ayrıca, Sözde-İlişiklik Geri Bildirimi (PRF) tekniğinin geri getirme performansına olan etkisi incelenmiştir. Bunun yanı sıra, mevcut modelin benzerliğe dayalı yöntemlerle, kosinüs ve üçüz kayıpları kullanılarak eğitilebileceği ve daha sonra performansı artırmak için diğer modellerle birleştirilebileceği gösterilmiştir. Son olarak, tahminlerini kademeli olarak iyileştiren Beklenti-Enbüyütme (EM) tekniğine benzer yinelemeli bir eğitim yöntemi önerilmiştir. Bu yöntem, incelikli video-sorgu etkileşimlerini keşfetmek için bir sorgu kodlayıcının yanı sıra medyumlar arası bir dikkat mekanizması kullanır. Deneyler, RWTH-Phoenix2014T veri kümesi üzerinde gerçekleştirilmiş olup önerilen yöntemlerin başarımı gösterilmiştir. Sonuçlar, poz modellerinin GAT tabanlı kodlayıcılarla, yinelemeli bir şekilde eğitildiğinde geri getirme başarımını önemli derecede iyileştiğini göstermiştir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . iii
ABSTRACT . . . iv
ÖZET . . . v
LIST OF FIGURES . . . ix
LIST OF TABLES . . . x
LIST OF SYMBOLS . . . xii
LIST OF ACRONYMS/ABBREVIATIONS . . . xiv
1. INTRODUCTION . . . 1
2. RELATED WORK . . . 5
2.1. Sign Language Recognition . . . 5
2.1.1. Feature Extraction . . . 5
2.1.2. Classification . . . 7
2.2. Sign Language Translation . . . 9
2.3. Keyword Search . . . 10
2.4. Sign Spotting . . . 12
3. METHODOLOGY . . . 13
3.1. Feature Extraction . . . 13
3.1.1. Pose Feature Extraction with OpenPose . . . 13
3.1.2. Handshape Feature Extraction with DeepHand CNN . . . 14
3.1.3. Handshape Feature Extraction with MultiTask CNN . . . 14
3.2. Baseline System . . . 14
3.2.1. Encoder Architectures . . . 14
3.2.1.1. Graph Encoder . . . 15
3.2.1.2. Vector Encoder . . . 16
3.2.2. Keyword Search Module . . . 17
3.2.2.1. Query Embeddings . . . 17
3.2.2.2. Attention-based Selection Mechanism . . . 17
3.2.3. Fusion Strategy . . . 18

3.3. Pre-Trained Word Embeddings . . . 18
3.3.1. Word2Vec Embeddings . . . 18
3.3.2. FastText Embeddings . . . 19
3.4. Improving Graph Encoder with Graph Attention Networks . . . 19
3.4.1. Graph Attentional Layer . . . 20
3.4.2. Spatial-Temporal Graph Attention Networks . . . 21
3.5. Pseudo-Relevance Feedback . . . 23
3.5.1. Mathematical Background . . . 23
3.5.2. Pseudo-Relevance Feedback for SL-KWS . . . 24
3.5.2.1. Query-Specific Distance Metric Learning . . . 24
3.5.2.2. Score Normalization and Learning Mixing Coefficient . 25
3.6. Similarity-Based SL-KWS . . . 27
3.6.1. Cosine Similarity Loss . . . 27
3.6.2. Triplet Loss . . . 28
3.7. Iterative SL-KWS . . . 29
3.7.1. Query Encoder . . . 30
3.7.2. Cross-Modal Attention Layer . . . 30
3.7.3. Training Strategy . . . 31
3.7.4. Inference . . . 32
4. EXPERIMENTS AND RESULTS . . . 33
4.1. Experimental Setup . . . 33
4.1.1. Dataset . . . 33
4.1.2. Evaluation Metrics . . . 34
4.1.2.1. Mean Average Precision (mAP) . . . 34
4.1.2.2. Precision at N (p@N) . . . 34
4.1.3. Implementation Details . . . 34
4.2. Improved Graph Encoder Results . . . 35
4.2.1. Comparison of Different Graph Encoder Architectures . . . 35
4.2.2. Analysis of the Learned Mask . . . 36
4.2.2.1. Investigating the Most Significant Nodes . . . 36
4.2.2.2. Investigating the Least Significant Nodes . . . 38

4.3. Pseudo-Relevance Feedback Results 39
4.4. Similarity-based Models 40
4.5. Iterative SL-KWS Results 42
4.5.1. Iterative and Default Training Methods 43
4.5.2. Prediction Refinement 43
4.6. Cross-Lingual Search 45
4.7. Fusion Results 46
5. CONCLUSION 47
REFERENCES 49

LIST OF FIGURES

Figure 3.1. General pipeline of the baseline system. 15

Figure 3.2. Spatial-temporal graph. 16

Figure 3.3. Handshape encoder architecture. 16

Figure 3.4. Graph encoder comprised of ST-GCN and ST-GAT layers. 23

Figure 3.5. Pseudo-Relevance Feedback Algorithm 26

Figure 3.6. Cosine loss configuration. Green represents positive and red represents negative examples. PC: positive context. PQ: positive query. NC: negative context. NQ: negative query. 28

Figure 3.7. Triplet loss configuration. Green represents positive, red represents negative and gray represents anchor examples. PC: positive context. PQ: positive query. NC: negative context. NQ: negative query. 29

Figure 3.8. Iterative SL-KWS pipeline. . . . . .

Figure 4.1. Handshape A, Handshape B and OpenPose hand keypoints layout. 38

Figure 4.2. Improvement over iterations. 44

LIST OF TABLES

Table 4.1. The effect of each modification. All the models are trained with OpenPose features. Temporal Att. stands for temporal attention. '+' means included and '-' means excluded from the encoder. . . . 36

Table 4.2. The nodes with the most contribution and their corresponding scores. RH stands for right hand. Numbers within parantheses denote the index of the joint per Figure 4.1(c). 37

Table 4.3. Number of times a node is the most significant node for another. RH stands for right hand. 38

Table 4.4. The nodes with the least contribution and corresponding scores. Numbers within parantheses denote the index of the joint per Figure 4.1(c). 39

Table 4.5. PRF results. Scores are given in percentages. 40

Table 4.6. Results for cosine similarity-based models and their fusion with classification models. Rel. Impr. stands for relative improvement. . . . 41

Table 4.7. Results for triplet similarity-based models and their fusion with classification models. Rel. Impr. stands for relative improvement. . 42

Table 4.8. AP difference between similarity and classification models for some queries. 42

Table 4.9. Results for iterative and default training methods. Results are given as percentages. 43

Table 4.10. Prediction refinement progress. ST-GCN model is used for OpenPose features. 44

Table 4.11. Cross-lingual search results. Scores are given as percentages. The models marked with '*' are baseline models. 45

Table 4.12. Fusion of iteratively trained OpenPose and DeepHand models. ST-GCN + ST-GAT architecture is used for OpenPose models. . . .

LIST OF SYMBOLS

A	Self-connected adjacency matrix according to human anatomy
$\mathbf{A}$	Video-to-query attention
a	Anchor example *or* exponential weight parameter
$\vec{a}$	Weights of shared attention mechanism
$\mathbf{B}$	Query-to-video attention
c	Confidence score
$\mathbf{c}$	Context vector
D	Diagonal matrix *or* feature dimension of input nodes
D'	Feature dimension of an output node
$d(.,.)$	Distance metric
$D(P_Q, X)$	Distance between example X and pseudo-positive set P_Q
E	Set of edges in spatial temporal graph
E_s	Set of spatial edges in spatial temporal graph
E_t	Set of temporal edges in spatial temporal graph
e_{ij}	Excitation between node i and node j
$f(.,.)$	Score function
G	Spatial-temporal graph
$\mathbf{G}$	Output set of nodes
G_{in}	Input undirected graph
$\vec{g_i}$	Feature vector of an output node
$\mathbf{H}$	Input set of nodes
$\vec{h_i}$	Feature vector of an input node
l	Number of frames per query constant
l_2	Euclidian distance metric
M	Mahalanobis matrix
m	Tolerance margin
m_{ij}	Mask coefficient between node i to node j
$\mathcal{N}_i$	Neighborhood set of node i

n	Negative example
P_Q	Pseudo-positive set of examples for query Q
p	Positive example
p_i^Q	Initial predictions for query Q
p_{prf}^Q	PRF predictions for query Q
$\mathbf{Q}$	Sequence of query embeddings
$\mathbf{q}$	Query embedding
$\mathbf{S}$	Similarity matrix for query and video sequence
$\mathbf{S_c}$	Similarity matrix after softmax applied along columns
$\mathbf{S_r}$	Similarity matrix after softmax applied along rows
$S(Q, X)$	Prediction score of example X for query Q
$SIM(P_Q, X)$	Similarity between example X and pseudo-positive set P_Q
$\mathbf{s_i}$	ith frame of the encoded video sequence
s_j	Significance score of node j computed from the learnable mask
V	Set of vertices in spatial temporal graph
$\mathbf{V}$	Sequence of encoded video features
$\mathbf{W}$	Weight matrix
$\mathbf{W}_H$	A linear transformation in cross-modal attention layer
$\mathbf{W_S}$	A linear transformation in cross-modal attention layer
X_Q	A pseudo-positive example for query Q
x	x-axis coordinate
y	y-axis coordinate *or* similarity label
α_{ij}	Attention coefficients
β	Affine scale parameter
β_{kt}	Separated temporal attention coefficients
γ	Mixing coefficient
Δ	Temporal kernel size
θ	Affine bias parameter
σ	Non-linear activation function

LIST OF ACRONYMS/ABBREVIATIONS

1D-CNN	One Dimensional Convolutional Neural Network
AP	Average Precision
ASR	Automatic Speech Recognition
BN	Batch Normalization
C3D	3D CNN-based Action Recognition Network
CHMM	Coupled Hidden Markov Model
CNN	Convolutional Neural Network
CSLR	Continuous Sign Language Recognition
CTC	Connectionist Temporal Classification
DTW	Dynamic Time Warping
EM	Expectation-Maximization
EMG	Electromyography
GAN	Generative Adversarial Networks
GAT	Graph Attention Network
GCN	Graph Convolutional Network
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
HMM	Hidden Markov Model
I3D	Inflated 3D Convolutional Network
ISLR	Isolated Sign Language Recognition
KWS	Keyword Search
LDA	Latent Discriminant Analysis
LHMM	Linked Hidden Markov Model
LR	Learning Rate
LSTM	Long-Short Term Memory
MS-G3D	Multi Scale Spatial-Temporal Graph Convolution
mAP	mean Average Precision
NCA	Neighborhood Component Analysis

NMT	Neural Machine Translation
OOV	Out of Vocabulary
PCA	Principal Component Analysis
PE	Pose Estimation
PHMM	Parametric Hidden Markov Model
PRF	Pseudo-Relevance Feedback
RELU	Rectified Linear Unit
RNN	Recurrent Neural Network
SIFT	Shift-Invariant Feature Transform
SL	Sign Language
SL-GCN	Sign Language Graph Convolutional Network
SL-KWS	Keyword Search in Sign Language
SLR	Sign Language Recognition
SLT	Sign Langauge Translation
SSTCN	Separable Spatial-Temporal Convolution Network
ST-GAT	Spatial-Temporal Graph Attention Network
ST-GCN	Spatial-Temporal Graph Convolutional Network
SURF	Speeded Up Robust Feature
SVM	Support Vector Machine
TCN	Temporal Convolutional Network
US	United States
WPD-2	Two Dimensional Wavelet Packet Decomposition

1. INTRODUCTION

Sign languages are the primary mode of communication used by the Deaf community. They are developed through interactions among deaf people through a natural process, much like spoken languages. Therefore, they are natural languages with unique grammar, structure, and morphology. They are different from signed languages (e.g., Signed English), which were invented to convey exact spoken language content in a visual medium and usually share the same grammar as the spoken language [1].

Sign languages use hand movement, hand configuration, facial expressions, and body pose to convey meaning. Some signs might consist of only hand movement or hand shape, whereas others might also incorporate facial expressions and body pose. Similar to pitch, prosody, and intensity in spoken languages, the way the signs are performed (i.e., use of facial gestures, transitions between signs, and signing pace) can alter the tone and the feeling attached to the plain meaning [1]. Furthermore, there are sign language poems, much similar to written poems, that contort conventional meaning to excite new perspectives and feelings. Deaf poets might utilize repetitive hand movements and altered flow of the signs to create those effects. Therefore, one might expect to see a significant difference between formal and informal use of the language, pointing out the strong dependence on the context. Overall, it is clear that sign languages are as complex as spoken natural languages and require meticulous analysis whenever studied.

According to a survey [2], about 0.22% of the U.S population is "functionally deaf," and more than half is above 65 years old. It is also reported that as opposed to 82% of the hearing population between the age of 18 to 44, 58% of deaf or hard of hearing people in the same age group participates in the labor force [2]. Similarly, 12.8% of the hearing population graduates from college, but this number drops to 5.1% for deaf or hard of hearing people [2]. These statistics demonstrate that the Deaf community is still facing challenges in terms of joining the workforce, education, and

overall integration into the society in a developed country such as the U.S.. It is only fathomable that the gap is even wider in developing and underdeveloped countries. Since the percentage of deaf people in the population is relatively low, it is challenging to raise awareness that could impact social change at the moment. Nevertheless, the development of technology can alleviate a good portion of the problems faced by the Deaf. For instance, a precise sign language translation system can mitigate many daily issues for the Deaf and improve their integration into society. However, many sign language problems are still open research subjects due to its aforementioned complexity.

With the growing amount of information and content online, access to information retrieval systems is more critical than ever. However, a tremendous amount of information remains inaccessible for deaf people due to the lack of widely available sign language retrieval systems and low literacy in the Deaf community. Therefore, it is crucial to develop retrieval systems that facilitate access to sign language content. However, developing such systems is not trivial as they require advanced video processing techniques with high computational costs. Hence, sign language research in computer vision remained inert for a long time. The rise of GPUs and the following revival of deep learning has attracted more attention to sign language research, especially to sign language recognition (SLR) and sign language translation (SLT). Even though deep learning methods improve the state-of-the-art for both of these tasks by a large margin, they still need improvement in performance and computational cost before they become widely available on various edge devices. Along with SLR and SLT, new deep learning based retrieval systems for sign language have also been introduced in the last decade. However, it is still a relatively new field that demands much more effort from the computer vision community.

