SCORE LEVEL MULTI CUE FUSION FOR SIGN LANGUAGE RECOGNITION

by

Çağrı Gökçe

B.S., Computer Engineering, İstanbul Technical University, 2017

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 Computer Engineering

Boğaziçi University

2020

ACKNOWLEDGEMENTS

Firstly, I would like to thank my thesis advisor Prof. Lale Akarun for her expertise, guidance, assistance, and patience during my masters education. I would like to thank Assist. Prof. Berk Gökberk and Assist. Prof. İnci Meliha Baytaş for participating in my thesis jury and sharing their valuable comments and suggestions.

I would like to thank my collaborators Oğulcan Özdemir and Ahmet Alp Kındıroğlu for their guidance and help during the thesis writing process. I also would like to thank Boğaziçi University Perceptual Intelligence Laboratory members for their endless support and valuable friendship, namely, Ahmet Alp Kındıroğlu, Alptekin Orbay, Doğa Siylı, Gizem Esra Ünlü, Mehmet Burak Kurutmaz, Oğulcan Özdemir, Ufuk Can Biçici, and Uras Mutlu. I would also like to thank research assistants Ahmet Tekden and Meriç Turan for valuable friendship and their support during the thesis writing process.

ABSTRACT

SCORE LEVEL MULTI CUE FUSION FOR SIGN LANGUAGE RECOGNITION

In this thesis, we propose a Score-Level Multi Cue Fusion approach that improves the sign language recognition performance of the three dimensional convolutional neural networks. Sign Language is the communication language of the Deaf and Hearing-impaired individuals and performed using hand movements, facial gestures, and body alignment. Sign Language Recognition is the task that aims to understand sign language and gaining increasing popularity with the task becoming feasible due to the efficiency of the neural network. Previous work uses 3D CNN network variants to inspect SL properties in different settings. The vanilla 3D variant uses 3D kernels with high processing cost, the mixed convolution variant applies both 3D and 2D kernels respectively, and R(2+1)D variants exploit bottleneck connections to exploit the bottleneck dimension. Various studies use these networks to generate an end to end framework for tasks such as sign classification and translation. To achieve better performance, 3D CNN methods use the complicated neural network architectures that have a branch for every cue system. We evaluate the 3D network performances and propose a more straightforward approach which only adopts a single neural network that can process multiple cues at test time. We exploit the hand, body, and face cues by training single individual networks and fuse results by using a weighted score fusion. We test our method on the recently published Turkish Isolated SLR dataset. Despite the simple architecture, our method achieves %94 percent classification rate on 744 different sign glosses. We hope that the multi cue approach can help with the other SLR tasks such as translation, which is stated as future work.

ÖZET

İŞARET DİLİ TANIMA İÇİN SONUÇ SEVİYESİNDE ÇOKLU İPUCU KAYNAŞIMI

Bu tezde işaret dili tanıma problemi için kullanılan Üç boyutlu Evrişimsel Sinir Ağları (3B ESA) yapılarının sınıflandırma performansını artıran Skor Seviyesinde Çoklu İpucu Kaynaşımı yöntemi önerilmiştir. İşaret Dili yazılı dil içeriğinden farklı olarak çoğunlukla el hareketleri, vücut pozu ve yüz ifadelerine odaklandığı için bu bölgelere ait ipuçları önem teşkil etmektedir. İşaret Dili Tanıma için son dönemde derin yapay sinir ağlarının gelişmesi sebebiyle bir çok yöntem önerildiği görülmektedir. Önerilmiş olan yöntemler farklı 3B ESA yapılarını kullanarak işaret dili sınıflandırması yapmaktadır ve sırasıyla Standart 3B ESA, Karma ESA ve (2+1)B ESA methodları olarak sınıflandırılmıştır. Standart 3B ESA yapısı sadece 3B filtresini kullanmakta, Karma ESA yapısı 2B ve 3B filtrelerini birlikte kullanmakta ve (2+1)B ESA yapısı ESA atlama mimarisinde darboğaz oluşturan (2+1)B bloğunu kullanmaktadır. İşaret dili tanıma problemi için bahsedilen yapıların kullanıldığı çalışmalar bulunmaktadır, ancak bu çalışmalarda işaret dilinin farklı vücut, el ve yüz gibi farklı ipucu bölgelerinin değerlendirilmediği veya gereksiz büyüklükte mimariler kullanıldığı için zayıf sonuçlar elde edildiği görülmektedir. Bu eksikliklerin çözülmesi kolayca uygulanabilen ve yalnızca Karma ESA yapısı ile tüm işaret dili ipuçlarını işleyebilen bir sistem tasarlanmıştır. Bu sistem ile el, vücut ve yüz ipucu modelleri karma ESA yapısı ile eğitilmiştir ve sonuçlar ağırlıklı skor kaynaşımı ile birleştirilmiştir. Bosphorus Sign 22k Türk İşaret Dili veri kümesi üzerinde deneyler yapılmıştır. Tasarladığımız sistem alternatiflerine göre daha hızlı çalışmakta ve 744 sınıf tahmini probleminde %94 sınıflandırma başarısı ile diğerlerinden daha başarılı sonuç elde etmektedir. Gelecek çalışmalarda önerdiğimiz çoklu ipucu yapısı kullanılarak, işaret dili çevirisi gibi diğer işaret dili temelli problemlerdeki başarımın da artırılabileceği öngörülmektedir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . iii
ABSTRACT . . . iv
ÖZET . . . v
LIST OF FIGURES . . . viii
LIST OF TABLES . . . xii
LIST OF SYMBOLS . . . xiv
LIST OF ACRONYMS/ABBREVIATIONS . . . xvi
1. INTRODUCTION . . . 1
1.1. Contributions . . . 4
1.2. Organization of the Thesis . . . 6
2. SIGN LANGUAGE RECOGNITION . . . 7
2.1. Convolutional Neural Networks . . . 7
2.2. Video Recognition . . . 11
2.3. Sign Language Recognition . . . 15
2.3.1. Sign Language Recognition Datasets . . . 16
2.3.2. Sign Language Recognition Methods . . . 17
3. SPATIAL SAMPLING . . . 20
3.1. Body Model . . . 20
3.2. Hand Model . . . 21
3.2.1. Hand Movement Detection . . . 21
3.2.2. Hand Cue Settings . . . 23
3.3. Face Model . . . 25
4. TEMPORAL SAMPLING . . . 26
5. MULTI CUE FUSION . . . 29
5.1. Score Level Fusion . . . 29
5.1.1. Standard Score Fusion . . . 29
5.1.2. Weighted Score Fusion . . . 30
5.1.3. Failed Settings . . . 31

6. EXPERIMENTS AND RESULTS 37
6.1. Experiment Setting 37
6.2. Cue Sampling and Score-Level Fusion 38
6.3. Spatio-Temporal Multi-Cue Fusion 40
6.4. Discussion and Analysis 42
6.4.1. Spatial Ablation Study 43
6.4.2. Fusion Model Evaluation 47
7. CONCLUSION 49
REFERENCES 51

LIST OF FIGURES

Figure 2.1. Residual architectures proposed by Tran et al. [1] visualised. While R2D and R3D are using a single block type, others use a combination of different scale convolution operations. 14

Figure 3.1. Hand cue movement on a sample video in Bosphorus Sign 22k dataset is visualized using the optical flow algorithm. 20

Figure 3.2. Hand movement detection process is visualized. We use the 9th hand center keypoint as an anchor in the first frame. Then we start the iterate the following frames. We calculate the current frame keypoint distance compared to the anchor. If the distance is bigger than threshold, we declare the hand as moving. 22

Figure 3.3. Hand movement detection applied on a frame where left hand movement occurs. We compare the anchor position and current hand keypoint position. Since the distance is bigger than the threshold, we detect the left hand as moving. 22

Figure 3.4. Hand movement detection applied on a frame where right hand does not show any movement. We calculate current keypoint position and compare with the anchor. Since there is barely any movement, and the distance is smaller than the threshold, we conclude that the right hand is not moving. 23

Figure 3.5. Spatial Sampling operation is visualized. From left to right; cue regions selected for the process, and hand crop settings. 24

Figure 4.1. Different temporal sampling operations are shown in the above figure. Selected frames are shown with color. Two branches represent uniform sampling and the Active Window Based Sampling Process. 26

Figure 4.2. Active window detection algorithm is visualized. We start by using first frame anchor and iterate over following frames and detect the frame where the first movement occurs. We apply the process in reverse by starting with the last frame and with reverse iteration. Finally we declare the part between the first move and inverse move as the active window. 27

Figure 4.3. Exceptions in the active window detection process. The first exception is hand already starting in the signing position. We detect such occurrences by checking the distance between hand(4) and hip(8) keypoints. The second exception occurs when there is poor hand detection or irregular hand movements. We discard the sampling in the following occurrences. 28

Figure 5.1. Score-level multi cue fusion operation applied at the test time. Note that cue networks have different test weights even the architecture is same. 29

Figure 5.2. Weighted score fusion approach is visualized. After getting the each cue softmax activations, we firstly multiply them with the each model's validation accuracy. Secondly, we calculate the sum of scores, and finally we divide the scores by the total weight. . . . 30

Figure 5.3. Model selection for the self-fusion is explained. For the sample CNN training, we calculate the validation accuracy after each training epoch (leftmost table) and provide the corresponding validation graph. We select the checkpoints that increase the best previous accuracy and save their model parameters. Finally, we extract these saved model parameters at corresponding epochs as different models as seen in the bottom part of the image. 33

Figure 6.1. Our sampling and fusion pipeline is visualised. We divide our approach into three stages, spatial sampling, temporal sampling and score-level fusion. 39

Figure 6.2. Comparison of the models in the Top-N accuracy setting. The horizontal axis denotes the increasing N value, and the vertical axis denotes the accuracy value. The first plot shows the Single Cue Setting comparison, and the second plot shows the Multi Cue Setting additive comparison. Despite the difference in single setting performance, each cue boosts the fusion results. 42

Figure 6.3. Class confusions of the EXCHANGE(v) sign gloss for the Body cue model. 45

Figure 6.4. Class confusions of the IDENTIFY(v) sign gloss for the Body cue model. 46

Figure 6.5. Modified confusion matrices of the top ten classes where the Hand model has an advantage over the Body Model. The rows represent the top ten classes, and the first ten columns also correspond to the same classes. The color strength represents the number of total predictions of a given column, where the row is the ground truth. The diagonal starting from $0, 0$ to $10, 10$ represents the true predictions. The rest of the column data represent other confused classes. 48

LIST OF TABLES

Table 2.1. Sign Language Recognition Datasets and their respective properties. Asterisk (∗) mark is used to denote that signers are native. Pose column represents if authors have provided the corresponding depth and pose data. 16

Table 3.1. Hand activity distribution in the BS22k dataset detected through movement estimation algorithm. 24

Table 5.1. Self Fusion Results on the single Body cue setting. Models' identities are ranging from 0 to 9, which represents the fusion pool, sorted by their single validation set performance in the descending order. Different subset lengths (n_{sub}) are evaluated and found the best performing subsets on the validation set. Results show the score fusion and weighted score fusion accuracy results of the given subsets. 34

Table 6.1. Test results of the different hand crop settings and resulting accuracy values. 39

Table 6.2. Classification accuracy results of the sampling and fusion settings. Three different settings are provided in the table. From left to right, (1) Single cue spatial sampling results, (2) Active Window Based Temporal Sampling applied to each crop, and (3) Spatial & Temporal settings are combined in one setting. Note that the bottom two rows include the fusion result of the above three models in each setting. 41

Table 6.3. Comparison with the baseline approaches IDT and MC3_18 model. 41

Table 6.4. Effects of excluding individual cue units from the final fusion model. Using the different two cue settings and their performance, we infer to the excluded setting and its effect on the final mix. 43

Table 6.5. F1-score comparison for the top ten sign glosses that hand sampling outperforms body sampling. (Sorted in the alphabetical order) . . 44

LIST OF SYMBOLS

$\star$	Convolution operator
$\|W\|_1$	L1 norm of the tensor W
$\|W\|_2$	Euclidean norm of the tensor W
A_N	Anchor value for the region N
a_{ij}	Description of a_{ij}
b	Bias term
C	Regularization parameter for the SVM classifier
$\partial a/\partial b$	Partial derivative
E	Expected Value
$exp(n)$	Exponential function of the value n
f_i	Function applied at the network layer i
J	Objective function
$log(n)$	Logarithmic function of the value n
$max(a, b)$	Maximum value function of a and b
N_M	Number of models in the self fusion pool
n_{sub}	Subset length selected for self fusion
$< P_1, P_2... >$	Model pool values for self fusion
$p(a\|b)$	Conditional probability of a given b
p_D	Probability of the node inactivity for Dropout
$softmax(n)$	Softmax function of the value n
T_N	Threshold value of the task N
X_i	Input data for the network layer i
y_i	Output of the network layer i
λ	Regularization control parameter
Ω	Regularization function
Θ	Neural network parameter set
Δ_a	Gradient vector with respect to a

η	Learning rate

LIST OF ACRONYMS/ABBREVIATIONS

2D	Two Dimensional
3D	Three Dimensional
ADAM	Adaptive Momentum
ANN	Artificial Neural Network
ASL	American Sign Language
AvgPool	Average Pooling Layer
AW	Active Window
BN	Batch Normalization
CNN	Convolutional Neural Network
CSLR	Continuous Sign Language Recognition
CSL	Chinese Sign Language
CTC	Connectionist Temporal Classification
DL	Deep Learning
DT	Dense Trajectories
FC	Fully Connected Layer
GA	Genetic Algorithm
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
GSL	German Sign Language
HMM	Hidden Markov Model
HMU	Hand Motion Understanding
HOG	Histogram of Oriented Gradients
I3D	Two-Stream Inflated 3D Convolutional Network
IDT	Improved Dense Trajectories
IN	Instance Normalization
ISLR	Isolated Sign Language Recognition
KL	Kullback–Leibler
LDA	Linear Discriminant Analysis

LSTM	Long Short-term Memory
ML	Machine Learning
MLP	Multilayer Perceptron
NLP	Natural Language Processing
PaHMMs	Parallel Hidden Markov Models
PCA	Principal Component Analysis
PSO	Particle Swarm Optimization
R2D	Residual Convolutional Network
R3D	3D Residual Convolutional Network
R(2+1)D	2+1 Block Residual Convolutional Network
RMC_x	Mixed Residual Convolutional Network
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SRN	Improved Simple Recurrent Network
SVM	Support Vector Machine
TanH	Hyperbolic Tangent
TSL	Turkish Sign Language
XGBoost	Extreme Gradient Boosting

1. INTRODUCTION

Sign language is the main medium of communication among the hearing impaired community. Sign Languages differ from the spoken language in terms of grammar structure and linguistic rules. This makes understanding and translating the sign languages hard tasks since those require human expertise. Acquiring human SL expertise is a long and tedious process, and there is a significant increase in digital data without accessibility to Deaf individuals. There should be a system or pipeline that extract patterns of the SL and then apply the recognition. We built our research on the Sign Language Recognition (SLR). It involves many different tasks such as SL understanding and classification, and aim to close the accessibility gap in real life and online systems.

Sign language is expressed through the shape of hands as well as other cues, such as hand movements, body posture, and facial gestures. Unlike a video recognition system, an SLR system needs to build on given fine details, and needs to capture smaller cues that are provided in the context of communication. Thus, it requires a specially tailored solution that requires a complex mix of carefully handcrafted methods and state-of-the-art machine learning techniques.

There are many challenges that need to be considered throughout the design of the SLR system. Sign languages differ between each community and culture; thus, this creates a gap between communication. Different SLR systems are proposed for American, British, Chinese, and Turkish Sign Languages to investigate the different sign languages individually. While language-independent sign language analysis is a future goal, the current state of the research is built with a focus on language-specific cues.

Sign languages can be performed by multiple individuals that have their own cue patterns, and this presents a novel challenge. SLR systems need to be signer-independent since the aim is to extract the information about the provided sign pattern.

Another problem is the big gap in sign language and the spoken counterpart, which makes it impossible to achieve a one-to-one mapping between sign phrases and the text, which not only makes the SLR task harder, it also makes even the data labeling or designing a rule-based system almost impossible to have. To ease the data labeling, sign glosses are used which are achieved by independently labeling signs or sign phrases are to their closest spoken language counterpart.

According to the tasks involved with signed language recognition, we can classify the previous work into two categories, Isolated and Continuous Sign Language Recognition. Isolated SLR aims to distinguish the isolated words or word phrases performed in isolation. However, continuous SLR aims to distinguish continuous SL phrases performed after another, which can range between a few short sentences, a short story, to an SL translation of a full-length movie. Domain-specific challenges stem in each category that shows the critical properties of the task.

To tackle the SLR problem, researchers have published datasets with a focus on either Isolated or Continuous SLR. As common in both tasks, dataset providers share the data in the video format and additionally provide the 3D depth heatmap with pose estimation data. Isolated datasets involve multiple users performing hundreds of sign gloss categories, which are labeled with a single sign gloss and shared in a selected language such as ASL [2, 3], CSL [4] and TSL [5]. Continuous datasets include each video that is segmented into the active gloss parts with the respective label and also shared in a selected language, eg. GSL [6].

A straightforward approach for inspecting SL signs, is achieved by evaluating the possible SL cues and building a system based on the extracted patterns of the cues. One method is inspecting the problem as a gesture recognition task and focusing on the face region, especially on mouth and eye regions. This approach deems to fail since the linguistic features are not evaluated, and the main form of communication via the hand region is discarded. Analysis of the hand region requires a complex hand shape and activity tracking, which is expensive in terms of processing since data size is more

significant compared to the other computer vision tasks.

To inspect the SLR data with a focus on the temporal dimension of the video and the flow between each frame, researchers have proposed handcrafted methods [7–12]. These methods, such as IDT [13] stems from the information that a temporal data should include a temporal variation and that variation can be extracted as a pattern. These approaches are widely used in the video recognition or action recognition tasks [14, 15] , and also adopted in the SLR literature [16–20] due to their proven empirical performance of extracting temporal information. In applications, handcrafted methods are usually paired with a final stage ML-based classifier to exploit the extracted patterns [21–23].

Increasing computational power and the rise of the deep neural networks made end-to-end solutions viable for many tasks along with SLR [24–27]. The SLR community increased focus on the video-based solutions, such as 3D convolution-based residual networks [2, 4, 28, 29]. Increased data size and availability of processing power gave rise to more reliable empirical results, one step closer to achieving the fully automated SLR task.

Neural network-based methods for SLR perform well on some tasks such as translation and perform state-of-the-art performance. However, due to the nature of the deep networks, the method needs to be trimmed to a specific task itself and sometimes this results in complex architectures and unnecessarily long pipelines compared to their handcrafted counterparts. Another problem with the end to end approaches is the lack of insight involved in the system design. Because of the one-stage methodology, there needs to be a mechanism that can include the vital prior knowledge of SLR into the system, such as hand regions and hand shapes.