One of the main challenges for sign language research is the scarcity of publicly available general domain datasets. The majority of the datasets are collected from sign interpretations of television broadcasts. As a result, obtained collections usually only include formal use of sign language, leaving out informal use cases such as daily conversations and storytelling. Furthermore, the number of interpreters or native signers

is insufficient to label the vast amount of data required to train deep models. Due to information being conveyed on multiple channels, annotating all attributes of sign utterances is very expensive. On top of that, sign language research is usually underfunded, even in developed countries. All the reasons listed above contribute to the deceleration of the developments in the field.

In efforts to close the gap in the field, this thesis tackles keyword search for sign language problem. Keyword search is a well-established research subject. Numerous commercial systems are widely available, allowing easy and fast access to written or audio-visual content. The keyword search can be defined as the retrieval of documents related to a given query in a corpus. There are many different sub-definitions depending on what is recognized as related and the medium of the documents. In this thesis, we define the problem as retrieving sign utterances containing a given query. Queries are given in the text form, and we do not retrieve the temporal position of the query within the utterance. Queries can be sign glosses as well as cross-lingual words though our work mostly focuses on gloss search. The system is trained with weak supervision, thus, removing the need for strongly annotated datasets. The development of such a system can significantly improve the navigation experience in databases/internet for the Deaf. Using written queries instead of visual entries improves the accessibility of the system, allowing users to perform a search without a camera. Moreover, non-signers can also use the system to search in a sign language database by entering cross-lingual queries.

The work presented in this thesis is built upon the system proposed in [3–6]. The main contributions of this thesis can be listed as:

- We propose a Spatial-Temporal Graph Attention Network (ST-GAT) based graph encoder to replace/improve the baseline graph encoder [3]. We introduce novel improvements to ST-GAT architecture by employing a learnable neighborhood mask and a separate temporal attention mechanism to boost the temporal modeling capabilities further. Even though ST-GAT architecture was used for action recognition, no SL study employs ST-GAT architecture within our knowledge.

The effectiveness of ST-GAT is demonstrated with experiments.

- We adapt the Pseudo-Relevance Feedback (PRF) method, which is commonly used for text and speech retrieval, to our problem to increase retrieval accuracy. We experiment with query-specific distance metrics and standard distance metrics such as cosine distance and l_2 distance to compute PRF scores. In order to combine PRF scores with original predictions, we apply score normalization and learn a mixing coefficients.
- We introduce a new training method for the baseline models by employing similarity losses. The similarity-based models can be employed to boost the performance of handshape models via the fusion strategy introduced in [5]. The results are published at a national conference.
- Finally, as the main contribution of this thesis, an iterative approach to keyword search in sign language is proposed. By incorporating a query encoder and a cross-modality attention module into the system, an iterative training method similar to Expectation-Maximization (EM) is proposed. The effectiveness and prediction refinements introduced by the method are demonstrated.

The rest of the chapters are organized as follows:

- In Chapter 2, the related work in sign language recognition, sign language translation and keyword search are reviewed.
- In Chapter 3, the methods are introduced:
  - In Section 3.2, a summary of the baseline system is provided.
  - In Section 3.4, the improved graph encoder architecture is explained.
  - In Section 3.5, PRF algorithm is described.
  - In Section 3.6, similarity-based models are introduced.
  - In Section 3.7, an iterative training method is proposed.
- In Chapter 4, the experiments and results are reported.
- In Chapter 5, a conclusion is drawn, and possible future work is discussed.

2. RELATED WORK

In this chapter, we provide a summary of related works in sign language and keyword search research.

2.1. Sign Language Recognition

Sign language recognition (SLR) has been the primary subject of sign language research. It is studied under two main subtasks: isolated sign language recognition (ISLR) and continuous sign language recognition (CSLR). In ISLR, the aim is to classify the given video segment into predefined sign classes. Each video is known to contain one sign utterance. On the other hand, the goal of CSLR is to identify every sign in a continuous sign stream. There is no apriori information about the number of signs or temporal alignment of the signs. Additionally, co-articulation of the signs in a continuous stream introduces new challenging aspects to the problem. Despite its challenges, CSLR is more relevant to real-life scenarios and has been studied extensively in recent years.

In this section, we examine previous work in SLR. We analyze SLR as a two-step problem: i) feature extraction, ii) temporal modelling and classification.

2.1.1. Feature Extraction

There is no universally accepted fundamental subunit of sign languages such as phonemes in spoken languages. It is known that the signs may incorporate hand shape, movement, orientation, facial expressions, mouthing, and body pose. However, none of these channels can singlehandedly cover all signs (e.g., signs may share hand shape and movement but might be performed with different facial expressions). A speaker might also utilize different facial expressions and hand movements to emphasize some portions of the speech. However, these are used as auxiliary methods; they are not

essential. We cannot disregard other channels in sign languages as we do in spoken languages since they are integral to the meaning. However, it is safe to assume that among all those channels, hands are the most informative channel. Hence, they are always regarded as the primary channel of sign languages.

There are numerous ways to extract features from hands. Data gloves were prevalent in SLR and gesture recognition in early days [7]. These gloves are equipped with a gyroscope and accelerometer to collect hands' orientation, movement, and position information. Another sensor-based data acquisition method is Electromyography (EMG), which is used to track muscle activities in hand. Sensor-based systems often perform well in small sets of signs [7]. The primary advantage of sensor-based feature extraction is that they allow data acquisition with very little noise compared to vision-based approaches. However, the proposed systems usually do not apply to real life due to the inaccessibility of such sensors by the end-user.

Due to the limited accessibility of sensor-based systems, researchers have focused on vision-based systems. Features are extracted using shift-invariant feature transform (SIFT) and speeded up robust feature (SURF) methods [8, 9], principal component analysis (PCA), and linear discriminant analysis (LDA) are applied for feature selection and separation [9]. Other traditional machine learning techniques that are used for sign and gesture recognition include Convexity defects and K-curvature [10], frequency domain methods such as Fourier Descriptors (FD), 2-D Wavelet Packet Decomposition (WPD-2), and Discrete Wavelet Transform (DWT) [11,12]. Multiple feature extraction methods are often combined to achieve more robust features. Hu moments and SURF performed better when joined than used separately [13].

CNN-based models attained impressive results in many vision tasks in the last decade [14, 15]. They have become increasingly popular since they remove the need for manual features. CNNs can learn visual representations from raw images without any human intervention. They learn scale and rotation invariant representations when employed together with pooling layers. Proposed systems showed that CNN could

learn more powerful representations than manual features. Following this trend, many CNN-based models are used to extract features from hand pose estimation [16,17], and hand shape [18] classification pipelines. Even though 2D CNNs are effective tools to process still images, they are often insufficient to capture the temporal nature of the videos. Therefore, they are often used together with recurrent neural networks (RNN) such as long-short term memory (LSTM) and gated recurrent unit (GRU) to model temporal relationships [19]. Alternatively, 3D CNN models can be employed to extract features from videos [20, 21].

Other prevalent SL features are obtained from human pose estimation systems. Pose estimation (PE) aims to predict 2D or 3D positions of predefined human joints. They are frequently used in human-computer interactions, augmented and virtual reality systems and are essential to human action understanding [22]. For single-person 2D PE, Newell et al. introduced the stacked hourglass model consisting of CNN layers that work on different input scales [23]. Wei et al. proposed Convolutional Pose Machines that iteratively improve keypoint estimations using 2D belief maps from the previous iteration [24]. There are two main approaches in multi-person 2D PE: bottom-up and top-down approaches. In the top-down approach, human bounding boxes are estimated, then a single-person PE model is applied to each bounding box. In contrast, bottom-up models first estimate all the joint locations in the image and then associate joints with body parts and different people. Multi-person PE is practical when extracting features for conversational SLR that include multiple people in the same frame.

2.1.2. Classification

Before the emergence of deep neural networks, HMM was the primary method for temporal modeling and classification [25], similar to speech recognition. The main reason HMM was so widely-adopted for recognition tasks is that HMM can capture temporal relationships better than other traditional machine learning techniques. Cooper et al. use HMMs to classify combined sub-unit features obtained from vision and

tracking data [26]. Elmezain et al. employed Gaussian Mixture Model (GMM) for segmentation, and later on, segmented sequences were classified with HMM [27]. There are many variants of HMM to mitigate various issues. Coupled HMM (CHMM) and Linked HMM (LHMM) are developed to improve the scalability of HMM-based recognition systems [7]. Parametric HMMs (PHMM) incorporate global parametric variations in output probabilities to better robustness against noise in the input [28]. As an alternative to HMM, DTW was used for temporal alignment and classification of gesture sequences [11]. Finite State Machines (FSM)s were employed for gesture classification since they can learn the alignment of the data during training in contrast to static states of HMMs [29].

Although HMMs and other mentioned methods obtain significant results in dynamic gesture recognition and ISLR, they are often seen as insufficient to model complex temporal dynamics of CSLR [30]. Studies in the last decade have shown that Connectionist Temporal Classification (CTC) methods usually outperform HMM and DTW-based recognition systems. However, CTC-based systems introduce new issues as well. They are susceptible to overfitting, and while training end-to-end models, gradients from CTC loss may not be optimal for feature extraction modules [30]. Subsequently, various modifications to CTC loss were introduced to address the issues above [31, 32]. CTC with entropy regularization was proposed to distribute the error among alternative paths in a balanced way, preventing overconfident peaks [31]. Now, we will introduce current deep learning methods for SLR.

Koller et al. [33] proposed a CNN-HMM network for CSLR where hand features extracted with 2D CNN networks are classified with the help of HMM. They extend their work by incorporating LSTM layers to the feature extractor and using multiple data channels, including full-frame, hand, and mouth crops [34]. Cui et al. introduced 2D CNN-LSTM architecture that includes a regularization network for weakly supervised training [35]. Afterward, they proposed a module comprised of 1D CNN layers, connecting the feature extractor and LSTM layers [36]. 3D CNN-based networks are also employed for SLR to process depth images or model temporal segments of the

sequence. Pu et al. adapted a 3D CNN-based action recognition network (C3D) [37] to ISLR [20]. Later on, C3D architecture was utilized in a two-stream CSLR system combined with a hierarchical attention network [38]. Similarly, pre-trained I3D architecture [39] from the action recognition domain was applied to ISLR [40]. Other deep 3D CNN networks, such as 3D-ResNet architecture, were employed as feature extractor modules [41]. In order to reduce the complexity of LSTM layers, they also developed a dilated temporal convolutional network and trained the network with CTC loss in an iterative manner [41].

In addition to RGB-based models, there are also skeleton-based SLR systems that use pose estimation outputs as the main features. Spatial-temporal graph convolutional networks (ST-GCN) were initially proposed for action recognition [42] and later on adapted to SLR [43]. Numerous versions of ST-GCN are developed for SLR, including attention-enhanced GCN (AEGCN) [44], sign language GCN (SL-GCN), and separable Spatial-Temporal Convolution Network (SSTCN) [45] to work on pose features. Parelli et al. introduced a CSLR framework trained with Guided CTC [46]. The proposed method combines ST-GCN with BiLSTM networks to capture short and long-term dynamics [46]. Enriquez et al. adopted the MS-G3D network [47], enabling multi-scale graph operations and skip connections for direct information flow across layers to the ISLR task [48].

2.2. Sign Language Translation

Sign language translation (SLT) can be defined in two ways: i) converting written language to sign language video, conveying the same meaning in a target sign language, and ii) generating written translations in the target spoken language from sign language videos. The former task includes producing high-quality videos from written text which is a challenging task even for spoken languages and existing language models for them. Stoll et al. translate spoken language data to gloss level annotations via an encoder-decoder network, then the model learns a mapping between glosses and their corresponding skeleton representation [49]. Sign language production is only performed

on an avatar based on skeleton data with the help of generative adversarial networks (GAN) [49]. Afterward, they extend their work by eliminating the gloss recognition step and directly associating written text with pose representations [50]. This task is still very much dependent on the ability of generative networks to produce high-quality videos from skeleton representations.

The latter problem is more attainable and relevant in the current literature. Camgoz et al. proposed an attention-based encoder-decoder architecture inspired from neural machine translation (NMT) literature [51]. They experiment with three settings: i) translating gloss ground truths to spoken language, ii) directly converting sign language videos to spoken language, iii) performing CSLR as an intermediate step, then converting recognition predictions to spoken language. As expected, the first scenario results in better translation outputs. Comparing the other two settings shows that obtaining intermediate gloss representations from the sign video significantly surpasses the performance of direct conversion from sign videos. This is mainly due to the insufficiency of RNNs to model long-term relationships, as the number of frames in a sign video is much higher than the number of glosses. However, recognizing glosses as an intermediate step creates an undesired information bottleneck. The authors adopted the popular transformer architecture [52] to mitigate the bottleneck, which significantly improved NMT performance for spoken languages partly due to its superior ability to recognize longer dependencies than the previous RNN-based NMT systems. The transformer architecture trained jointly and end-to-end to simultaneously recognize glosses in the sign sequence while translating sign videos to spoken language [52].

2.3. Keyword Search

Keyword search is a well-established research area, especially for text and speech data. Written content retrieval has been the backbone of the internet since the rise of search engines. As keyword search for written data mainly depends on string matching and deterministic algorithms, we will focus on keyword search in spoken content in this chapter.

Spoken content retrieval can be achieved by applying text retrieval techniques to the output of automatic speech recognition (ASR) systems such as lattice search [53]. The systems based on this principle can attain very high performance given that the underlying ASR module is accurate enough [54]. However, such ASR systems can only be trained in the presence of vast annotated corpora, which is available for a few languages in the world. Therefore, researchers focused on other methods that do not rely on the performance of an ASR system. One of such approaches is to match speech segments at the acoustic level. The most popular method for this approach is DTW. Anguera et al. used subsequence DTW to match speech segments in every scale, alleviating the problems related to significant differences in the duration of queries and document segments [55]. Acoustic subunit matching techniques work well for out-of-vocabulary (OOV) queries since there is a direct correlation between an utterance of a word and its written form. However, it is not feasible to apply such techniques to SL-KWS as there is no relationship between a written query and its sign articulation.