In this thesis, we aim to inspect the shortcomings of the 3D CNNs methods compared to the handcrafted variants and try to improve them in terms of task accuracy, cue localization, and performance. Inspecting the empirical results are shared in the

recent work on BosphorusSign 22k dataset [5], authors conclude that the CNN based baseline is performs worse. Our investigations on the cue regions and SLR tailored training pipeline builds on this inspection, and we provide our methodology to solve the given problem and further improve the neural network-based solution.

1.1. Contributions

Our first priority is repeating the baseline CNN experiments. We selected a Turkish SLR dataset Bosphorus Sign 22k [5] and replicated classification results using mixed convolutional residual networks [1], which is an efficient CNN method for classification of the video data and is a practical approach when using multimedia data. We strictly follow the pipeline shared in the baseline experiment and repeat it using a given dataset.

Our contribution starts with evaluating multiple cues in the SLR task. We inspect the cues that have big spatial sizes, such as full-body, hand, and face. In the video data, there is a temporal domain that also needs to be evaluated along with the spatial region, so we focus on the covariance between the two domains by evaluating each cue between frames by evaluating a continuous pattern. For this purpose, we take the most active cue, hand region, and design a tracking system based on the Euclidean distance. Using the tracking data, we are able to determine the active segmented regions.

We build a cue detection pipeline that focuses on the dense regions, the two domains, spatial and the temporal. We use sampling operation on the given two domains. Firstly, spatial sampling is applied to selected regions involving a cropping operation followed by the 2D Bilinear Interpolation. Our experimental results show that the Hand cue region has achieved the best classification performance. Secondly, we exploit the temporal regions on the SLR data. We built an active window detection pipeline based on the dense cue regions on the temporal domain, followed by the standard uniform sampling process. We observe the spatial sampling is most effective when hand cues are used, and overperforming the baseline. Temporal sampling also

boosts the baseline results.

Finally, we search for a mechanism to use both sampling operations to use all of the cues effectively. We combine the multiple expert cue models by applying Score-Level fusion. Score-level fusion is applied through extracting the last layer features and applying arithmetic averaging operation after the softmax operation. We also build a weighted alternative to evaluate each cue performance on its relative success on the validation set. Our weighted fusion setting results in %94 sign classification accuracy for the 744 sign gloss classification problem.

Using the strong empirical performance of the multi cue fusion model, we aim to investigate the effect of each cue on the final fusion model. Our Top-N accuracy based comparison shows that model performances are consistent through increasing prediction counts. Also, we can observe that the multi cue setting benefits the model even when incrementally added to the final results. We suspect the increase of the performance is based on each cue having an increased spatial resolution to the region that operate on. This increased resolution results in extracting the densest features related to the corresponding region. We observe that the hand model extracts the finger shapes and position, whereas the face model captures the lip movements and facial gestures.

We investigate the behavior of cue models in the gloss based setting by comparing their performance gloss by gloss. We show that cue confusions are caused by a lack of hints during the evaluation and correlate with semantic similarities. These semantic similarities cause closely related patterns in the syntactic movements. Each failure hints at fine details such as finger shape, body alignment, and head position. Our proposed method solves the shortcoming of one cue model with addition to the other and manages to be successful in a single gloss setting.

We summarize our contributions in this thesis as follows:

(i) We have adapted the mixed convolutional residual networks on SLR task and replicated the baseline setting
(ii) We have exploited the spatial dense cue regions in multiple settings by applying spatial sampling
(iii) We have detected active windows in the temporal domain by detecting hand movements and applied the temporal sampling
(iv) We have combined the different cues via Score-Level fusion and its weighted variant to improve upon each model

1.2. Organization of the Thesis

The rest of the thesis is organized as follows: In Chapter 2, we explore the background information and the related work for the sign language recognition then proceed with describing SLR datasets and methods. In Chapter 3, we describe our Spatial Sampling setting. In Chapter 4, we describe our Active Window Based Temporal Sampling setting. In Chapter 5, we describe our Multi-Cue Fusion methods, including failure cases. In Chapter 6, we provide the experiment results that we have achieved on the Bosphorus Sign 22k dataset, then discuss the results. Chapter 7 includes the final discussion and possible future work.

2. SIGN LANGUAGE RECOGNITION

In this section, we review the Sign Language recognition methods that we have used in our system. Since SLR methods are closely related to convolutional networks and video recognition, we firstly explore those areas and then proceed with the SLR.

2.1. Convolutional Neural Networks

State-of-the-art SLR methods adopt the convolutional neural networks (CNNs) for the end-to-end image and video processing. We will describe the CNN model and widely used CNN based architectures such as ResNet [30] and Inception [31] in this section. Before diving into the convolutional neural networks, we will briefly describe the feedforward neural networks.

A simple feedforward neural network (multilayer perceptron) applies a linear function at each node in hidden layers for a given input data. [32, 33]

For a given linear function with a single weight and bias values, the single layer output is:

$$f_i(y|W, b) = WX_i + b \tag{2.1}$$

and for the multiple stacked layers, the formula becomes:

$$f_k(y_i|\theta) = f_k(f_{k-1}...f_1(X_i)) \tag{2.2}$$

where θ denotes all of the model's parameters.

An activation function is applied after each node to provide nonlinearity to the system and help the gradient flow for easier training. Sigmoid, TanH and ReLU are

the popular choices as activation functions [34]. ReLU applies the applies following equation:

$$f(y_i|W, b) = max(0, WX_i + b) \tag{2.3}$$

Most networks are trained with negative log-likelihood (cross-entropy) cost function. It can be denoted as follows:

$$J(\theta|y) = -\log P_{output}(y|\theta) \tag{2.4}$$

Depending on the application, we need to customize the output layer and define an objective function. For the binary classification task, where we want to estimate $y = P(y = 1|x)$, Bernoulli distribution is used to obtain valid probability, which leads to the choice of sigmoid function at the output layer [33]. For multiple class classification problems, where we want to estimate $y_i = P(y = i|x)$, we expand it to Multinoulli distribution, with Softmax output units as shown below:

$$softmax(z)_i = \frac{\exp(z_i)}{\sum_j exp(z_j)} \tag{2.5}$$

Sigmoid and softmax functions work as normalized class probabilities. They also negate log term in the cost function with the exponential term, which also scales the output. Their behavior gives a high gradient on an incorrect answer, which results in a more significant gradient step as the desired property in gradient-based learning.

For regression problems, or when any continuous output is desired, the output layer is constructed as a function of desired variables. For example, if we want to estimate properties of Gaussian distribution on input $f(x) \approx G(\mu, \sigma^2)$, we can build the output layer as a function of mean and/or variance [35]. To get rid of the logarithmic term, it is a common practice to estimate log results of standard deviation or variance instead of the original value. Objective function should also be modified accordingly for an estimated continuous property, KL divergence is a typical example [36].

At the training phase, we need to calculate the gradient before the optimization. We use backpropagation [37], which is an efficient algorithm that builds on the chain rule of calculus. During the forward pass, the network builds its output, the cost function is calculated, and then backpropagation will calculate the gradient. Considering the three-layer network with x as input, y and z are hidden layer outputs; the gradient is calculated as follows:

$$\Delta_x z = \eta \frac{\partial y}{\partial x} \frac{\partial z}{\partial y} \tag{2.6}$$

where η is hyperparameter known as step size or learning rate. The neural network needs gradient for optimization algorithm (e.g., SGD, ADAM [38]) to update weights and train the model.

Another important topic to consider when training neural networks is the regularization term, which aims to reduce the generalization error [39] of the model. Since we aim at both low bias and variance in our model, we need to apply regularization to find a close to optimal equilibrium, parameters that can result in that balanced state between the two.

A simple regularization can be achieved by adding the regularization term $\Omega(\theta)$ to our objective function:

$$\tilde{J}(\theta|X, y) = J(\theta|X, y) + \lambda\Omega(\theta) \tag{2.7}$$

where α is hyperparameter which controls regularization amount. L2 regularization (Weight Decay/Ridge Regression) ($\Omega(\theta) = \frac{1}{2}\|W\|_2^2$) and L1 regularization ($\Omega(\theta) = \|W\|_1$) are common choices for Ω.

Another way of regularization when training neural networks is achieved via ensemble methods [40]. We can train multiple models on the same data, or a slightly varied subset of the data, and at the test time, we can calculate the arithmetic average

of the result from these models. Furthermore, a popular technique based on the idea of ensemble models, the Dropout [41], drops the connection between nodes with given probability at training time, which causes the network to train different sub-networks at each training iteration. Then a geometric average applied to sub-network results to update weights. It is a widely popular algorithm because it can derive good empirical results, and the implementation is trivial.

Finally, early stopping is also popular as a regularization technique, which is based on monitoring accuracy on the validation set while training. The network can be stopped when validation error starts to rise, and so we can prevent overfitting. [34] Validation set data can be later included in the training set by either stopping in a similar error rate or training with the same number of steps.

Convolutional Neural Network (CNN) is a type of feedforward deep neural network that applies convolution operation to given data in grid form [42]. Convolution operation can be seen as weighted averaging on continuous data, and can be denoted as:

$$s(a) = \int x(b)w(a-b)db = (x \star w)(a) \tag{2.8}$$

where w is the kernel function.

Convolution operation itself is widely used for filtering in the image processing domain. Kernel function shifts over the image and applies a weighted sum and generates results. In the CNN, each convolution layer has kernel functions, which are learned from the data:

$$S(i,j) = (K \star I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{2.9}$$

Discussions on CNN networks are focused on the structure of the mammalian brain, which also generates impulses when analyzing images. Convolution operation

gives similar responses, such as low-level features, lines, and edges at lower layers, and detects high-level features, shapes, and edges, at higher layers generated by the network. Neural Style Transfer [43] algorithm exploits this structure by building on the low-level filter responses at the lower layers of a pre-trained neural network.