Even though DTW is a very effective tool, it may fail to capture high-level linguistic features. Hence, model-based approaches are developed to incorporate high-level semantic information in the search. Most work in model-based search consists of three steps: i) segmentation, ii) clustering and iii) model training. Segmented speech units are clustered together, and patterns are learned for each cluster [54]. After learning acoustic patterns, DTW-based matching can be applied to the posteriorgrams [56]. Furthermore, ASR-based and DTW-based methods can be fused together to improve retrieval accuracy, especially for out-of-vocabulary (OOV) queries [57].

The recent DL-based keyword spotting (closely related to keyword search) systems consist of three main components: i) feature extractor, ii) DL-based acoustic modeling, and iii) posterior processing [58]. Different architectures are employed for acoustic modeling, fully connected layers [59], RNN variants for their temporal modeling capability [60], and CNNs for their low computational complexity [61]. RNN models are usually trained with CTC loss to generate posterior probabilities [60]. The conditional independence assumption of CTC is not a realistic approach for speech

signals. Thereof, sequence-to-sequence (Seq2Seq) [62] models used in NMT attracted KWS researchers. He et al. proposed an end-to-end Seq2Seq model that jointly learns acoustic and language models [63]. This model predicts subword units, eliminating OOV related issues. DL-based models drastically improved the state-of-the-art keyword spotting and search in terms of accuracy and computation complexity.

2.4. Sign Spotting

The objective of sign spotting is to find and localize query signs in a continuous sign stream. It differs from CSLR as it is not aimed to recognize every sign in the sequence but only a pre-defined set of queries. Viitaniemi et al. proposed a DTW-based sequence matching method on skin distribution histograms [64]. Hierarchical Sequential Patterns were used to spot signs in a continuous stream. However, these methods rely on strong gloss-level annotations, usually unavailable for large datasets. Hence, the systems trained with weak supervision have been proposed. Buehler et al. proposed a method that groups signs per visual feature distance using Multiple Instance Learning (MIL) framework by only using cross-lingual subtitles [65].

Similarly, a multiple instance SVM classifier was employed to spot signs by incorporating mouthings [66]. Recently, Momeni et al. introduced a MIL-based sign spotting method [67]. This method utilizes automatic mouthing annotations as strong supervision on top of weak cues collected from subtitles. The model is trained with a noise contrastive loss by computing the cosine distance between candidate windows and sign entries in a dictionary.

Sign spotting is very similar to keyword search in sign language definition put forward beforehand. The main difference is that keyword search does not aim to localize query signs temporally but instead decides on their presence in an utterance.

3. METHODOLOGY

In this chapter, we introduce the novel contributions of this thesis. We first introduce the features that are used in both baseline and the other proposed models. Then we provide a summary of the baseline system introduced in [3–5] and then propose improvements upon the existing system. Afterward, we introduce an iterative approach to keyword search in sign language (SL-KWS).

3.1. Feature Extraction

Two different types of features are extracted, handshape and pose features. Pose features are obtained using OpenPose framework [68] and hand shape features are extracted from the intermediate layers of deep CNN networks; MultiTask CNN [69] and DeepHand CNN [18]. Both baseline and the models introduced in this thesis are trained with these features.

3.1.1. Pose Feature Extraction with OpenPose

OpenPose is an open-source human pose estimation framework [68]. The model uses confidence maps to estimate keypoint positions and part affinity maps to associate keypoints with body parts, and the people in the image [68]. The framework outputs 21 keypoints for each hand and 25 keypoints for the body, including general points of interest in the face and the feet. The keypoints below the waistline are eliminated since they are not informative for SL. For each keypoint, (x, y) coordinates are provided along with confidence score c, which is a number between 0 and 1. The confidence score is incorporated as a feature since it informs the model about the accuracy of the keypoint.

3.1.2. Handshape Feature Extraction with DeepHand CNN

DeepHand CNN is trained with multiple SL corpora to classify each hand crop into one of 60 pre-defined handshape classes or a junk class [18]. The model is trained in a weakly-supervised way, iteratively improving temporal alignment via HMM to label sequences. 1024-dimensional features are taken from the second-last layer of the network.

3.1.3. Handshape Feature Extraction with MultiTask CNN

MultiTask CNN model was developed as a tokenization layer to be used before encoder-decoder-based SLT networks [69]. The model is trained with multiple SL corpora. One of them is a small but strongly annotated dataset that includes hand context information along with handshape labels [70]. The model learns to classify handshapes and identify hand context simultaneously [69]. 2048-dimensional features are taken from shared layers of the network.

3.2. Baseline System

The baseline system was introduced in [3] and extended in [4] by adding cross-lingual search capabilities to the system. Lastly, hand-shape features were incorporated on top of pose-based features to improve fusion performance [5]. In this section, we introduce the components of the baseline system. The flowchart of the baseline system is given in Figure 3.1.

3.2.1. Encoder Architectures

There are two types of encoder architectures used in the baseline model. When pose features are utilized, an ST-GCN-based encoder is employed. Handshape features are encoded with a 1D CNN network.

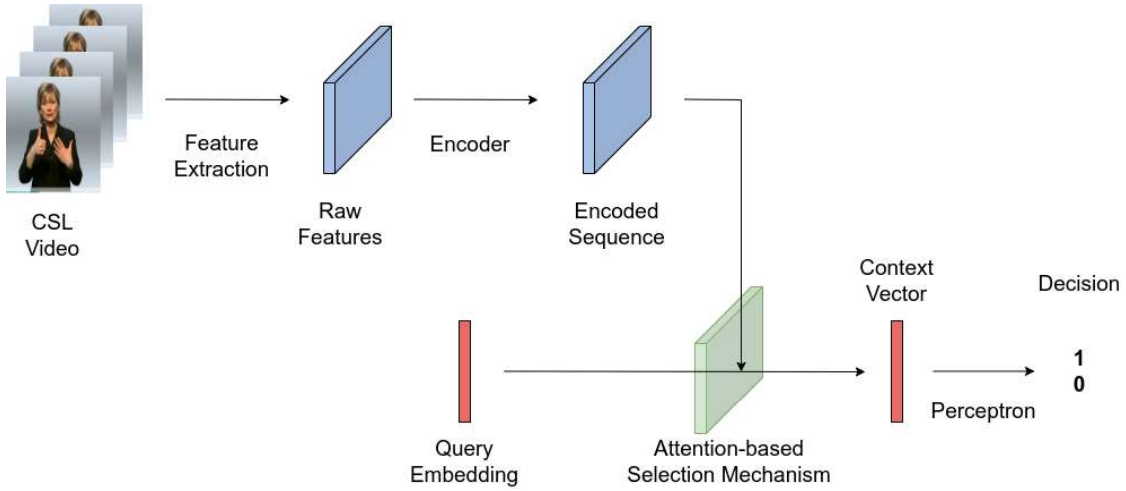


Figure 3.1. General pipeline of the baseline system.

3.2.1.1. Graph Encoder. The human skeleton can be seen as an undirected graph if we assume joints as nodes and bones as edges. Each joint's connectivity can be expressed with an adjacency matrix. Since pose features contain 2D spatial positions of the joints, they can be processed as graphs. Even though vanilla CNN networks can be trained with pose features as they could be expressed in the grid form, the underlying structure can be learned more effectively if a graph-based network is utilized. For this purpose, ST-GCN architecture was proposed for action recognition [42].

Pose features are expressed as a spatial-temporal graph. Joints represent nodes in the graph. There are two types of edges: i) spatial edges that represent anatomical bone connections between joints, and ii) temporal edges that connect the same node across time steps. This way, the whole sequence is collected into a single graph. Figure 3.2 illustrates an instance of the spatial-temporal graph.

ST-GCN layers apply graph convolutions in spatial and temporal directions. Graph convolution is defined as

$$G_{out} = \sigma(D^{-\frac{1}{2}}AD^{-\frac{1}{2}}G_{in}W), \tag{3.1}$$

where A is the self-connected adjacency matrix, G_{in} is the input undirected graph, D is a diagonal matrix computed as $D = \sum_j A_{ij}$, W is the learnable weight matrix and σ is a non-linear activation function.

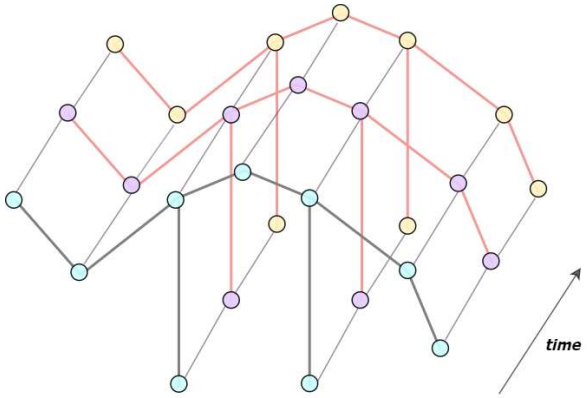


Figure 3.2. Spatial-temporal graph.

The encoder network consists of 12 ST-GCN layers with a temporal kernel size of three. Between each layer, batch normalization and ReLU activation functions are applied. In order to represent each encoded frame with a single vector, an average pooling layer is employed in the keypoint dimension.

3.2.1.2. Vector Encoder. Whenever handshape features are used, it is convenient to utilize CNN layers since handshape features represent each frame with a fixed dimensional vector. In order to capture the relationship between consecutive frames, temporal modeling is required. Therefore, 1D-CNN architecture was chosen as they discover temporal relationships and computationally inexpensive [5]. The length of the sequence is preserved using padding. No temporal pooling is applied. The architecture of the vector encoder is illustrated in Figure 3.3.

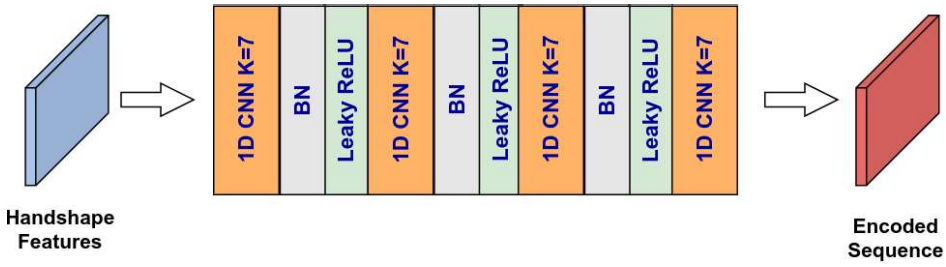


Figure 3.3. Handshape encoder architecture.

3.2.2. Keyword Search Module

The keyword search module is the component responsible for deciding on the input query's presence. It takes the encoded sequence as input and combines it with the query embedding such that the model can be trained end-to-end.

3.2.2.1. Query Embeddings. The system admits queries in the text form. The network learns embeddings from scratch for each query. This way, the selection mechanism can recognize the relationship between queries and the sequence frames.

3.2.2.2. Attention-based Selection Mechanism. The selection mechanism enables end-to-end system training by combining encoded sequence and query embeddings using the attention technique [3]. The relevant parts of the sequence with respect to the query are detected via attention. The scoring function $f(\mathbf{q}, \mathbf{s_i})$ calculates a score for the relationship between query $\mathbf{q}$ and ith frame of the sequence $\mathbf{s_i}$. Squared cosine similarity with learnable affine parameters (β, θ) is selected as the scoring function

$$f(\mathbf{q}, \mathbf{s}_i) = \beta \left[\frac{\mathbf{q} \cdot \mathbf{s}_i}{\|\mathbf{q}\| \cdot \|\mathbf{s}_i\|} \right]^2 + \theta. \tag{3.2}$$

The score between $\mathbf{q}$ and all sequence frames is normalized with the softmax function to obtain attention coefficients. Afterward, attention coefficients are used to compute the sequence's weighted average so that the sequence's relevant parts are emphasized. The resulting vector is called a context vector. For each query in the vocabulary, a context vector $\mathbf{c}$ is computed as

$$\mathbf{c} = \sum_i \left[\frac{\exp\left(f(\mathbf{q}, \mathbf{s}_i)\right)}{\sum_{i'} \exp(f(\mathbf{q}, \mathbf{s}_{i'}))} \right] \cdot \mathbf{s}_i. \tag{3.3}$$

Obtained context vectors are fed to a single layer perceptron with sigmoid activation to decide on the presence of the query in the video. The perceptron classifies every context vector corresponding to queries in the vocabulary for a single video. The system is trained with binary cross-entropy loss.

3.2.3. Fusion Strategy

As explained above, two different feature types are used; pose features and handshape features. Models trained with different feature types might show varying performance depending on the articulation of the sign. For instance, a sign performed with a combination of right and left hands may not be detected by handshape models since the hand features are extracted from only the right hand. Therefore, by combining the predictions of pose and handshape models, one can attain a better performing ensemble. For this reason, a cold fusion approach is adopted [5] as

$$\log p = (1 - \gamma) \cdot \log k + \gamma \cdot \log h. \tag{3.4}$$

Here k denotes the prediction of the pose keypoint-based model and h denotes the prediction of the handshape model, and γ is the mixing coefficient. γ is learned on the development set.

3.3. Pre-Trained Word Embeddings

Query embeddings hold a crucial role in the system as they are used to detect the relevant parts of the feature sequence. Even though learning the query embeddings from scratch works well for the baseline system, incorporating pre-trained word embeddings might improve the representational power of the query embeddings. However, pre-trained word embeddings are learned from written languages, and they might not be representative of the gloss queries since sign languages, and written languages have different grammar and word orderings. Therefore, it is not guaranteed to obtain better representations by fine-tuning pre-trained embeddings. Nevertheless, we experiment with two types of word embeddings, word2vec, and fastText.

3.3.1. Word2Vec Embeddings

Word2vec model was proposed to learn distributed representations of the words [71]. It is trained in a self-supervised manner, meaning that the model tries to predict the words in the vicinity of the input word token. By doing so, the network discovers

positive and negative correlations between words. They employ hierarchical softmax, which reduces the computation for large vocabulary systems [71]. Additionally, the frequent words are intentionally subsampled since they are not as informative as the rare words. This model has been shown to improve the word representations at the time drastically.

A word2vec model trained on German Wikipedia was made available by deepset [72]. We use this model to initialize queries in the vocabulary. However, the vocabulary of this word2vec model does not include all the words in our vocabulary. Since there is no way to produce embeddings for out-of-vocabulary (OOV) queries with the word2vec model, those queries are initialized with random values sampled from a Gaussian distribution.

3.3.2. FastText Embeddings

Inspired by the skip-gram model introduced in [73], Bojanowski et al. proposed a method to learn word embeddings by incorporating subword information [73]. Instead of learning representations at the word level, they split each word into n-grams and learn their representations. Afterward, word representations are produced by summing its sub-units [73]. One significant advantage of this method is that it can generate word embeddings even for OOV words.

A fastText model trained on German Wikipedia and Common Crawl is used to initialize query embeddings [74]. Since the fastText method can generate representations for OOV queries, all the entries are initialized by the model.

3.4. Improving Graph Encoder with Graph Attention Networks

The baseline model employs an encoder with 12 ST-GCN layers. However, ST-GCN architecture is very memory exhaustive for a large number of channels. Therefore, Graph Attention Networks (GAT) are studied as a replacement for ST-GCN layers.

GAT was initially proposed for node classification of graph-structured data [75]. Afterward, Huang et al. proposed Spatial-Temporal Graph Attention Networks (ST-GAT) for skeleton-based action recognition that can work on spatial-temporal graphs [76]. They report superior performance compared to ST-GCN [76]. It is also shown that the inference speed of ST-GAT architecture is higher [76]. Even though ST-GAT demonstrates superior results in action recognition, no study within our knowledge has investigated the effectiveness of ST-GAT for SL tasks. In this section, we first introduce the mathematical background of GAT layers, then investigate ST-GAT networks and their implementation.

3.4.1. Graph Attentional Layer

The layer takes a set of nodes as the input $\mathbf{H} = \{\vec{h_1}, \vec{h_2}, .., \vec{h_N}\}, \vec{h_i} \in \mathbb{R}^{\mathrm{D}}$ with feature dimension D and outputs a set of nodes $\mathbf{G} = \{\vec{g_1}, \vec{g_2}, .., \vec{g_N}\}, \vec{g_i} \in \mathbb{R}^{\mathrm{D}'}$ with feature dimension D'. A linear transformation, $\mathbf{W} \in \mathbb{R}^{\mathrm{D}\times\mathrm{D}'}$, is applied to input features. Then, "excitation" between every node is computed by using a shared attention mechanism

$$e_{ij} = \vec{a}^T(\mathbf{W}\vec{h_i} \,||\, \mathbf{W}\vec{h_j}), \tag{3.5}$$

where $\vec{a} \in \mathbb{R}^{2\mathrm{D}'}$ and $||$ is the concatenation operation. Then, attention coefficients are computed as

$$\alpha_{ij} = \frac{\exp(\sigma(e_{ij}))}{\sum_{k\in\mathcal{N}_i}\exp(\sigma(e_{ik}))}, \tag{3.6}$$

where $\mathcal{N}_i$ is first degree neighborhood matrix of the ith node and σ is *LeakyReLU* activation function. Non-neighbor nodes are masked before applying softmax to only include first-degree neighbors in attention calculation. Hence, each output node is the weighted average of linearly transformed neighbor nodes for a single attention head. Optionally, a non-linearity σ can be applied as

$$\vec{g_i} = \sigma\left(\sum_{j\in\mathcal{N}_i}\alpha_{ij}\mathbf{W}\vec{h_j}\right). \tag{3.7}$$

In the case of multi-head attention, output features can be concatenated or averaged across heads. In our implementation, we average across multiple heads to reduce the memory requirements of the layer. Unlike the original implementation, we employ a

learnable adjacency matrix initialized with 1's for first-degree neighbors and a zero-mean small variance Gaussian noise for the other nodes. This adjacency is updated during training, and adjacency values are used as a scaling factor for excitations before computing attention coefficients with softmax. Thus, the effect of irrelevant nodes is reduced. The purpose of learnable adjacency instead of hard selection is for the network to discover relationships between joints that are not directly connected. For instance, the wrist joint is not connected to keypoints in the head. However, there might be a correlation between them in terms of sign articulation that is not obvious from human anatomy. With this modification, attention coefficients are computed as

$$\alpha_{ij} = \frac{\exp(\sigma(e_{ij}) \cdot m_{ij})}{\sum_{k=1}^{N} \exp(\sigma(e_{ik}) \cdot m_{ik})}, \tag{3.8}$$

where m_{ij} is the adjacency factor between node i and j. By introducing a learnable mask, we alter the graph structure from undirected to directed since it is not guaranteed to have the same weights in both directions (i.e., $m_{ij} \neq m_{ji}$).

3.4.2. Spatial-Temporal Graph Attention Networks

After establishing the mathematical background for graph attention, we can expand our discussion to applying GAT to spatial-temporal graphs. ST-GAT model proposed by [76] extends the GAT network by simply re-defining neighborhood function such that temporal and spatial connections are included as the first order neighbors. They employ hard selection for adjacency and increase the temporal ability of the model by incorporating a temporal convolutional network (TCN) and residual connections between each ST-GAT layer [76].

We have experimented with three versions of ST-GAT:

(i) an ST-GAT architecture with only residual connections
(ii) an ST-GAT architecture with TCN and residual connections
(iii) an ST-GAT architecture that disentangles spatial and temporal attention mechanisms

Since the first two settings are trivial, we will discuss the third setting. Let G be a spatial-temporal skeleton graph $G = (V, E)$. The nodes of the graph $V = \{v_{ij} : i = 1, .., N, j = 1, .., T\}$ where N is the number of spatial nodes in each timestep and T is the number of timesteps. Similarly, the edges comprised of two disjoint sets $E = E_s \cup E_t$, $E_s \cap E_t = \varnothing$. $E_s = \{v_{it}v_{jt} : (i, j) \in A\}$ where A is the set of bone connections in accordance with the human anatomy and $E_t = \{v_{it}v_{i(t+1)} : t = 1, .., T\}$. The information encoded in these two types of edges is not from the same modality. Therefore, computing temporal and spatial attention separately and then combining them might increase the expressive power of the network compared to aggregating spatial and temporal relationships into a single attention mechanism. To this end, the adjacency matrix is divided into two parts corresponding to spatial and temporal connections. Temporal adjacency is applied as hard selection as opposed to learned neighborhood coefficients for spatial connections. The modified output of a node representing ith joint in kth timestep is

$$\vec{g_{ik}} = \sigma\left(\sum_{j=1}^{N} \alpha_{ij}\mathbf{W_1}\vec{h}_{jk} + \sum_{t\in\mathcal{N}_k} \beta_{kt}\mathbf{W_2}\vec{h}_{it}\right), \tag{3.9}$$

where α_{ij} is spatial attention coefficients, β_{kt} is temporal attention coefficients and $\mathcal{N}_k = \{k, k-1, .., k-\Delta\}$ where Δ is the temporal kernel size. Attention coefficients are computed using Equation (3.8).

Separating temporal and spatial attention mechanisms increases the network's computational cost significantly due to the number of parameters getting doubled. Due to memory limitations of the GPU, this setting could not be exhaustively tested. However, the effect of each modification to GAT architecture and the results obtained from each listed setting are comparatively discussed in Section 4.2.

Instead of employing only ST-GCN or only ST-GAT in the graph encoder, they can be joined together to leverage the advantages of both architectures. In fact, the experiments showed that optimal performance is achieved when these two architectures are combined. In the final graph encoder, ST-GAT is used together with TCN and residual connections. This graph encoder is illustrated in Figure 3.4.

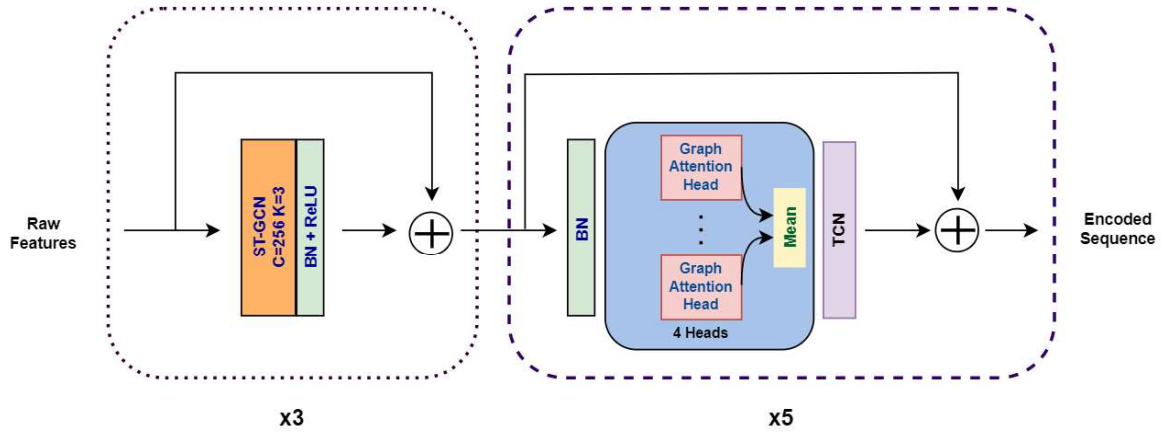


Figure 3.4. Graph encoder comprised of ST-GCN and ST-GAT layers.

3.5. Pseudo-Relevance Feedback

Relevance feedback is a known beneficial strategy that has been applied for text information retrieval [77]. By receiving feedback from the user, the search system improves itself. It has also been applied in spoken content retrieval [78]. In pseudo-relevance feedback (PRF), the system generates feedback for itself without any user interaction. Since our system has not been tested with users, we chose to adapt PRF to SL-KWS. Let us review the mathematical background of example-based PRF [79] that is used in speech retrieval.

3.5.1. Mathematical Background

Let P_Q be the pseudo-positive set of examples for query Q and $X_Q \in P_Q$. The distance between an example X and a positive example set is defined as

$$D(P_Q, X) = \sum_{X_Q \in P_Q} d(X_Q, X), \tag{3.10}$$

where $d(.,.)$ is a distance metric used for calculating the distance between positive example X_Q and X feature vectors. DTW is a common choice for speech applications. After computing the total distance of a query to the positive set, we calculate the similarity score

$$SIM(P_Q, X) = 1 - \frac{D(P_Q, X)}{M_Q}, \tag{3.11}$$

where M_Q is the maximum possible value of $D(P_Q, X)$. Now the original prediction scores $S(Q, X)$ are updated to obtain new predictions $S_1(Q, X)$,

$$S_1(Q, X) = S(Q, X)(SIM(P_Q, X))^a, \tag{3.12}$$

where a is an exponential weight parameter. The pseudo-positive set is obtained by selecting K examples with the highest scores for the given query. This selection entails a problem of confidence in the system's initial predictions. If the system's retrieval performance is low, the application of PRF might even worsen the results as the top K examples might belong to the negative class. Therefore, fine-tuning K is crucial. Additionally, we do not apply PRF to low-confidence queries.

3.5.2. Pseudo-Relevance Feedback for SL-KWS

Upon establishing the mathematical background for example-based PRF, its application to our problem can be discussed. Instead of sequential features, we use context vectors that are produced by the selection mechanism. We introduce three modifications to the original algorithm as presented in [80]:

(i) We experiment with three distance metrics:
- Cosine distance.
- l_2 distance.
- Query-specific Mahalanobis distance.

(ii) Score normalization for distribution matching between the initial predictions and PRF scores.

(iii) Learning a mixing coefficient for combining the initial and PRF predictions.

3.5.2.1. Query-Specific Distance Metric Learning. Even though cosine and l_2 distance metrics work well for most scenarios, the performance can be improved by learning a new distance metric that maximizes the distance between positive and negative examples. To this end, we learn a distance metric for each query in the training set vocabulary. All the positive examples of the query and the same number of nega-

tive examples are combined to form the input and the labels from the training set provided to the metric learning algorithms. We use Neighborhood Component Analysis (NCA) [81] method to learn Mahalanobis distance. NCA tries to maximize the performance of the leave-one-out k-Nearest Neighbors (kNN) algorithm by learning a transformation matrix that separates positive and negative examples in a new space.

After learning Mahalanobis matrix M, the distance between two points is computed as

$$d_M(x_1, x_2) = \sqrt{(x_1 - x_2)^T M (x_1 - x_2)}\,. \tag{3.13}$$

This is equivalent to the Euclidian distance between two points in a new space.

3.5.2.2. Score Normalization and Learning Mixing Coefficient. The initial prediction scores lie between 0 and 1, representing the probability of the query being present in a specific video. However, Mahalanobis distance is unbounded like Euclidian distance. Hence, the resulting distances lie in a much wider range than $(0, 1)$. Even though the distance values can be mapped into the $(0, 1]$ range by dividing its maximum value, the distribution of the scores still needs to be normalized before mixing them so that the relative meaning of the scores is comparable. Thus, we applied two score normalization techniques adopted from [82], namely, Gaussian normalization and Median normalization defined as

$$\mathbf{s_z} = \frac{\mathbf{s} - \mu(\mathbf{s})}{\sigma(\mathbf{s})} \tag{3.14}$$

$$\mathbf{s_b} = \frac{\mathbf{s} - median(\mathbf{s})}{\sigma(\mathbf{s} > median(s))}. \tag{3.15}$$

Then, the normalized prediction scores are than mixed together as

$$p^Q = \gamma \cdot p_i^Q + (1 - \gamma) \cdot p_{prf}^Q, \tag{3.16}$$

where p_i^Q and p_{prf}^Q denote original and PRF prediction scores for query Q, respectively. γ is learned with a linear search on the training set by choosing the value that maximizes mean average precision (mAP). The overall PRF algorithm is described in Figure 3.5.

INPUT:

C : Context vectors of the target corpus

P : Initial prediction for the target corpus

V : Selected subset of the vocabulary of the target corpus

N : The number of utterances in the target corpus

$d(x_1, x_2) \Leftarrow$ Choose cosine, l_2 or Mahalanobis distance as the distance metric

K : The number of queries with the highest scores that are assumed to be correct

γ : Mixing coefficient learned from tranining set

OUTPUT:

S : PRF scores.

F : Final predictions after combined with PRF.

FUNCTION PRF(**C**, **P**, **V**, N, d, K, γ):

for q in V **do**

  if $\max(\mathbf{P}^q) < 0.9$ **then**

    continue {skip the low confidence queries}

  end if

  p $\Leftarrow$ argsort($\mathbf{P}^q$)

  for $i = 1$ to N **do**

    for $k = 0$ to $K - 1$ **do**

      $j \Leftarrow p[k]$

      $\mathbf{S}_i^q \Leftarrow \mathbf{S}_i^q + d(\mathbf{C}_j^q, \mathbf{C}_i^q)$

    end for

    $\mathbf{S}_i^q \Leftarrow \mathbf{S}_i^q / N$

  end for

  $\mathbf{F}^q = \gamma \cdot \mathbf{P}^q + (1 - \gamma) \cdot \mathbf{S}^q$

end for

return F, S

end FUNCTION

Figure 3.5. Pseudo-Relevance Feedback Algorithm.