A typical CNN building block uses three different operations. Firstly a convolution operation is applied, then the activation function is used. Finally, the pooling function is applied. Pooling can help the network to get a robust representation of data, since "most active" or "average" responses are kept, which helps with the generalization. It also gives further nonlinearity after the activation function. Modern neural network structures utilize Batch Normalization (BN) [44], which eases the training of deep neural networks. Instance Normalization (IN) [45] is another normalization variant that is efficient for the image generation tasks.

Widely used CNN variants, such as ResNet [30] and Inception [31] networks has multiple convolutional building blocks. Between each block, the pooling operation is used to reduce the spatial size. The final block ended with a global average pooling layer and followed by an FC layer providing an output structure specialized to classification or regression tasks. ResNet [30] includes the residual skip connections with an identity function between each block. Inception network [31] has a special block structure starting with bottleneck 1×1 convolutions followed up with convolutions that have different receptive fields. Inception networks also have auxiliary classifiers in the intermediate layers resulting in early-stage feature classification.

2.2. Video Recognition

Real-time video recognition is becoming more and more feasible each day with the increasing processing power of modern specialized computer hardware. This generated an increasing trend of video recognition tasks and methodology. The success of neural networks also encouraged researchers to apply CNN based methods to the problem. We will investigate video recognition datasets and methods, used for the human action

recognition task. We omit to share the SLR methods here and investigate it in the next section with more detail.

Firstly, we will briefly describe the video recognition datasets. We choose the human action recognition datasets because they offer classification benchmark and are also used as an extra training data for the Sign Language Recognition. Human action recognition datasets consist of different human to human, or human to object interactions shared in multiple categories. The following list will briefly describe video recognition and human action recognition dataset.

- UFC 101 [14] dataset has videos classified in one of 101 action classes. Dataset set is collected in the wild, which means background or non-human elements are not strictly controlled. Datasets consist of 27 hours of video content and over 10 thousand video clips.
- HMDB 51 [15] dataset includes videos classified in 51 action categories. Dataset is collected through the web, so videos are in the wild setting. In the dataset, more than 7000 different videos are provided.
- Sports 1M [46] dataset includes videos classified in 487 video classes. Videos follow the in the wild setting, collected throughout the internet, and manually labeled. In the dataset, there are more than one million different video clips.
- Kinetics [47] dataset includes videos classified in 400 action categories. Dataset is collected from the videos shared through the internet. In the dataset, each category has more than 400 action categories, which makes a total of more than 160,000 videos.

While datasets grow in size and complexity over time, the methods are also changed to capture more fine details to capture each specific features in different classes.

Secondly, we will investigate the conventional video recognition methods which consists a baseline to many SLR methods. We can divide the standard video recognition methods as handcrafted and neural network-based. Handcrafted methods are known

to be more expensive in the calculation, yet could acquire the coarse domain-specific features. Neural network-based methods are more data-hungry and can perform well when the dataset is bigger [48], yet they have increased training times.

For video analysis, optical flow is one of the common themes in the community. Optical flow aims to estimate the dynamic flow map between two given frames in a given video clip. [49] Sparse optical flow uses points of interest, such as edges or corners, then calculates the difference between frames on such points. Dense optical flow algorithms extend the number of tracked points, with a penalty of more processing power requirements. Optical flow is a handcrafted algorithm that consists of a base for more advanced methods. [50–52]

Going forward from the frame by frame comparisons to multi-frame comparisons, handcrafted algorithms have followed the direction of the feature trajectories. They were using a more established feature extractor, eg. SIFT [53] can be kept throughout the temporal stage using a tracker algorithm. Frequently the tracked features can not capture the domain-specific ROIs and points. Wang et al. propose an extension Improved Dense Trajectories [13], which follows the dense sampling similar to the dense optical flow variants but handles the irregularity caused by outliers by discarding them. It performs a well-known motion specialized feature extractor in the video data.

A well known neural network approach to classify videos is applied by Simonyan et al. [54]. Two Stream Convolutional Networks use a branching structure that differs between spatial and temporal regions of video input data. The spatial approach uses a standard CNN; the temporal stage is where the method differs from previous counterparts. It uses a pre-processed optical flow data to memorize the transitions, the temporal information, between frames. Training both networks at once provides an ability to generate features that depend both on spatial and temporal sub-components. Achieving such feature extraction eases the training and generates good results on several empirical benchmarks.

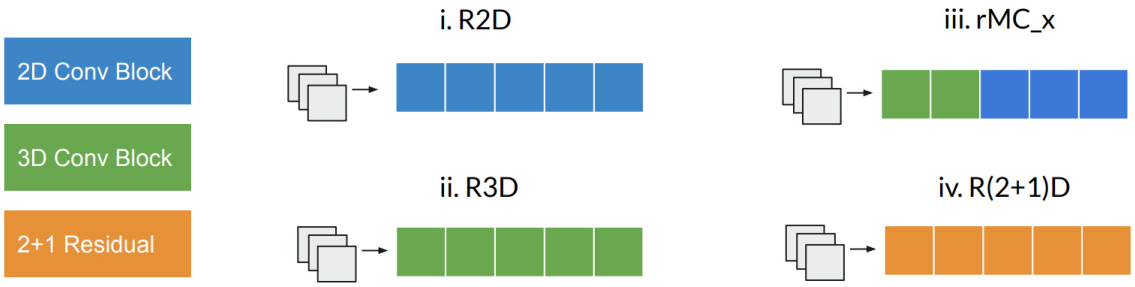

Figure 2.1. Residual architectures proposed by Tran et al. [1] visualised. While R2D and R3D are using a single block type, others use a combination of different scale convolution operations.

To get a fully automatized neural network approach, one needs to find a feasible way to stream temporal information directly through the neural network. 3D convolution operation can handle the extra temporal dimension when feeding input video clips directly into the network [55]. Such an approach is power-hungry and differs from the classical neural network literature in terms of training approaches [56–58]. Carreira and Zisserman [47] propose a three dimensional convolutional modification to the inception architecture $I3D$, which outperforms Two Stream 2D CNN and CNN-RNN hybrid variants. Due to the high complexity of the Inception network, it is hard to scale the network for different applications.

Tran et al. [1] share a simple 3D CNN based on the residual network. With reliable empirical results, Tran et al. also investigate different network structures to process the temporal dimension in the CNNs, as shown in Figure 2.1. All networks use standard residual-based CNN architectures, 1) in a full 3D convolution setting, 2) a 2D and 3D mixed setting, and 3) residual bottleneck 2+1 setting. These networks provide novel ways to process the results. While the R(2+1)D bottleneck setting is efficient and most suitable for the big datasets, the mixed convolutional variant has trivial implementation and generate successful results with the fewer data.

Advanced neural architectures also proposed for the task are still investigated. Tran et al. [59] propose 3D group convolutions in the Channel-Separated CNN variant.

Dividing the spatiotemporal and the channel interactions are shown to lead to more accurate results. Further improvement is achieved via weakly supervised pre-training as reported by Ghadiyaram et al. [60]. To improve upon the training time, Wu et al. [61] tackle the problem by proposing a grid schedule that can work out of the box with video models. Feichtenhofer et al. [62] use the attention paradigm and design a divided network that operates on both slow and fast frame rates.

Along with video recognition methods, pose or depth estimation frameworks are commonly used to estimate dense information locations. Using the pose or depth data from an external source can improve the video recognition performance [63]. We classify common frameworks in two categories: 2D pose based and depth based. The first approach is using is a pose estimation framework, eg. OpenPose [64] can apply 2D real-time multi-person keypoint detection by processing the image via a pre-trained network. It can generate 25 keypoints for the body region, and 2 keypoints for the hand region as an output. The second approach is using a 3D depth sensor, such as Microsoft Kinect [65].

Both handcrafted and neural network-based methods work together, achieving the state of the art result on the most challenging action recognition datasets. The specific problem, sign language recognition benefits from the gains of such methods, and we will describe the advancements on the topic in the next section.

2.3. Sign Language Recognition

Automatically estimating human movements using a computer is a big opportunity that can ease the day to day life. Sign language recognition approaches aim to ease the life of the deaf and is a well-researched topic since the late 1990s [7, 18, 20]. There have been approaches to the problem involving the extraction of knowledge from well-structured gestures and gesture patterns, which involves designing a rule-based system for capturing and matching the given pattern. [10] Another approach is using the external non-manual modalities such as sound and mouthing patterns [19, 21]. A robust

classifier combines both information to achieve the best result [12, 23]. Thus the data-heavy algorithms such as neural networks should consider feeding all of the available information, sometimes emphasizing the specialized units [63, 66] in order to exploit the attentive property. In this chapter, we will discuss how sign language research has progressed in terms of available data and methods.

2.3.1. Sign Language Recognition Datasets

Sign language recognition datasets release with clear distinction on different sign types, usually with attached sign label, the sign gloss. Sign glosses are the annotations that are the closest match of operated hand sign to translates the spoken language. Using this gloss information as labels, we are one step closer to capturing the performed sign and closing the gap between the spoken language and performed sign language.

Sign language recognition datasets can be classified into two categories: first is the word level or isolated datasets, which consists of a single sign and its word gloss counterpart; on the other hand, continuous datasets range between few word phrases to long dialogues. Table 2.1 shows the common SLR datasets and their comparisons.

Table 2.1. Sign Language Recognition Datasets and their respective properties. Asterisk ($*$) mark is used to denote that signers are native. Pose column represents if authors have provided the corresponding depth and pose data.

Dataset	Year	SL	Glosses	Samples	Signers	Task	Pose
RWTH-Phoenix [67]	2012	German	1231	6,500	7	CSLR	No
Devisign [68]	2015	Chinese	2000	24,000	8	ISLR	Yes
CSL [4]	2018	Chinese	500	1,25m	50*	ISLR	Yes
MS-ASL [3]	2018	American	1000	25,000	200*	ISLR	No
WL-ASL [2]	2019	American	2000	34,000	100	ISLR	No
BophorusSign22k [5]	2020	Turkish	744	22,000	6*	ISLR	Yes

Sign language recognition problem application scenarios involve solving complex phrases and understanding performed language thoroughly. While isolated datasets help us approach the problem via the divide and conquer approach, researchers pursue more in the wild setting, the more complicated phrase level or sentence level continuous datasets. For example, DGS Corpus Project [6] dataset includes phrases from a wide range of real-world situations and RWTH-PHOENIX [67] dataset includes videos collected from commercial weather forecast presentations. Recognition of a continuous dataset presents a different challenge than an isolated counterpart and can further boost SLR development. In the next section, we will investigate the conventional methods for sign language recognition.

2.3.2. Sign Language Recognition Methods

The Sign Language Recognition task follows closer to action recognition in terms of methodology due to the similarities between the two tasks. We can group similarities three-fold. The first is the dataset organization: Both tasks include learning from video data, which means the raw data included in machine learning systems are in similar shape. Secondly, both aim to extract the movement, which requires capturing dynamic patterns throughout the video requiring similar methods. Finally, while action recognition requires processing and guessing labels, sign language recognition has sign glosses that can provide similar guidance and challenge. With similarities, sign language literature benefits much from the action recognition methods, and we will discuss them briefly.

Older sign language recognition research uses physical sensor data such as the wearable gloves and carefully designed a rule-based system or a simple ML-based system [7, 10]. Liang et al. use glove data along with a 3D point tracker to generate posture distributions and use Hidden Markov Model (HMM) [69] classifier. Kim et al. [20] use the glove feature as an input to fuzzy min-max neural network classifier for the Korean SL recognition. Vogler and Metaxas [19] use 3D tracker data and apply parallel HMMs (PaHMMs) for the 22 gloss German SL classification. Fang et al. [8] use

an improved simple recurrent network (SRN) to extract isolated regions from sign sensor data, and then apply classification with machine learning algorithms. Hernandez et al. [16] use the linear discriminant analysis (LDA) algorithm and Bayes' classifier to predict the ASL alphabet, along with stop and continue special characters. Vogler et al. [21] use hand motion capture data and extract features using the rule-based system, then apply the ML-based classification.

In order to improve the sensor-based processing, researchers adopted different computer vision and achieved competitive empirical results on the SLR task [10]. Holden and Owens [12] build a Hand Motion Understanding (HMU) system based on a combination of 3D hand tracking data and 2D image sequence data with a fuzzy expert system classifier. Farhadi et al. [9] use an avatar-based transfer learning approach to achieve signer independent classification. Two module SLR process shared by Zhang et al. [17] firstly detects hands and subtracts the background, then applies the HMM training after the PCA dimensionality reduction. Liwicki et al. [11] build a British SLR system that starts with the hand segmentation, proceeds with the generating HOG descriptors, and applies the HMM classifier at the end. Lang et al. [23] apply 3D depth estimation then follow with 3D reconstructed avatars, which are used as a feature base for the HMM classification.

We will discuss the recently proposed SLR methods that achieve state-of-the-art results on the recognition benchmarks. For the isolated sign classification task, Zhang et al. [4] apply a novel trajectory-based method on the mapped hand regions in 3D Kinect data. To refine features, they follow up with the PCA and then generate HOG descriptions before feeding it into the HMM model. Ozdemir et al. [5] use dense trajectories [13] as a feature trajectory method, then build a SVM based classifier on top of the features. Deep neural networks also have been used for the SLR task. [5] uses end to end approach with 3D residual networks, [3] firstly experiments with 2D CNN and RNN two-stage structure, then adopts a 3D variant of the Inception network, I3D. They also adopt skeleton-based input data and successfully tackle sign classification and signer identification problems. For the Indian SL recognition, Hore et al. [24] use

the Genetic Algorithm (GA) based Particle Swarm Optimization (PSO) along with the neural networks.

For the continuous sign classification task, Koller et al. [67] apply a sub-unit based method by tracking high-frequency sub-units. They tackle the sign search task by training a neural network, which combines the visual model and language model. Camgoz et al. [70] aim to solve simultaneous alignment and recognition problem in the sign language domain via including an attention-based mechanism. For a given input sequence, spatial modeling is achieved via CNNs, and then the bidirectional LSTM model is used for modeling the temporal domain. In [66], the sign language translation task is tackled by matching each gloss with the corresponding visual clue, aims to shrink the gap between spoken language and performed sign language even further. The pre-trained expert language model is fine-tuned with an end-to-end setting to achieve competitive results. The following work [71] introduces a widely used transformer model to the SLR task. The training gloss prediction objective is improved with the usage of the connectionist temporal classification (CTC) loss. Tackling the translation problem is one of the more laborious tasks that are still worked on and have continuous progress.

Each work exploiting different aspects of sign language improves upon already known SLR benchmarks and further gives the opportunity to create more and different tasks imitating the real-world problems such as SLR translation [66, 71]. Despite the popularity of the end-to-end architectures, they fall short on specialization on each sign in terms of multiple cues, which places a room for improvement in those scenarios. In the following chapters, we will describe the methods for extracting cues in detailed fashion and propose our multi cue fusion method.

3. SPATIAL SAMPLING

Sign Language Recognition focuses on body pose, hand shape, and face gestures. We build our spatial sampling with these three sources of cues. Figure 3.1 shows the activity of the hand cue region.

3.1. Body Model

We followed the body region sampling provided in the baseline paper setting. Detailed description is provided in Section 6.1.



Figure 3.1. Hand cue movement on a sample video in Bosphorus Sign 22k dataset is visualized using the optical flow algorithm.

3.2. Hand Model

Hand movement is a crucial information medium in SL communication. We aim to use the hand region for the sampling process. However, there is an ambiguity of which hand needs to be selected during the sampling process.

3.2.1. Hand Movement Detection

SLR literature suggests the primary mean of communication is conveyed thought the most active hand, the dominant hand. We proceed with detecting the most active hand. We have applied the following steps for detecting the hand keypoints, as shown in Figure 3.2.

(i) Detect the Thumb keypoint using the Openpose pose estimation framework [64]. The thumb keypoint corresponds to the 9th keypoint in Hand Keypoint set.
(ii) Declare the Thumb keypoint in the first frame as an anchor, denoted as A.
(iii) Define a hand movement threshold in terms of image pixels as T_S.
(iv) Calculate the distance of the between the anchor and the thumb keypoint (KP_n) on the following frames. If the distance is more than the threshold, declare that hand is moving.

We have employed the Euclidian Distance as our distance metric and defined the $T_s = 150$. Our threshold calculations are done on the 1920×1080 image before any resize operation. For each sign video, we have applied this process for each hand, signer's left hand, and signer's right hand.

To show the hand detection process more closely, we share one moving hand example in Figure 3.3. This process is applied to the signer's left hand. Using this frame, we declare that the left hand is moving using the information that the distance between keypoint and the anchor is bigger than the threshold.

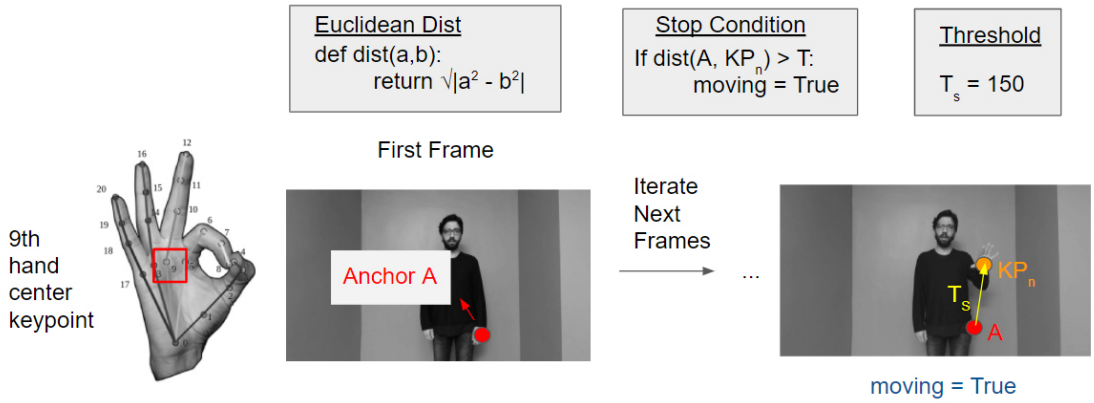

Figure 3.2. Hand movement detection process is visualized. We use the 9th hand center keypoint as an anchor in the first frame. Then we start the iterate the following frames. We calculate the current frame keypoint distance compared to the anchor. If the distance is bigger than threshold, we declare the hand as moving.