3.6. Similarity-Based SL-KWS

The system explained in the baseline section is trained with binary cross-entropy loss by classifying context vectors via perceptron. However, it is possible to train a model with similarity loss functions by establishing a relationship between context vectors and query embeddings and removing perceptron. Even though such models are not expected to perform as well as the models trained with classification loss, they can still explore different relationships between queries and context vectors. Hence, they might perform better for some queries than classification models. The fusion strategy can improve retrieval accuracy by combining similarity-based and classification-based models. After the fusion of those two types of models, the fusion strategy can still be applied to join handshape and pose models. Therefore, introducing an intermediate fusion step where each type of model is fused with its similarity counterpart before combining cross-feature models can be beneficial. To this end, we develop two training techniques based on cosine similarity loss and triplet loss.

3.6.1. Cosine Similarity Loss

Cosine loss is frequently used for learning high dimensional non-linear representations. It is defined as

$$L_{\cos}(x_1, x_2, y) = \begin{cases} 1 - \cos(x_1, x_2), & y = 1 \\ \max(0, \cos(x_1, x_2) - m), & y = -1, \end{cases} \tag{3.17}$$

where x_1 and x_2 are input vectors, y is the similarity label and m denotes tolerance margin. This method is illustrated in Figure 3.6.

Using cosine loss, the context vectors of positive queries are brought closer to corresponding query embeddings, while the context vectors of negative queries are separated from corresponding query embeddings. Thus, the similarity of the positive queries increases, whereas the negative similarity scores are suppressed. The weighting coefficients of the two terms are equal.

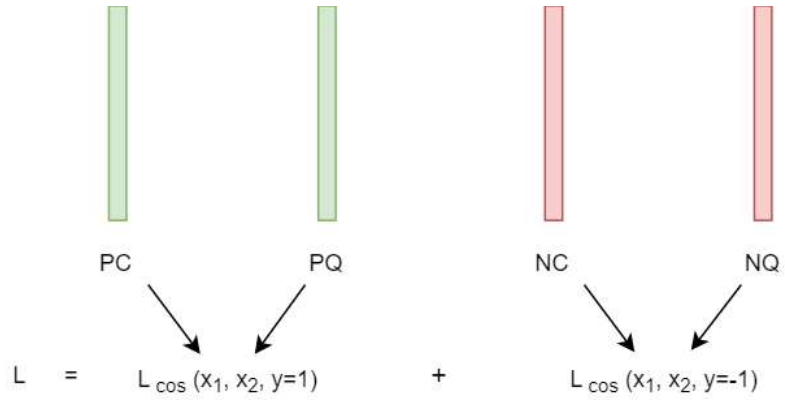


Figure 3.6. Cosine loss configuration. Green represents positive and red represents negative examples. PC: positive context. PQ: positive query. NC: negative context. NQ: negative query.

3.6.2. Triplet Loss

Triplet loss is frequently used for few-shot learning problems in computer vision [83]. It aims to make the distance between positive and anchor examples smaller than the distance between negative and anchor examples by a margin. It is defined as

$$L_{\text{trip}}(p, a, n) = \max\{d(a, p) - d(a, n) + m, 0\}, \tag{3.18}$$

where p, a, and n are positive, anchor, and negative examples, respectively, and m is the minimum margin between the difference of distances. $d(.,.)$ is the distance metric chosen as cosine distance. This method is illustrated in Figure 3.7.

Triplet loss is applied to two sets of inputs. In the first setting, positive context vectors as anchors, positive query embeddings as positive examples, and negative query embeddings are selected as negative examples. In the second setting, negative context vectors as anchors, positive context vectors as positive examples, and negative query embeddings are given as negative examples. The first setting increases the similarity between positive context vectors and positive query embeddings. Thus, leading to the detection of positive queries. The second setting broadens the gap between negative query embeddings and negative context vectors, helping to prevent false alarms.

During inference, both methods' predictions are generated by computing cosine similarity between the embedding of given query and context vectors corresponding to the same query in all utterances of the corpus. The similarity scores are then ranked and presented to the user as the retrieval result for the given query.

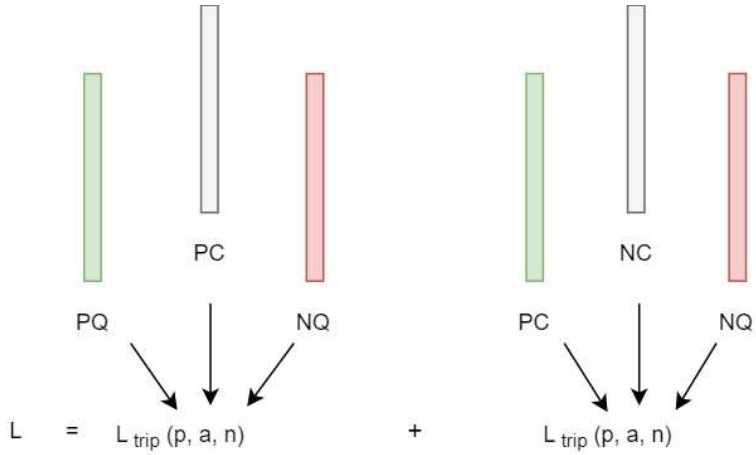


Figure 3.7. Triplet loss configuration. Green represents positive, red represents negative and gray represents anchor examples. PC: positive context. PQ: positive query. NC: negative context. NQ: negative query.

3.7. Iterative SL-KWS

In the baseline model, each query is assumed to be independent, and the relationships between queries are not exploited. Including a query encoder that captures queries' interactions may yield better retrieval results. Therefore, we have implemented a Transformer-based [84] query encoder. Transformer architecture is shown to be superior to RNN-based encoder-decoder models in Neural Machine Translation (NMT) [84]. After obtaining encoded query representations, they must be combined with the encoded video sequence. We employed a cross-modal attention mechanism that is used for temporal sentence localization in videos [85]. This module utilizes a two-way attention technique whose details are explained in Section 3.7.2. The remaining components of the system are kept the same. The training and the inference strategies of this model are explained in Sections 3.7.3 and 3.7.4. The overall flowchart of the system is given in Figure 3.8.

3.7.1. Query Encoder

Transformer architecture takes a sequence of inputs. The input length is decided by the longest sequence in the dataset. After determining the longest sequence, other sequences are padded to match this length. Later on, they are masked according to their length to ensure proper computation of the backpropagation. Positional encoding is usually used to preserve the sequential nature of the input in Transformers. However, we do not apply any positional encoding as our model works in a bag-of-words manner. The selection of the bag-of-words method will be explained in Section 3.7.3 where we discuss the training strategy of the model.

Transformer architecture employs self-attention on the inputs, normalizes them, and feeds the result into a dense network. Residual connections between layers are also used to mitigate vanishing gradient problems. The resulting query encoder architecture consists of 3 Transformer encoder layers.

3.7.2. Cross-Modal Attention Layer

Cross-modal attention is an effective method for discovering interactions between two sequences from different modalities. It is often used in video-text applications. It establishes two-way attention between queries and the video sequence.

The encoded query $\mathbf{Q} \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ and video $\mathbf{V} \in \mathbb{R}^{\mathrm{T}\times\mathrm{D}}$ sequences are first carried into a shared space via $\mathbf{W_S} \in \mathbb{R}^{\mathrm{D}\times\mathrm{D}}$. Then, the similarity matrix $\mathbf{S}$ is computed as

$$\mathbf{S} = \mathbf{V}(\mathbf{Q}\mathbf{W_S})^T \in \mathbb{R}^{\mathrm{T}\times\mathrm{N}}. \tag{3.19}$$

In this matrix, each column holds a similarity score between query $\mathbf{q}_i$ and all the frames in the video sequence. Similarly, the rows store the similarity score between video frame $\mathbf{v}_f$ and all the queries in input. Similarity scores are masked according to the video and query lengths.

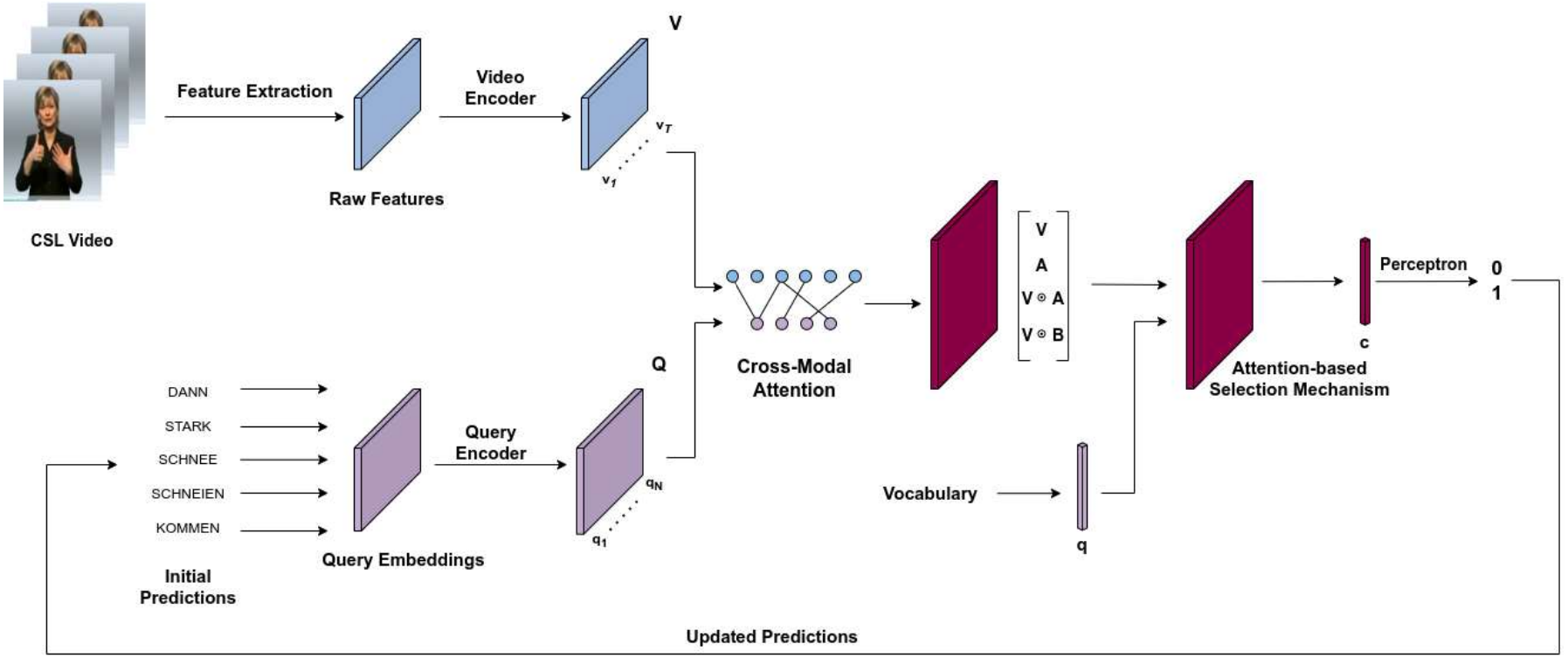


Figure 3.8. Iterative SL-KWS pipeline.

Then, two attention weights are computed as

$$\mathbf{A} = \mathbf{S_r}(\mathbf{QW_S}) \in \mathbb{R}^{\mathrm{T}\times\mathrm{D}} \tag{3.20}$$

$$\mathbf{B} = \mathbf{S_r}\mathbf{S_c}^{T}\mathbf{V} \in \mathbb{R}^{\mathrm{T}\times\mathrm{D}}, \tag{3.21}$$

where $\mathbf{S_r}$ and $\mathbf{S_c}$ are obtained by applying the softmax function on rows and columns of $\mathbf{S}$, respectively. Here, $\mathbf{A}$ stores video-to-query attention whereas $\mathbf{B}$ holds query-to-video attention.

The final output of this layer is obtained as

$$\mathbf{H} = \mathbf{W}_H\left[\mathbf{V}; \mathbf{A}; \mathbf{V}\odot\mathbf{A}; \mathbf{V}\odot\mathbf{B}\right] \in \mathbb{R}^{\mathrm{T}\times\mathrm{D}}, \tag{3.22}$$

where $\mathbf{W}_H$ is used to map the concatenated output of the layer back into the original dimensions $\mathbb{R}^{\mathrm{T}\times\mathrm{D}}$, and $\odot$ stands for the element-wise multiplication.

3.7.3. Training Strategy

Unlike the original model, this system configuration admits a set of queries as input. The selection of those queries is crucial to the model's training. Initially, query inputs were directly taken from the labels of the input videos. However, this approach trivializes the model's training since it simply learns to repeat query inputs at the output. Upon this observation, we have decided to move forward with two other approaches. The first approach is to input the top predictions of a pre-trained model for a given video. The other approach is to train the model similar to the Expectation-Maximization (EM) method. In the latter technique, the model first generates its predictions without updating model parameters and then uses those prediction results as input to the query encoder. This iterative approach is the primary reason behind the selection of the bag-of-words technique since the system does not output any sequential information about the queries. Therefore, it is not advantageous to employ positional embeddings. In the first epoch, queries are randomly sampled from the vocabulary.

The queries to be inputted are selected by ranking scores of all queries for the given video. Top K query is selected as the input. K is decided by dividing the

number of frames in the video by a number of frames per query constant l. l is decided by sweeping integer values between 5 and 30. The number that minimizes the mean squared error is selected as l. Additionally, the ranked query inputs are shuffled to ensure the stochasticity further. The experiments prove the superiority of the latter training approach.

3.7.4. Inference

Another advantage of iterative training is to have a natural way of combining different models' predictions by simply feeding one with the other's predictions. This attribute can be exploited to refine the predictions of the system further. Thus, we store two models during training, one that leads to a maximum mean AP (mAP) score and the one that yields the best loss in the development set. During inference, we randomly initialize the best loss model's inputs and then feed its outputs to itself for N times. Then, the resulting predictions are fed into the best mAP model as the initial predictions and are refined for N' iterations. This way, we can transfer information from the best loss model to the best mAP model, leveraging multiple checkpoints.