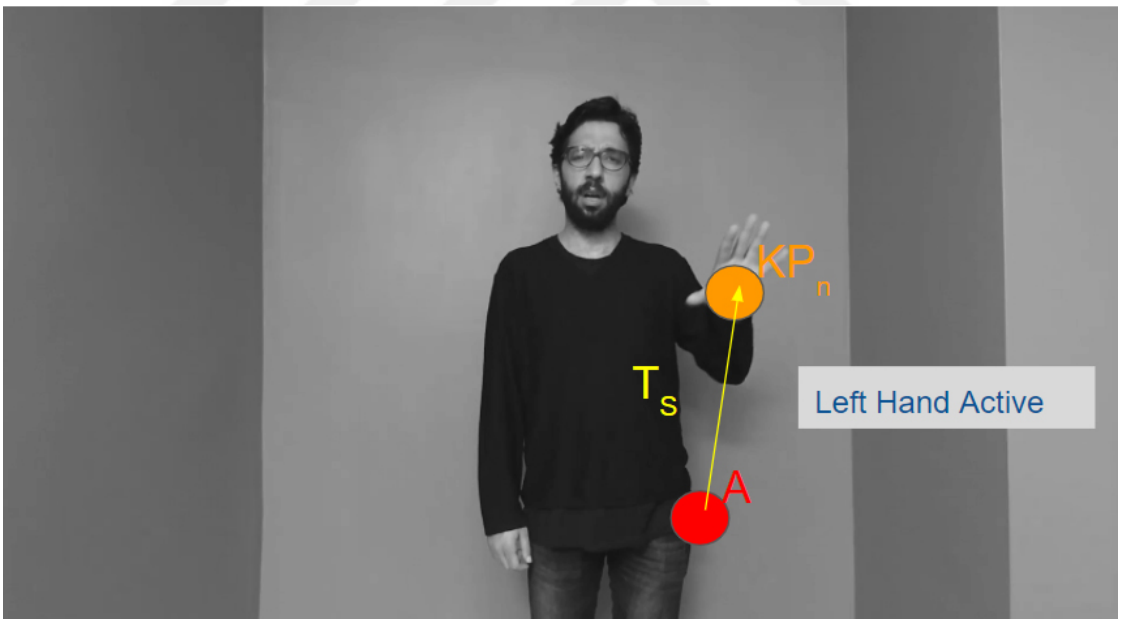

Figure 3.3. Hand movement detection applied on a frame where left hand movement occurs. We compare the anchor position and current hand keypoint position. Since the distance is bigger than the threshold, we detect the left hand as moving.

We also show one negative case where the right hand is declared as not moving in Figure 3.4. We can see that, compared to the anchor frame, hand movement is



Figure 3.4. Hand movement detection applied on a frame where right hand does not show any movement. We calculate current keypoint position and compare with the anchor. Since there is barely any movement, and the distance is smaller than the threshold, we conclude that the right hand is not moving.

almost non-existent. Since we compare anchor with the new keypoint, the distance is not greater than the threshold, and we declare the right hand as not moving.

3.2.2. Hand Cue Settings

We proceed with modeling the hand cue in three settings, single hand, both hands, and mixed. For single hand setting, we selected the dominant hand and applied cropping to 350×350 around the center of second hand keypoint. In both hands setting, we tried to use both cues by using both hand crops. Cropping is applied on the Thumb keypoint with an area of 175×350; then, horizontal concatenation is applied.

Our final mixed setting aims to use all of the cues of previous settings at once. In the mixed setting, we adopt a rule-based sampling dependent on how many hands are active at the given video. The first rule is, if the signer only uses one hand, crop only

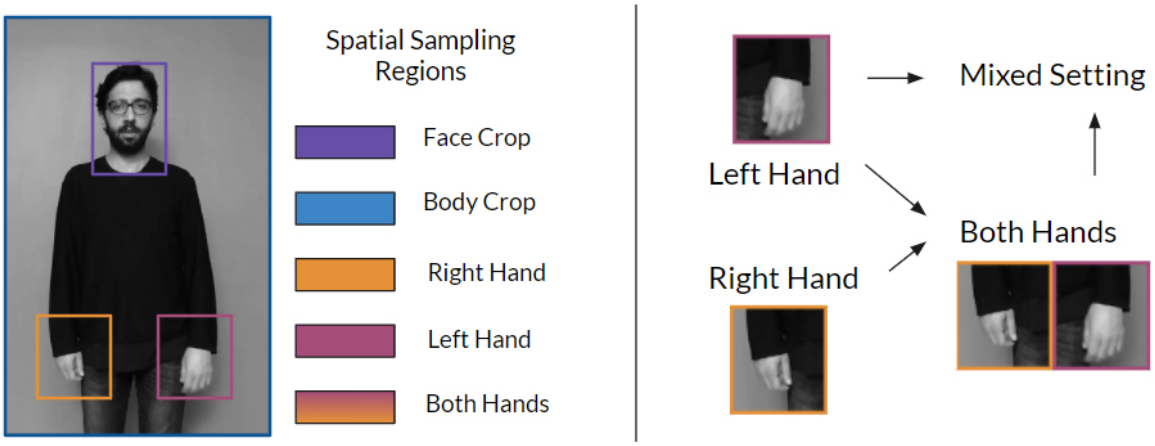

Figure 3.5. Spatial Sampling operation is visualized. From left to right; cue regions selected for the process, and hand crop settings.

the active hand as in a single setting. The second rule is, if the signer uses his/her both hands, apply sampling to each hand and concatenate horizontally, as in both hands setting. Using the two rules, we process all of the datasets to generate mixed setting results and train the network using this mixed data.

After all of the sampling operations, we follow up with an image downsampling to resize each frame to 112×112 resolution, which is the input scale of the MC_x network. The downsampling operation is applied using the bilinear interpolation. Test results of each hand setting are provided in Table 6.1.

Table 3.1. Hand activity distribution in the BS22k dataset detected through movement estimation algorithm.

Distribution	Relative Frequency (%)
Both Active	**66.44**
Only Left Active	33.07
Only Right Active	0.40

For the single hand setting, we have also tried to use the non-dominant hand, and for the mixed setting, we have also experimented with adding a black padding alignment to single parts. Both attempts resulted in severe accuracy loss.

3.3. Face Model

Sign Language Recognition systems also focus on the cues in face regions, such as lip shape and head alignment. We decided to use the face region as a whole instead of separating each cue since the face region is relatively small, and cues are connected. Our face cue sampling is applied by firstly extracting the nose keypoint in Openpose [64] Body 25 setting, then cropping in the 200×200 region.

We have also experimented with extending the best performing hand setting with the face crop. Face crops are horizontally concatenated to each hand setting. Horizontal crop size is defined as $350/N$ pixels, where N is the number of subunits in the image. Vertical crop size has remained at 350 pixels. The final image is resized to 114×114 as in the previous setting. This setting has resulted in 76.99% percent accuracy on the test set, with a 10.73% related performance drop compared to the hand counterpart. We suspect that the performance drop occurs due to decreasing spatial size with each cue included in a single image, which causes a race condition between each cue.

4. TEMPORAL SAMPLING

Frame sampling of SLR videos is an essential preprocessing step and needs to apply before feeding into the CNN. Standard Temporal Sampling practice involves choosing frames with equal distances in a uniform fashion. We show that previous work using cues on the temporal domain by either employing another branch with pose data [54], or different frame rate and resolution [62]. Instead, our Active Window based Temporal Sampling pipeline selects the temporal region in which the hand movement occurs. We then follow up with the uniform sampling process in the given window before feeding the data into the network.

The temporal sampling operation is visualized in Figure 4.1. The top row shows the standard uniform sampling process, and the bottom row shows our active window based sampling process.

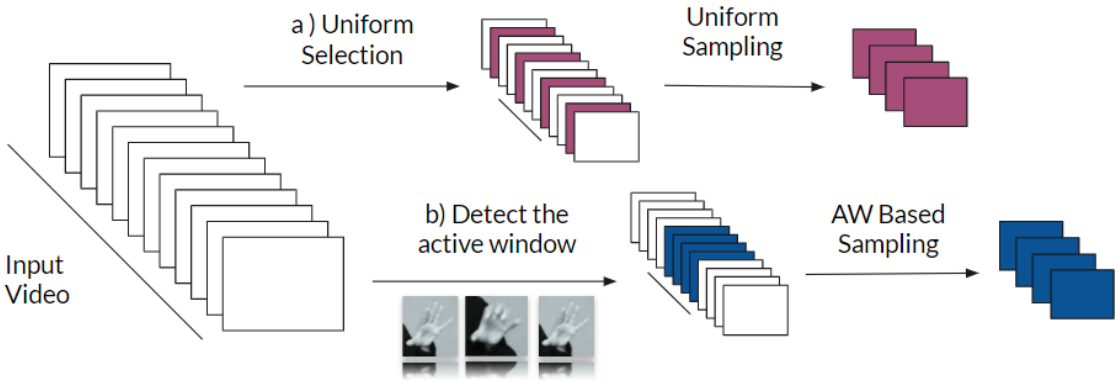

Figure 4.1. Different temporal sampling operations are shown in the above figure. Selected frames are shown with color. Two branches represent uniform sampling and the Active Window Based Sampling Process.



Figure 4.2. Active window detection algorithm is visualized. We start by using first frame anchor and iterate over following frames and detect the frame where the first movement occurs. We apply the process in reverse by starting with the last frame and with reverse iteration. Finally we declare the part between the first move and inverse move as the active window.

We need to determine the moving hand parts to detect the Active Window Temporal Region. The following steps summarize the algorithm for segmenting the active window, as also seen in Figure 4.2.

(i) Use the moving hand detection framework in the Section 3 to detect the active hand(s).

(ii) Define a selected hand as the active hand. If both hands are active, select the dominant hand.

(iii) For the selected hand, track hand movements using the Euclidean distance. Save frame numbers as an two anchors. First one, the start of the first-hand movement saved as the A_S. Second one, end of the last hand-movement saved as the A_E.

(iv) Define start and end thresholds T_S and T_E. Filter the boundary regions from the start and end frame ids using corresponding anchors defined earlier, and use extracted frames for the training.

We have selected the two threshold values, $T_S = 90$, and $T_E = 50$, starting and the ending thresholds, respectively.