4. EXPERIMENTS AND RESULTS

In this chapter, conducted experiments and their results are explained and discussed. We first introduce an experimental setup that includes the dataset, evaluation metrics, and the implementation details in Section 4.1. In Section 4.2, we present the results for improved graph encoder. In Section 4.3, PRF results are reported and discussed. Results of the similarity-based models are given in Section 4.4. We report and discuss the results of the new iterative training method in Section 4.5. In Section 4.6, we present the results for cross-lingual search. Lastly, we report the best performing models obtained by the fusion of iterative pose and handshape models in Section 4.7.

4.1. Experimental Setup

In this section, we introduce the dataset in which the experiments are carried out, and we define the evaluation metrics used to measure the success of the experiments. Lastly, the details regarding the implementation are explained.

4.1.1. Dataset

All the experiments are carried out on RWTH-PHOENIX-Weather 2014T dataset [86]. The dataset contains 9.2 hours of training, 37 minutes of development, and 43 minutes of test footage of weather forecast in German sign language, signed by nine different signers, in 25 fps videos. The dataset also includes sentence-level gloss transcriptions without temporal annotations and cross-lingual (German) translations of the sentence, which are utilized to form vocabularies for gloss and cross-lingual search, respectively.

Gloss vocabulary contains 1085 queries in the training set, and 398 of those are also present in the test set. Similarly, German keyword vocabulary contains 2887 queries in the training data, and 942 of those are also encountered in the test data.

Since there is no correlation between the written form of a query and its sign, the search can only be done on in-vocabulary queries. Consequently, results are reported on queries present in both training and test sets, which share 398 queries for gloss search and 942 queries for cross-lingual search.

4.1.2. Evaluation Metrics

4.1.2.1. Mean Average Precision (mAP). Average precision (AP) for a query q can be written as

$$\text{AP} = \frac{1}{N} \sum_{n=1}^{N} \text{Precision@}n(q), \tag{4.1}$$

where N is the number of occurrences of query q in the dataset. Upon computing AP scores for all the queries in the vocabulary, mean AP (mAP) score is calculated by averaging all the queries.

4.1.2.2. Precision at N (p@N). p@N is calculated by computing the precision for the first N retrieved items, where N is the number of occurrences of a query in the target set.

4.1.3. Implementation Details

The code is written in Python 3.8. The neural networks are implemented and trained using PyTorch 1.8. Nvidia GeForce GTX 1080Ti and RTX 2080Ti are used for training. The models are trained using AdamW optimizer with a 0.003 learning rate and weight decay. Learning rate scheduling is also applied. If the development loss does not decrease for three consecutive epochs, then the learning rate is halved. After every halving, three epoch cooldown period is introduced. The halving process continues until the lower bound for the learning rate is reached, which is determined as 10^{-5}. Training stops when no improvement is observed in the mAP score of the development set for six consecutive epochs. The model with the best mAP score on the development set is saved and used for testing. Both models with the best

development loss and the best mAP score are saved for the iterative training method. The videos over 225 frames (9 seconds) are removed from the dataset to reduce memory requirements. They approximately correspond to 1.5% of the videos in the dataset.

4.2. Improved Graph Encoder Results

The modifications to the graph encoder were introduced in Section 3.4.2. In this section, we provide an ablation study for the graph encoder that emphasizes the individual contribution of each proposed modification. We also analyze the learned mask to deduce underlying relationships within pose keypoints.

4.2.1. Comparison of Different Graph Encoder Architectures

In this section, we investigate the effect of each component introduced to the graph encoder. The resulting mAP score for the different configurations are given in Table 4.1.

If we compare Setting 1 and 2, it is seen that the contribution of TCN is quite significant. The temporal modeling power of the encoder severely degrades when TCN is excluded. Similarly, it is observed that the contribution of TCN is more significant than the introduced temporal attention mechanism when Settings 2 and 4 are compared. However, the temporal attention mechanism still improves temporal modeling significantly as the performance of Setting 4 is far greater than that of Setting 2. We may also inspect the effect of employing a learnable mask instead of hard selection. It is seen that the learnable mask introduces approximately 6% improvement to mAP score when Setting 2 and 3 are compared. Lastly, when Setting 5 and 6 are compared, it is seen that the joining ST-GCN and ST-GAT architectures yield the best results. We also expect improvement in the performance if the temporal attention mechanism were incorporated into the encoder. However, due to memory limitations of the GPU, it is left out of the final architecture.

Table 4.1. The effect of each modification. All the models are trained with OpenPose features. Temporal Att. stands for temporal attention. '+' means included and '-' means excluded from the encoder.

	TCN	**Temporal Att.**	**Learnable Mask**	**ST-GCN**	**mAP (%)**
1	-	-	+	-	12.89
2	+	-	+	-	28.31
3	+	-	-	-	22.51
4	-	+	+	-	23.21
5	+	-	+	+	**34.91**
6	-	-	-	+	29.61

4.2.2. Analysis of the Learned Mask

In this section, we examine the learnable mask to explore some underlying dynamics between joints. As explained in Section 3.4.2, the learnable mask is used as a scaling factor before the computation of the attention coefficients. If m_{ij} is big compared to other coefficients in the matrix, it can be interpreted as the effect of the node j on the node i is relatively significant. Keeping this in mind, we made the following analyses:

(i) Learn the most significant nodes (i.e. ones that significantly contribute to other nodes) by computing $s_j = \sum_i m_{ij}$ for each node and comparing them. The greater the s_j, the greater the significance on other nodes.

(ii) Learn the least significant nodes (i.e., ones that have a relatively low impact on the computation of the others) by computing the s_j as in (i). This analysis can be used for feature selection.

4.2.2.1. Investigating the Most Significant Nodes. First we compute $\vec{s}$ where $s_j = \sum_{i \notin A} m_{ij}$ and A is the set of natural connections. In other words, we exclude the direct connections dictated by human physiology since those coefficients are inherently

large, and the nodes with many connections to others can skew the results. The nodes with the highest contribution score are given in Table 4.2. The contribution scores are normalized by dividing 52, which is the maximum possible value when we exclude coefficients of the anatomical connections.

Table 4.2. The nodes with the most contribution and their corresponding scores. RH stands for right hand. Numbers within parantheses denote the index of the joint per Figure 4.1(c).

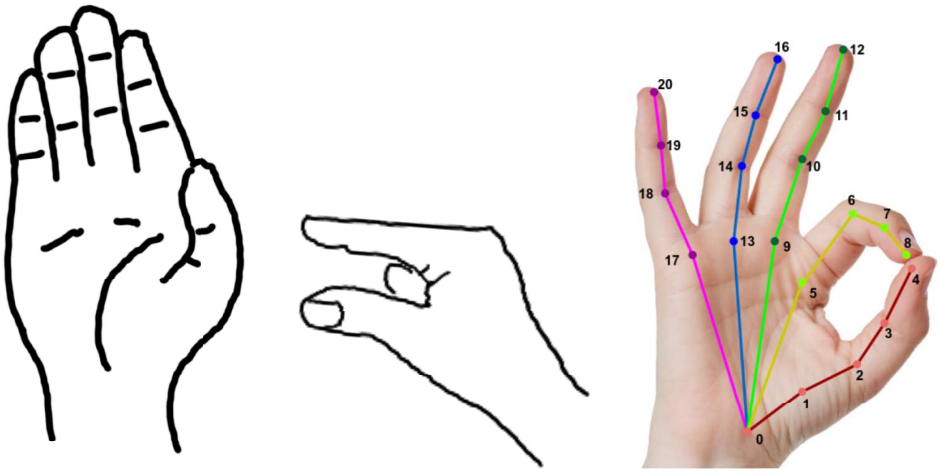
	Joint Name (Joint Index)	**Contribution Score**
1	RH - Thumb Middle (3)	0.193
2	RH- Thumb End (4)	0.168
3	RH- Ring Finger Start (14)	0.163
4	RH - Ring Finger End (16)	0.153
5	RH - Thumb Start (2)	0.150
6	RH - Index Finger End (8)	0.137
7	RH - Middle Finger Start (10)	0.136
8	RH- Middle Finger End (12)	0.133
9	RH- Ring Finger Middle (15)	0.128
10	RH - Middle Finger Middle (11)	0.112

It can be seen from Table 4.2 that the most significant nodes are the first four fingers of the right hand. It is not surprising since it is known that the handshape is the most informative SL channel and the positions of the first four fingers determine the handshape to a great extent. However, a high contribution score can be obtained if a joint is significant for a few nodes with a large coefficient. To answer this question, we can also investigate how often a node is the most significant node for another node. In other words, we ask that the node in question is the primary contributor for how many nodes, excluding natural connections. The results are reported in Table 4.3 and the results are consistent with Table 4.2. This way, we showed that these keypoints are relevant for many others instead of having large coefficients for a few nodes.

Table 4.3. Number of times a node is the most significant node for another. RH stands for right hand.

	Joint Name (Joint Index)	**#Nodes**
1	RH - Thumb Middle (3)	11
2	RH - Thumb Start (2)	8
3	RH- Thumb End (4)	7
4	RH- Ring Finger Start (14)	6
5	RH - Middle Finger Start (10)	4

4.2.2.2. Investigating the Least Significant Nodes. An analysis similar to Section 4.2.2.1 might be carried out for the least significant nodes. These nodes contribute very little and might be removed from the feature set for future studies to reduce complexity. In Table 4.4, s_j scores of the least significant nodes are given. Contribution scores are not normalized.



(a) (b) (c)

Figure 4.1. Handshape A, Handshape B and OpenPose hand keypoints layout.

These joints can be eliminated from the feature set as they contribute very little to overall attention computation. The results are not surprising because the movement of these parts does not convey much information about the sign articulation. For example, rather than the position of the eyes, the information about blinking speed,

eye openness, or the blinking type is more informative for SL. Similarly, the position of the nose is not very relevant. The most informative features of the face are mouthings, facial expressions, and eye and eyebrow movements, none of which are included in this study.

Table 4.4. The nodes with the least contribution and corresponding scores. Numbers within parantheses denote the index of the joint per Figure 4.1(c).

	Joint Name (Joint Index)	**Contribution Score**
1	Left Ear (11)	-1.834
2	Left Eye (12)	-1.620
3	Right Ear (9)	-1.467
4	Right Eye (10)	-1.328
5	Nose (0)	-0.390

4.3. Pseudo-Relevance Feedback Results

In order to evaluate the effectiveness of PRF, we take predictions from a DeepHand model. We first select a subset of queries from the test that occurs more than 60 times in the training set to select better-performing queries. Afterward, PRF scores for those queries are calculated in the training set and are used to learn an optimal mixing coefficient. Using this mixing coefficient, the initial test predictions and the test PRF scores are combined to obtain new predictions. Optionally, score normalization is applied. The top three entries are assumed correct. We also experimented with query-specific mixing coefficients learned on the training set. However, they worsen the results due to overfitting in the mixing coefficient. The results are reported in Table 4.5.

The results show that PRF introduces negligible improvements. Now we will hypothesize why this is the case. The main difference between our approach and the conventional example-based PRF used in speech retrieval is that our features are not

sequential since our system does not produce temporal hypotheses. Therefore, we use context vectors. However, it seems that we cannot infer any additional information from the context vectors other than what was already inferred by the perceptron. This might be due to the fact that the score distributions from our system are quite different from the speech KWS systems. Therefore, the application of score normalization techniques is not helpful as they are for speech retrieval.

Table 4.5. PRF results. Scores are given in percentages.

Distance Metric	**Score Normalization**	**mAP Gain**	**p@N Gain**
cosine	-	0.08	0.02
l_2	-	0.09	0.02
Mahalanobis	-	-0.01	-0.05
cosine	Median	0.05	0.03
l_2	Median	0.05	0.03
Mahalanobis	Median	0.02	0.03
cosine	Gaussian	0.05	0.03
l_2	Gaussian	0.05	0.03
Mahalanobis	Gaussian	0.02	0.03

4.4. Similarity-based Models

The training of similarity-based models are explained in Section 3.6. We report their performance and the fusion results in Table 4.6 and Table 4.7. It is seen in Table 4.6 that the models trained with DeepHand features outperform the models trained with MultiTask features, as the DeepHand model was also trained on the RWTH-Phoenix dataset which we use in our experiments. Fusion of Multitask classification and Deephand similarity models achieve the highest gain and relative improvement. Similarity models are expected to perform worse than classification models since the prediction scores are computed by cosine similarity between query embeddings and context vectors. Every query is represented with a single embedding that may not

be brought closer to every positive context vector in the corpus. It may be suggested that learning a query representation similar to all positive context vectors limits the expressive power of the model.

Table 4.6. Results for cosine similarity-based models and their fusion with classification models. Rel. Impr. stands for relative improvement.

Classification		**Cosine**		**Fusion**		
Feature	**mAP**	**Feature**	**mAP**	**mAP**	**Gain**	**Rel. Impr.(%)**
DeepHand	28.08	DeepHand	21.54	**30.42**	2.34	8.33
MultiTask	25.60	DeepHand	21.54	28.7	**3.10**	**12.11**
DeepHand	28.08	MultiTask	17.68	29.2	1.12	3.99
MultiTask	25.60	MultiTask	17.68	26.92	1.32	5.16

Similar observations can be made for triplet similarity results reported in Table 4.7. If we compare triplet and cosine loss, we see that they accomplish similar results for both features. For DeepHand features, cosine similarity works better. In order to analyze the performance of similarity-based models, we further investigate the query-specific performances of similarity and classification models. We inspect the queries that obtained higher retrieval scores when trained with cosine similarity loss. Some of the queries with the most performance improvement under similarity training are listed in Table 4.8. We subtract the baseline DeepHand model's AP scores from the AP scores of the cosine model for every query.

In order to discover why those queries listed in Table 4.8 performed better in similarity training, their sign articulations are viewed from a sign dictionary [87]. The primary handshape for the first four queries is determined as "flat closed fingers", which we will refer to as handshape A and is illustrated in Figure 4.1(a). For the fifth query "WENIG", the primary handshape is given in Figure 4.1(b), and we refer to it as handshape B for the rest of this section. When we check the classification performance of the DeepHand model for these handshapes, it is observed that handshape B was identified

with 3.9% precision, and handshape A obtained a 41.7% precision score [18]. They are the two handshapes with the lowest precision scores among the handshapes illustrated in the given confusion matrix [18]. Hence, one may suggest that the similarity-based models might be more robust to the noise in the feature embeddings assuming that the features obtained for those handshapes were not very representative. In order to verify this hypothesis, more analyses must be conducted, which is outside the scope of this thesis.