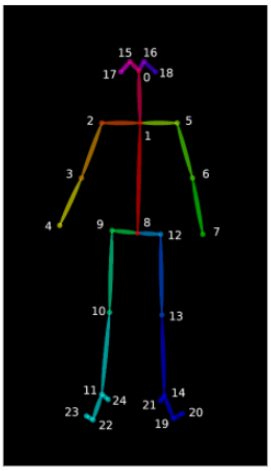

Figure 4.3. Exceptions in the active window detection process. The first exception is hand already starting in the signing position. We detect such occurrences by checking the distance between hand(4) and hip(8) keypoints. The second exception occurs when there is poor hand detection or irregular hand movements. We discard the sampling in the following occurrences.

Furthermore, our experiment shows that there are edge cases that need to be considered when detecting the active window. Exceptions are shown in the Figure 4.3, and described as following:

- **Hand starts/ends in sign position:** This setting leads to false trimming since our algorithm detects the movement only in the middle of the sign performance. We have eliminated this setting by introducing a filter that checks if the hand starts away from the range of 150 pixels to the Mid Hip (8th keypoint in Body 25 setting) region. If so, it is flagged as the hand starts or ends as moving. Also, the corresponding movement is tagged as included in the segmented part.
- **False positives:** We have found that when false positives happen, the resulting segment can be less than 10 frames. In such cases, we have used the whole video rather than the segmented part.

5. MULTI CUE FUSION

We have described the spatial and temporal sampling processes in previous sections. After collecting the multiple expert cue models, there is a need to combine the more specialized weaker classifiers into one final classifier. For this process, we have investigated multiple fusion processes.

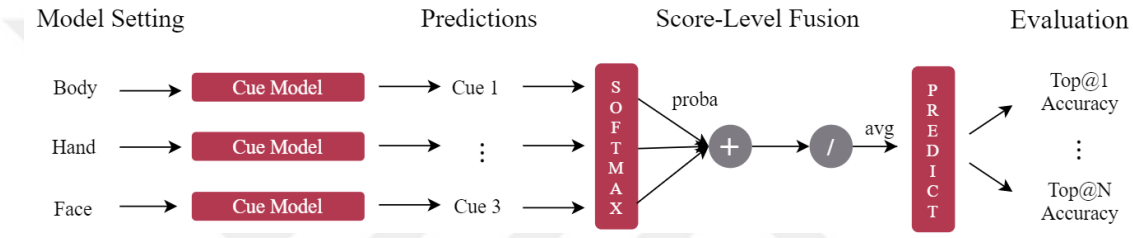

Figure 5.1. Score-level multi cue fusion operation applied at the test time. Note that cue networks have different test weights even the architecture is same.

5.1. Score Level Fusion

5.1.1. Standard Score Fusion

We firstly experimented with the standard score level fusion similar to the one by the Simonyan et al. [54]. After getting the final layer (FC) from the multiple networks, we get the scores applying the Softmax function to the results.

Score-Level fusion operation is performed by averaging the final score data, then making a prediction using the fused score. Figure 5.1 shows our score level fusion operation. Firstly, we extract different cue features from different models. Then we follow up with an individual softmax operation to the extracted features. Finally, we average scores and generate prediction results by taking the top gloss score.

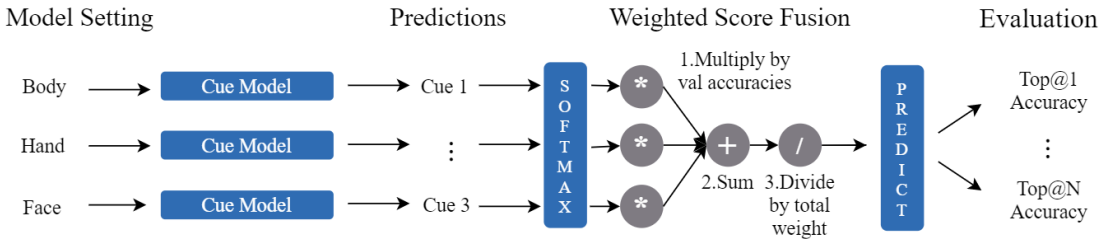

Figure 5.2. Weighted score fusion approach is visualized. After getting the each cue softmax activations, we firstly multiply them with the each model's validation accuracy. Secondly, we calculate the sum of scores, and finally we divide the scores by the total weight.

We have also tried applying the averaging before the softmax operation. This setting was identical with the other, except softmax and averaging operations are exchanging their order in the pipeline. We have found that this setting results in %2 to %3 accuracy loss. We suspect that softmax operation further boosts prediction confidence of the single cue models, and if operated later, it results in weaker overall estimation.

5.1.2. Weighted Score Fusion

In this setting, we start on the post softmax score fusion approach, and instead of averaging, we apply a weighted sum. This weighted approach allows each cue model voting with respect to it's expertise on the given cue.

Our weighted score fusion method is shown in the Figure 5.2. We firstly extract each cue model features, then follow with the softmax operation. Then we multiply each cue model score with it's previously calculated validation set weight separately. We follow up with a sum operation to reduce the score set into one. The final operation is the division by the total amount of weights, which is the sum of the previous validation accuracies. After getting the normalized weighted sum, we proceed with the prediction and score calculation.

5.1.3. Failed Settings

Fusion via Second Stage Classifier. We have also investigated applying score fusion via another second stage classifier. This classifier has taken network features for the input. We have used two sets of features, outputs of the final Fully Connected (FC) layer, and the previous Average Pooling (AvgPool) layer. For this setting, we have investigated three different classifiers.

(i) Multilayer Perceptron. The universal approximation theorem concludes that an MLP with arbitrary width can approximate the given function [33]. We have used one hidden layer with an additional Dropout layer with $p_D = 50$. We have applied a grid search to hidden layer size in the range $(500, 750...2000)$.

(ii) SVM. We have adopted the Support Vector Machine (SVM) classifier with linear and logarithmic kernels. The grid search has applied to the C value ranging in the range $(2^0, 2^1...2^6)$.

(iii) XGBoost. Extreme gradient boosting is an efficient iterative boosting algorithm built on the weaker binary tree stub classifiers [72]. It has the advantage of being efficient and scalable over the alternatives. Due to a huge number of hyperparameters, we have applied a small grid search on the author's suggested settings.

Despite efforts on selecting different algorithms and a different set of features, using a second stage classifier resulted in accuracy loss compared to final results settings. Our best classifier setting, the SVM classifier using the FC features, did not even perform better than the best single cue setting, which concludes that the experiment has failed in the current state.

Our one key failure reason stems from our limited hyperparameter search budget. Another failure reason is different cue models having a huge accuracy gap and face cue model having very low accuracy rate, which generates a cold-start situation when fed into the second classifier. In future work, we hope this method will be investigated thoroughly.

Self Score Fusion. We have also investigated an approach similar to the teacher-student networks and knowledge distillation [73] methodology.

Our approach stems from the evidence of the models with different parameter spaces, even trained on the same data, can generate different decision spaces, and improve final accuracy with some weighted voting schema [74]. During training, we have noted the validation accuracy scores in each epoch and saved the given parameter set if the current epoch results in a better score compared to the previous best accuracy. Using this, with a single training budget, we have built multiple classifiers that stem from the training stage of the expert network itself.

Our self score fusion approach builds on the assumption that training time itself can be a source of the self-supervision, and network can converge different local minimums during training. While the latter is an assumption with the evidence [33], we do not have any proof for the former argument. The closest hint is early stopping being a regularization technique, and thus each best local best performing checkpoint has different regularization during training and representing a different function as a result. However, recent work by Xie et al. [75] provides a direction for such research by using the training time as another parameter tuning dimension to work on and denoting the previous checkpoints as another weak classifier.

We have built an experiment setting by collecting the best checkpoints where the accuracy improves compared to the best previous accuracy. For the close examination, we describe our algorithm on the sample training scenario, as shown in Figure 5.3. We simulate a CNN training and share the Validation Set accuracies through Epoch 10 to Epoch 25. This training results in the best validation accuracy of %42 at the Epoch 20, and we apply early stopping after no improvement in Epoch 25. During the training, we have collected the model parameters where the current epoch accuracy is greater than previous epoch accuracies. For the given scenario, Epoch 11, 13, 18, and 24 parameters are saved as different four models. After collecting different models, we proceed to build our experiments.

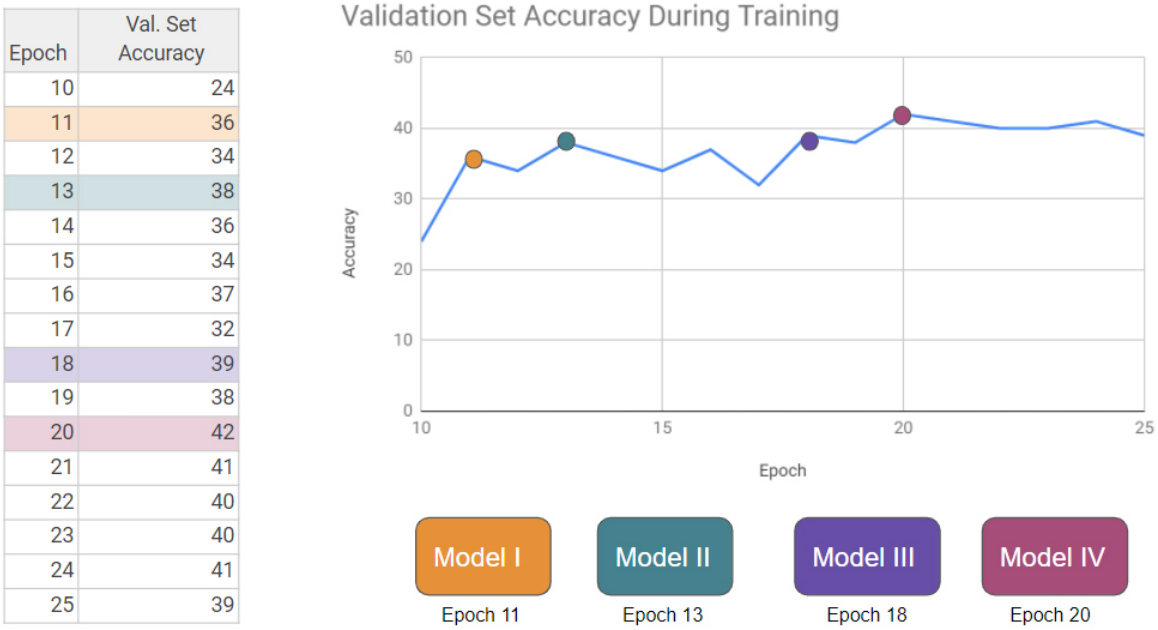

Figure 5.3. Model selection for the self-fusion is explained. For the sample CNN training, we calculate the validation accuracy after each training epoch (leftmost table) and provide the corresponding validation graph. We select the checkpoints that increase the best previous accuracy and save their model parameters. Finally, we extract these saved model parameters at corresponding epochs as different models as seen in the bottom part of the image.