Table 4.7. Results for triplet similarity-based models and their fusion with classification models. Rel. Impr. stands for relative improvement.

Classification		**Triplet**		**Fusion**		
Feature	**mAP**	**Feature**	**mAP**	**mAP**	**Gain**	**Rel. Impr.(%)**
DeepHand	28.08	DeepHand	20.39	**29.61**	1.53	5.45
MultiTask	25.60	DeepHand	20.39	28.45	**2.85**	**11.13**
DeepHand	28.08	MultiTask	17.76	29.14	1.06	3.77
MultiTask	25.60	MultiTask	17.76	26.66	1.06	4.14

Table 4.8. AP difference between similarity and classification models for some queries.

	Query	**AP Difference (%)**
1	DESHALB	56.4
2	OFT	50.0
3	WEITER	37.5
4	ANGENEHM	29.5
5	WENIG	28.7

4.5. Iterative SL-KWS Results

In this section, we report the results for the models trained with default and iterative methods. Additionally, we investigate the improvement introduced to the

results by each iteration and model combination.

4.5.1. Iterative and Default Training Methods

We present the iterative and default training results in Table 4.9. The best results are indicated in boldface. It is seen that the iterative approach has increased the performance of all configurations. The most significant improvement is observed for OpenPose features with approximately a 5% increase in mAP score. Note that the default training results for handshape models are different from those initially reported in [6]. This is due to improvements made in the 1D-CNN encoder and attention-based selection mechanism. 1D-CNN encoder is further regularized with batch normalization (BN). Additionally, layer normalization is applied to context vectors before they are fed into perceptron. Furthermore, we obtained comparable results for OpenPose ST-GCN architecture by reducing the number of layers from 12 to six.

Table 4.9. Results for iterative and default training methods. Results are given as percentages.

		Default		**Iterative**	
Feature	**Encoder**	**mAP**	**p@N**	**mAP**	**p@N**
OpenPose	ST-GCN	29.61	26.43	34.29	30.96
OpenPose	ST-GCN + ST-GAT	34.91	29.95	**39.55**	**35.18**
DeepHand	1D CNN	30.04	27.90	31.76	28.19
MultiTask	1D CNN	27.06	24.52	29.34	25.81

4.5.2. Prediction Refinement

One of the advantages of iterative training is that the model can improve upon its initial predictions. In Table 4.10, we listed how each iteration improves the retrieval performance. The improvement over iterations can be observed in Figure 4.2 as well. In the first iteration, the model is initialized with random queries. The previous iteration's

predictions are used as query input in the following iteration. The first two iterations are carried out by the model with the best loss on the development set, whereas in the last iteration, the predictions of the best development loss model are given to the model with the best mAP score on the development set. This number of iterations is found to be optimal after carrying out experiments in the development set. Increasing the number of iterations sometimes degrades the performance slightly. Additionally, the results at the inference time are not deterministic as the first inputs are given as random. This is why we observe different scores from what was reported in Table 4.9.

Table 4.10. Prediction refinement progress. ST-GCN model is used for OpenPose features.

	OpenPose		**DeepHand**	
Iteration	**mAP (%)**	**p@N (%)**	**mAP (%)**	**p@N (%)**
1	30.28	26.29	28.91	25.51
2	31.07	27.24	29.32	26.20
3	34.60	31.23	32.02	28.51

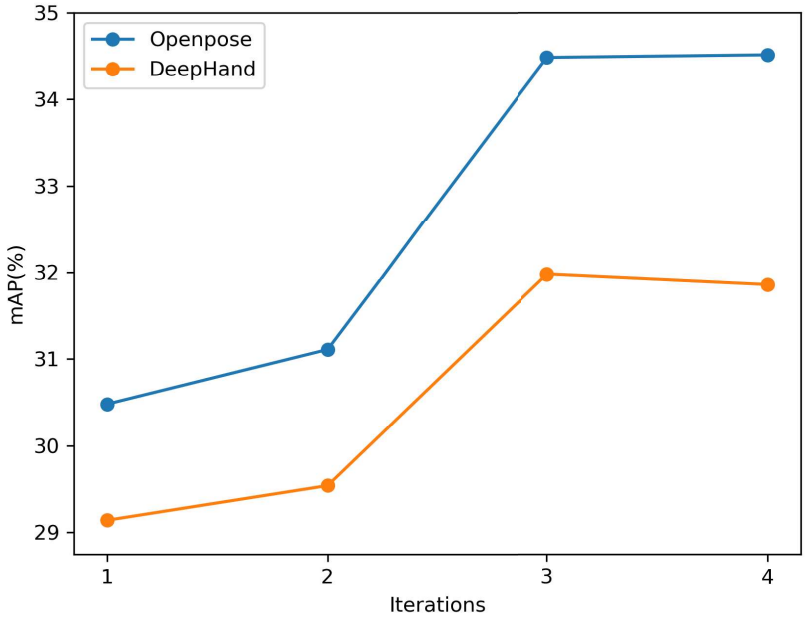


Figure 4.2. Improvement over iterations.

4.6. Cross-Lingual Search

In addition to the gloss search, the system can be used to retrieve cross-lingual queries. Since the cross-lingual labels are very noisy, the model's performance is relatively low compared to the gloss search. We also experiment with initializing query embeddings with pre-trained word embeddings. The models are trained with the default training method. The results, including baseline scores, are reported in Table 4.11. It is seen that combining ST-GAT and ST-GCN encoders is beneficial for the cross-lingual search even if the pre-trained embeddings are not used. Incorporating ST-GAT and pre-trained embeddings boosts the mAP score by 2.9% compared to the best baseline model. The results do not point to a superior pre-trained embeddings method as the performance changes with the feature. However, incorporating the pre-trained embeddings improved the results for all features.

Table 4.11. Cross-lingual search results. Scores are given as percentages. The models marked with '*' are baseline models.

Feature	**Encoder**	**Pre-Trained**	**mAP**	**p@N**
OpenPose*	ST-GCN	-	13.14	10.39
OpenPose	ST-GCN + ST-GAT	-	14.35	11.21
OpenPose	ST-GCN + ST-GAT	word2vec	**16.04**	**13.27**
OpenPose	ST-GCN + ST-GAT	fastText	15.04	12.07
DeepHand*	1D CNN	-	11.11	9.14
DeepHand	1D CNN	word2vec	12.55	9.71
DeepHand	1D CNN	fastText	13.74	10.50
MultiTask*	1D CNN	-	10.44	8.75
MultiTask	1D CNN	word2vec	12.88	10.02
MultiTask	1D CNN	fastText	12.73	9.29

4.7. Fusion Results

Fusion strategy is introduced in [5] and explained in Section 3.2.3. It is possible to combine the predictions of pose and handshape models using the fusion technique. We report the combination of the best OpenPose and DeepHand models in Table 4.12. MultiTask models are left out as DeepHand models outperform them. γ is the mixing coefficient. The values less than 0.5 point out higher reliance on the pose model, and the values greater than 0.5 indicates higher reliance on the handshape model. The best performing combination achieves 45.38% mAP score, which introduces approximately 13% mAP improvement upon what was reported in the baseline study [6]. It is seen in Table 4.12 that DeepHand models significantly benefit from the pre-trained embeddings, yielding to 1.53% for fastText and 2.77% mAP score improvement for word2vec embeddings.

Table 4.12. Fusion of iteratively trained OpenPose and DeepHand models. ST-GCN + ST-GAT architecture is used for OpenPose models.

OpenPose			**DeepHand**			**Fusion**		
Pre-Trained	**mAP (%)**	**p@N (%)**	**Pre-Trained**	**mAP (%)**	**p@N (%)**	**mAP (%)**	**p@N (%)**	γ
-	39.55	35.18	-	31.76	28.19	44.07	38.48	0.4
-	39.55	35.18	word2vec	34.53	30.86	44.40	35.65	0.48
-	39.55	35.18	fastText	33.29	29.23	44.13	38.33	0.56
fastText	39.97	36.75	-	31.76	28.19	44.07	39.63	0.38
fastText	39.97	36.75	word2vec	34.53	30.86	**45.38**	**40.71**	0.4
fastText	39.97	36.75	fastText	33.29	29.23	44.43	38.82	0.56

5. CONCLUSION

The Deaf lack the resources and tools to access information in the internet age easily. This excludes hard-of-hearing people from the many essential aspects of life, including education, getting and holding a job, and social relationships. Therefore, developing information retrieval tools designed explicitly for sign languages is necessary. The current state-of-the-art in the SL research must be pushed further to a point where powerful sign retrieval tools can be used easily by the Deaf in their edge devices. To this end, we studied the keyword search problem in sign language that can be utilized to retrieve SL videos from an archive via a written query.

In this thesis, we presented improvements to an existing keyword search system in sign language and proposed an iterative approach to the problem. We first introduced a new graph encoder comprised of ST-GAT layers. We improve the existing architecture by incorporating a learnable mask and a separable temporal attention mechanism. Although ST-GAT architecture was explored for action recognition, this work is the first to employ the architecture in an SL problem. Later on, we demonstrated the superiority of the proposed modifications by an ablation study. The most and least significant keypoints are determined by inspecting the learned coefficients in the mask, which can be used for feature selection. The PRF method, used frequently in speech retrieval, is adapted to the SL-KWS problem. The results are examined by discussing the differences between speech and sign retrieval problems. Similarity-based training methods are introduced, and it was shown that these models might be used to boost handshape models since they yield better AP scores than the classification methods for some queries depending on the sign articulation. As the main contribution of the thesis, we proposed an iterative training approach that allows the model to refine its predictions over time. Additionally, a query encoder and a cross-modal attention mechanism were incorporated into the system. They enable iterative training of the model and the discovery of subtle interactions between the queries and the video sequence. As an extension of this approach, a natural way of combining model predictions was

explored, and the improvement provided by each iteration were reported. It is shown that the iterative training approach significantly improves the results compared to the baseline system. In order to improve query embeddings, experiments were carried out with pre-trained word embeddings for German.

The current system achieves good retrieval performance on the frequently occurring queries in the training set, suggesting that the collection of sizeable general domain datasets might alleviate a significant portion of the issues related to SL tasks. Alternatively, few-shot training techniques can be utilized in the system to improve the performance in the less-frequent queries. The current system does not provide any information about the temporal position of the query within the utterance. Therefore, the system might be advanced to localize the signs within the sequence.

REFERENCES

1. Sandler, W. and D. Lillo-Martin, "Natural Sign Languages", *The Handbook of Linguistics*, pp. 533–562, Oxford, UK, 2003.

2. "Deaf Employment Reports", https://www.gallaudet.edu/office-of-international-affairs/demographics/deaf-employment-reports/, accessed in June 2022.

3. Tamer, N. C. and M. Saraçlar, "Keyword Search for Sign Language", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8184–8188, Barcelona, Spain, 2020.

4. Tamer, N. C. and M. Saraçlar, "Cross-Lingual Keyword Search for Sign Language", *Proceedings of the LREC 9th Workshop on the Representation and Processing of Sign Languages: Sign Language Resources in the Service of the Language Community, Technological Challenges and Application Perspectives*, pp. 217–223, Marseille, France, 2020.

5. Tamer, N. C. and M. Saraçlar, "Improving Keyword Search Performance in Sign Language with Hand Shape Features", *Computer Vision – ECCV Workshops*, pp. 322–333, Glasgow, UK, 2020.

6. Tamer, N. C., *Keyword Search for Sign Language*, Master's Thesis, Bogazici University, 2020.

7. Cheok, M., Z. Omar and M. Jaward, "A Review of Hand Gesture and Sign Language Recognition Techniques", *International Journal of Machine Learning and Cybernetics*, Vol. 10, No. 1, pp. 131–153, 2019.

8. Sykora, P., P. Kamencay and R. Hudec, "Comparison of SIFT and SURF Methods for Use on Hand Gesture Recognition Based on Depth Map", *AASRI Procedia*, Vol. 9, pp. 19–24, 2014.

9. Tharwat, A., T. Gaber, A. E. Hassanien, M. K. Shahin and B. Refaat, "SIFT-Based Arabic Sign Language Recognition System", *The Proceedings of Afro-European Conference for Industrial Advancement*, pp. 359–370, Addis Ababa, Ethiopia, 2015.

10. Shukla, J. and A. Dwivedi, "A Method for Hand Gesture Recognition", *Fourth International Conference on Communication Systems and Network Technologies*, pp. 919–923, Bhopal, India, 2014.

11. Rekha, J., J. Bhattacharya and S. Majumder, "Shape, Texture and Local Movement Hand Gesture Features for Indian Sign Language Recognition", *3rd International Conference on Trendz in Information Sciences Computing (TISC)*, pp. 30–35, Chennai, India, 2011.

12. Karami, A., B. Zanj and A. K. Sarkaleh, "Persian Sign Language (PSL) Recognition Using Wavelet Transform and Neural Networks", *Expert Systems with Applications*, Vol. 38, No. 3, pp. 2661–2667, 2011.

13. Rekha, J., J. Bhattacharya and S. Majumde, "Hand Gesture Recognition for Sign Language: A New Hybrid Approach", *IPCV : Proceedings of the International Conference on Image Processing, Computer Vision, & Pattern Recognition*, pp. 80–86, Las Vegas, NV, 2011.

14. Krizhevsky, A., I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", *Advances in Neural Information Processing Systems*, Vol. 25, pp. 84–90, Lake Tahoe, NV, 2012.

15. Simonyan, K. and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", *International Conference on Learning Representations*, pp. 1–14, San Diego, CA, 2015.

16. Deng, X., S. Yang, Y. Zhang, P. Tan, L. Chang and H. Wang, "Hand3d: Hand Pose Estimation using 3D Neural Network", *arXiv preprint arXiv:1704.02224*, 2017.

17. Escobedo Cardenas, E. J. and G. C. Chavez, "Multimodal Hand Gesture Recognition Combining Temporal and Pose Information Based on CNN Descriptors and Histogram of Cumulative Magnitudes", *Journal of Visual Communication and Image Representation*, Vol. 71, p. 102772, 2020.

18. Koller, O., 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", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3793–3802, Las Vegas, NV, 2016.

19. Rastgoo, R., K. Kiani and S. Escalera, "Sign Language Recognition: A Deep Survey", *Expert Systems with Applications*, Vol. 164, p. 113794, 2021.

20. Pu, J., W. Zhou and H. Li, "Sign Language Recognition with Multi-modal Features", *Advances in Multimedia Information Processing - PCM*, pp. 252–261, Xi´an, China, 2016.

21. Ge, L., H. Liang, J. Yuan and D. Thalmann, "Robust 3D Hand Pose Estimation in Single Depth Images: From Single-View CNN to Multi-View CNNs", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3593–3601, Las Vegas, NV, 2016.

22. Zheng, C., W. Wu, C. Chen, T. Yang, S. Zhu, J. Shen, N. Kehtarnavaz and M. Shah, "Deep Learning-Based Human Pose Estimation: A Survey", *arXiv e-prints*, 2020.

23. Newell, A., K. Yang and J. Deng, "Stacked Hourglass Networks for Human Pose Estimation", *European Conference on Computer Vision*, pp. 483–499, Amsterdam, The Netherlands, 2016.

24. Wei, S.-E., V. Ramakrishna, T. Kanade and Y. Sheikh, "Convolutional Pose Machines", *Proceedings of the IEEE Conference on Computer Vision and Pattern*

Recognition, pp. 4724–4732, Las Vegas, NV, 2016.

25. Er-Rady, A., R. Faizi, R. O. H. Thami and H. Housni, "Automatic Sign Language Recognition: A Survey", *International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*, pp. 1–7, Fez, Morocco, 2017.

26. Cooper, H., 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.

27. Elmezain, M., A. Al-Hamadi and B. Michaelis, "Real-Time Capable System for Hand Gesture Recognition Using Hidden Markov Models in Stereo Color Image Sequences", *Journal of WSCG*, Vol. 16, pp. 65–72, 2008.

28. Wilson, A. and A. Bobick, "Parametric Hidden Markov Models for Gesture Recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 21, No. 9, pp. 884–900, 1999.

29. Hong, P., M. Turk and T. Huang, "Gesture Modeling and Recognition Using Finite State Machines", *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 410–415, Grenoble, France, 2000.

30. Adaloglou, N., T. Chatzis, I. Papastratis, A. Stergioulas, G. T. Papadopoulos, V. Zacharopoulou, G. J. Xydopoulos, K. Atzakas, D. Papazachariou and P. Daras, "A Comprehensive Study on Deep Learning-Based Methods for Sign Language Recognition", *IEEE Transactions on Multimedia*, Vol. 24, No. 1, pp. 1750–1762, 2022.

31. Liu, H., S. Jin and C. Zhang, "Connectionist Temporal Classification with Maximum Entropy Regularization", *Advances in Neural Information Processing Systems*, Vol. 31, pp. 839–849, Montreal, Canada, 2018.

32. Heymann, J., K. C. Sim and B. Li, "Improving CTC Using Stimulated Learning

for Sequence Modeling", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5701–5705, Brighton, UK, 2019.

33. Koller, O., S. Zargaran, H. Ney and R. Bowden, "Deep Sign: Hybrid CNN-HMM for Continuous Sign Language Recognition", *British Machine Vision Conference*, pp. 136.1–136.12, York, UK, 2016.

34. Koller, O., N. C. Camgoz, H. Ney and R. Bowden, "Weakly Supervised Learning with Multi-Stream CNN-LSTM-HMMs to Discover Sequential Parallelism in Sign Language Videos", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 42, No. 9, pp. 2306–2320, 2020.

35. Cui, R., H. Liu and C. Zhang, "Recurrent Convolutional Neural Networks for Continuous Sign Language Recognition by Staged Optimization", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1610–1618, Honolulu, HI, 2017.

36. Cui, R., H. Liu and C. Zhang, "A Deep Neural Framework for Continuous Sign Language Recognition by Iterative Training", *IEEE Transactions on Multimedia*, Vol. 21, No. 7, pp. 1880–1891, 2019.

37. Tran, D., L. Bourdev, R. Fergus, L. Torresani and M. Paluri, "Learning Spatiotemporal Features with 3D Convolutional Networks", *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 4489–4497, Santiago, Chile, 2015.

38. Huang, J., W. Zhou, Q. Zhang, H. Li and W. Li, "Video-Based Sign Language Recognition without Temporal Segmentation", *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, pp. 2257–2264, New Orleans, LA, 2018.

39. Carreira, J. and A. Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4724–4733, Honolulu, HI, 2017.

40. Joze, H. V. and O. Koller, "MS-ASL: A Large-Scale Data Set and Benchmark for Understanding American Sign Language", *The British Machine Vision Conference*, p. 100, Cardiff, UK, 2019.

41. Pu, J., W. Zhou and H. Li, "Dilated Convolutional Network with Iterative Optimization for Continuous Sign Language Recognition", *Electronic Proceedings of IJCAI*, pp. 885–891, Stockholm, Sweden, 2018.

42. Yan, S., Y. Xiong and D. Lin, "Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition", *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32, pp. 7444–7452, New Orleans, LA, 2018.

43. de Amorim, C. C., D. Macêdo and C. Zanchettin, "Spatial-Temporal Graph Convolutional Networks for Sign Language Recognition", *Artificial Neural Networks and Machine Learning – ICANN: Workshop and Special Sessions*, pp. 646–657, Munich, Germany, 2019.

44. Liang, W. and X. Xu, "Skeleton-Based Sign Language Recognition with Attention-Enhanced Graph Convolutional Networks", *CCF International Conference on Natural Language Processing and Chinese Computing*, pp. 773–785, Qingdao, China, 2021.

45. Jiang, S., B. Sun, L. Wang, Y. Bai, K. Li and Y. Fu, "Skeleton Aware Multi-Modal Sign Language Recognition", *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3413–3423, Nashville, TN, 2021.

46. Parelli, M., K. Papadimitriou, G. Potamianos, G. Pavlakos and P. Maragos, "Spatio-Temporal Graph Convolutional Networks for Continuous Sign Language

Recognition", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8457–8461, Singapore, 2022.

47. Liu, Z., H. Zhang, Z. Chen, Z. Wang and W. Ouyang, "Disentangling and Unifying Graph Convolutions for Skeleton-Based Action Recognition", *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 143–152, Seattle, WA, 2020.

48. Vazquez-Enriquez, M., J. L. Alba-Castro, L. Docio-Fernandez and E. Rodriguez-Banga, "Isolated Sign Language Recognition with Multi-Scale Spatial-Temporal Graph Convolutional Networks", *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 3457–3466, Nashville, TN, 2021.

49. Stoll, S., N. C. Camgöz, S. Hadfield and R. Bowden, "Sign Language Production Using Neural Machine Translation and Generative Adversarial Networks", *Proceedings of the 29th British Machine Vision Conference (BMVC)*, p. 304, Newcastle, UK, 2018.

50. Stoll, S., N. C. Camgoz, S. Hadfield and R. Bowden, "Text2Sign: Towards Sign Language Production Using Neural Machine Translation and Generative Adversarial Networks", *International Journal of Computer Vision*, Vol. 128, No. 4, pp. 891–908, 2020.

51. Camgoz, N. C., S. Hadfield, O. Koller, H. Ney and R. Bowden, "Neural Sign Language Translation", *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7784–7793, Salt Lake City, UT, 2018.

52. Camgöz, N. C., O. Koller, S. Hadfield and R. Bowden, "Sign Language Transformers: Joint End-to-End Sign Language Recognition and Translation", *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10023–10033, Seattle, WA, 2020.

53. Saraçlar, M. and R. Sproat, "Lattice-Based Search for Spoken Utterance Retrieval", *North American Chapter of the Association for Computational Linguistics*, pp. 129–136, Boston, MA, 2004.

54. Lee, L.-s., J. Glass, H.-y. Lee and C.-a. Chan, "Spoken Content Retrieval—Beyond Cascading Speech Recognition with Text Retrieval", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, Vol. 23, No. 9, pp. 1389–1420, 2015.

55. Anguera, X. and M. Ferrarons, "Memory Efficient Subsequence DTW for Query-by-Example Spoken Term Detection", *IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, San Jose, CA, 2013.

56. Zhang, Y. and J. R. Glass, "Towards Multi-Speaker Unsupervised Speech Pattern Discovery", *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4366–4369, Dallas, TX, 2010.

57. Sarı, L., B. Gündoğdu and M. Saraçlar, "Fusion of LVCSR and Posteriorgram Based Keyword Search", *INTERSPEECH*, pp. 824–828, Dresden, Germany, 2015.

58. López-Espejo, I., Z.-H. Tan, J. H. L. Hansen and J. Jensen, "Deep Spoken Keyword Spotting: An Overview", *IEEE Access*, Vol. 10, pp. 4169–4199, 2022.

59. Chen, G., C. Parada and G. Heigold, "Small-Footprint Keyword Spotting Using Deep Neural Networks", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4087–4091, Florence, Italy, 2014.

60. Zhuang, Y., X. Chang, Y. Qian and K. Yu, "Unrestricted Vocabulary Keyword Spotting Using LSTM-CTC", *INTERSPEECH*, pp. 938–942, San Francisco, CA, 2016.

61. Rybakov, O., N. Kononenko, N. A. Subrahmanya, M. Visontai and S. Laurenzo, "Streaming Keyword Spotting on Mobile Devices", *INTERSPEECH*, pp. 2277–2281, Shanghai, China, 2020.

62. Sutskever, I., O. Vinyals and Q. V. Le, "Sequence to Sequence Learning with Neural Networks", *Proceedings of the 27th International Conference on Neural Information Processing Systems*, Vol. 2, pp. 3104–3112, Montreal, Canada, 2014.

63. He, Y., R. Prabhavalkar, K. Rao, W. Li, A. Bakhtin and I. McGraw, "Streaming Small-Footprint Keyword Spotting Using Sequence-to-Sequence Models", *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 474–481, Okinawa, Japan, 2017.

64. Viitaniemi, V., T. Jantunen, L. Savolainen, M. Karppa and J. Laaksonen, "S-Pot - a Benchmark in Spotting Signs within Continuous Signing", *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, pp. 1892–1897, Reykjavik, Iceland, 2014.

65. Buehler, P., A. Zisserman and M. Everingham, "Learning Sign Language by Watching TV (Using Weakly Aligned Subtitles)", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2961–2968, Miami, FL, 2009.

66. Pfister, T., J. Charles and A. Zisserman, "Large-Scale Learning of Sign Language by Watching TV (Using Co-occurrences)", *British Machine Vision Conference*, pp. 20.1–20.11, Bristol, UK, 2013.

67. Momeni, L., G. Varol, S. Albanie, T. Afouras and A. Zisserman, "Watch, Read and Lookup: Learning to Spot Signs from Multiple Supervisors", *Asian Conference on Computer Vision*, pp. 291–308, Singapore, 2021.

68. Cao, Z., G. Hidalgo, T. Simon, S.-E. Wei and Y. Sheikh, "OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 43, No. 1, pp. 172–186, 2021.

69. Orbay, A. and L. Akarun, "Neural Sign Language Translation by Learning Tokenization", *15th IEEE International Conference on Automatic Face and Gesture*

Recognition, pp. 222–228, 2020.

70. Siyli, D., "HospiSign: A Framewise Annotated Isolated Turkish Sign Language Dataset", http://dogasiyli.com/hospisign/, accessed in June 2022.

71. Mikolov, T., K. Chen, G. Corrado and J. Dean, "Efficient Estimation of Word Representations in Vector Space", *1st International Conference on Learning Representations Workshop Track Proceedings*, pp. 1–12, Scottsdale, AZ, 2013.

72. "German Word Embeddings — Deepset", https://www.deepset.ai/german-word-embeddings, accessed in March 2021.

73. Bojanowski, P., E. Grave, A. Joulin and T. Mikolov, "Enriching Word Vectors with Subword Information", *Transactions of the Association for Computational Linguistics*, Vol. 5, pp. 135–146, 2017.

74. Grave, E., P. Bojanowski, P. Gupta, A. Joulin and T. Mikolov, "Learning Word Vectors for 157 Languages", *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*, pp. 1–5, Miyazaki, Japan, 2018.

75. Veličković, P., G. Cucurull, A. Casanova, A. Romero, P. Liò and Y. Bengio, "Graph Attention Networks", *International Conference on Learning Representations*, pp. 1–12, Vancouver, Canada, 2018.

76. Huang, Q., F. Zhou, J. He, Y. Zhao and R. Qin, "Spatial–Temporal Graph Attention Networks for Skeleton-Based Action Recognition", *Journal of Electronic Imaging*, Vol. 29, No. 5, pp. 1–15, 2020.

77. Rocchio, J. J., "Relevance Feedback in Information Retrieval", *The Smart Retrieval System : Experiments in Automatic Document Processing*, pp. 313–323, 1971.

78. Yan, R., A. G. Hauptmann and R. Jin, "Negative Pseudo-Relevance Feedback in Content-Based Video Retrieval", *Proceedings of the Eleventh ACM International*

Conference on Multimedia, pp. 343–346, Seattle, WA, 2003.

79. Chen, C.-p., H.-y. Lee, C.-f. Yeh and L.-s. Lee, "Improved Spoken Term Detection by Feature Space Pseudo-Relevance Feedback", *INTERSPEECH*, p. 4, Makuhari, Japan, 2010.

80. Lee, H.-y., C.-p. Chen and L.-s. Lee, "Integrating Recognition and Retrieval With Relevance Feedback for Spoken Term Detection", *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 20, No. 7, pp. 2095–2110, 2012.

81. Goldberger, J., G. E. Hinton, S. Roweis and R. R. Salakhutdinov, "Neighbourhood Components Analysis", *Advances in Neural Information Processing Systems*, Vol. 17, pp. 513–520, Vancouver, Canada, 2004.

82. Gündoğdu, B. and M. Saraçlar, "Novel Score Normalization Methods for Keyword Search", *25th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, Antalya, Turkey, 2017.

83. Hoffer, E. and N. Ailon, "Deep Metric Learning Using Triplet Network", *International Workshop on Similarity-Based Pattern Recognition*, pp. 84–92, Copenhagen, Denmark, 2015.

84. Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, "Attention Is All You Need", *Neural Information Processing Systems*, pp. 5998–6008, Long Beach, CA, 2017.

85. Liu, D., X. Qu, J. Dong and P. Zhou, "Adaptive Proposal Generation Network for Temporal Sentence Localization in Videos", *Empirical Methods in Natural Language Processing*, pp. 9292–9301, Punta Cana, Dominican Republic, 2021.

86. Koller, O., 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.

87. “SignDict”, http://signdict.org/, accessed in June 2022.