For the different self-fusion settings, we have investigated two different hyper-parameters N_M and n_{sub}. First is the number of models included in the fusion pool denoted as N_M. Models included in the fusion pool are selected among the best performing models, so the selection is applied reversely in the training time. Following the above training example, $N_M = 1$ corresponds to using only checkpoint $< 24 >$, $N_M = 2$ denotes the fusion pool of the checkpoints $< 24, 18 >$, $N_M = 3$ denotes fusion pool of checkpoints $< 24, 18, 13 >$ and so on. The second hyperparameter is n_{sub}, which is the subset length that we apply fusion. We select all the subsets with a length of n_{sub} from the pool, apply the fusion, and select the best performing subset using the validation set performance.

In our work, we first tried the self fusion approach on a single cue setting where $N_M = 10$. We have changed the n_{sub} value as an adjustable hyperparameter in range $(1, 10)$. Experiment results are shared in the table 5.1. We can see a smaller n_{sub} values result in an increase in accuracy until the range of 4 to 5, where the peak fusion performance results with %4.42 increase in accuracy over the base model. Standard and weighted fusion variants have similar performance in our self-fusion setting.

Table 5.1. Self Fusion Results on the single Body cue setting. Models' identities are ranging from 0 to 9, which represents the fusion pool, sorted by their single validation set performance in the descending order. Different subset lengths (n_{sub}) are evaluated and found the best performing subsets on the validation set. Results show the score fusion and weighted score fusion accuracy results of the given subsets.

n_{sub}	Best Subset	Fusion Acc.	Weighted Fusion Acc.
1	(0)	75.73	75.73
2	$(0, 1)$	78.91	78.89
3	$(0, 1, 6)$	79.75	79.73
4	$(0, 1, 2, 8)$	**80.15**	80.13
5	$(0, 1, 2, 3, 9)$	80.13	**80.15**
6	$(0, 1, 2, 3, 4, 9)$	80.00	80.06
7	$(0, 1, 2, 3, 6, 8, 9)$	79.84	79.91
8	$(0, 1, 2, 3, 4, 5, 8, 9)$	79.64	79.73
9	$(0, 1, 2, 3, 4, 5, 6, 8, 9)$	79.55	79.55
10	$(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)$	79.13	79.24

We tried extending such a method into the multiple cue setting and achieved inferior results compared to other score-level fusion settings. We omit the sharing score details and instead investigate failure reasons for the approach to generate a baseline for the possible future work. The following are the failures of the Self-Fusion approach:

- Worse performance compared to other guided multi cue score fusion methods. Despite the wide number of available checkpoints, self-fusion method accuracy fluctuates and saturates with the increasing model count.
- Random behavior at the model selection. As seen in the single cue results, even though the best checkpoint models are heavily used, best combinations still include arbitrary checkpoints, e.g., the best $n_{sub} = 4$ setting also having model #8. This behavior also needs further investigation.
- Search Cost. The fusion operation requires evaluation of all of the subsets in search of the best validation set performance. This algorithm has a terrible $O(2^n)$ growth rate, which makes it not feasible for bigger feature pools, and better search methods should be investigated. Noted that the search cost does not involve CNN inference time since validation features can be precalculated.
- Cross-Validation Requirement. Since our algorithm requires evaluation of the given subset search space, it will be beneficial to implement a better validation mechanism, similar to the cross-validation.

Even with the inferior results with multi cue setting and self fusion is not being applicable in our experiments, it still has the following benefits with the applications that only have a single cue. The following benefits and research directions need to be investigated in future work.

- Training/Test time trade-off. Self-fusion is an option to boost the final accuracy result without an extra training time, with a trade-off of a testing time costs, with the reasonable $O(n)$ growth rate. For an application that requires scheduled training with close intervals, self fusion can use the limited budget on the test time while achieving better results.
- Pre-calculation. For the subset search operation, memoization could be a big advantage in terms of processing power. Due to the iterative nature of the training task, a subset search can be performed during the training process by saving the extracted validation features and processing search operation on the GPU memory.

- Distributed Computation. Since self-fusion operation builds on training a single model, multiple models can be trained on different devices to contribute to the fusion pool. This model search can also be more efficient if networks start at a pre-trained state, or employ regularization during training with noisy inputs between the actual data batch as in [75], or with the noise directly applied on the parameter space or in normalization parameters [76]. This LazySearch schema can collectively generate the fusion pool. While generating the fusion pool, another cluster can work for evaluation and finding the best subset.
- Allows working on an already-trained network. While it is not efficient to train a previously optimized checkpoint further for performance, it can be beneficial to destroy its state gradually. This process applies the model search in the reverse state and can be used to generate a fusion pool. The operation can be done with noisy inputs as in [77], or with heavy regularization. This ReverseSearch schema can boost and improve the performance of the already deployed real-world applications.

With the increasing computational power available as described in Moore's Law, and considering the inefficiency of building very deep neural networks [78], there needs to be an efficient way to utilize deep neural networks in the big scale. This utilization can also benefit using distributed and cloud computing [72]. Our proposed Self-Fusion operation is a straightforward way to utilize such processing power focusing on the score fusion and test time performance. We hope future work will investigate the self-fusion operation, provide benchmarks on the widely used datasets, and investigate the advantages of such distributed calculation.

6. EXPERIMENTS AND RESULTS

Signs are expressed using hand shapes, upper body movements, and facial expressions, which constitute cues [10, 63]. Our aim is to extract information from different cues in signs. For this reason, we attend to different regions in the frame and to different temporal windows and train expert CNNs concentrated on different cues. We compare the recognition accuracies of different experts and fuse them. Our experiments are based on filling the gap of spatial and temporal cue sampling in previous SLR methodology.

This section is organized as follows; firstly, we will talk about experiment setting and base results. It will be followed by our sampling results and the score fusion experiments. The third section will describe the combination of both spatial and temporal sampling methods and our best results. Finally, we will include a detailed analysis of the results by showing the role of each cue mechanism in our framework.

6.1. Experiment Setting

To achieve a competitive experiment setting, and to implement our proposal effectively, we have used a recently published Turkish isolated SLR dataset BosphorusSign22k [5]. The dataset contains six different native signers, performing 744 different word categories. Each category is labeled with a sign gloss that describes the performed sign. The dataset contains about 22,000 video clips. Authors share extra metadata, containing performed ids for each video, 3D body pose keypoints in Kinectv2 format, and 2D body and hand keypoints.

Following the authors in [5], we aim to compete on the sign language classification task. It is described as estimating the corresponding sign gloss for a given input video at test time, and scoring is evaluated in the accuracy of all of the test estimations. Out of all 6 performers, video clips of Signer 4 is defined as the test set. It has a size of

1/10 of the total dataset and includes samples from all of the 744 classes.

Ozdemir et al. share the performance of the Bosphorus Sign 22k dataset using two methods. Firstly they resize the videos to 640x640 resolution and use multiple trajectory estimation algorithms to extract features from the input video data. After extracting features on the training set, they train an SVM classifier. Then the classifier is tested on a test set with the same feature extraction pipeline. Best results are achieved via IDT [13] algorithm with 88.53 percent accuracy. Secondly, they use mixed convolutional residual networks [1], a deep convolutional neural network variant. 3D CNN is fine-tuned from the Kinetics dataset pre-trained weights. Input videos are cropped to 112 by 112 resolution before feeding into the network. The test set is evaluated on the trained network and achieved %78.85 accuracy in the best hyperparameter setting.

Our experiment setting follows the author's setting on a deep neural network. To achieve the best replication, we have performed a similar preprocessing pipeline, resized the image into 640 by 360, cropped the central square region, then resized via bilinear interpolation to achieve 112 by 112 input resolution. We have also adopted a similar hyperparameter setting, applied uniform frame sampling, fine-tuned the last three blocks, used the MC_3 variant of the CNN network. With the same setting proposed by the authors, our training is completed within 48 hours on Nvidia 1080 TI GPU hardware. We have used a batch size of 32 video clips and applied the test on the Signer 4 split. Our replicated network resulted in 75.23 accuracy, having a three percent difference from the reported result. We suspect the difference is caused by our preprocessing schema, optimizer initialization, and other possible hyperparameter choices such as the learning rate.

6.2. Cue Sampling and Score-Level Fusion

With a replicated baseline, we now have an opportunity to explore our cue extraction mechanisms. We design a three-stage pipeline consisting of sampling operations

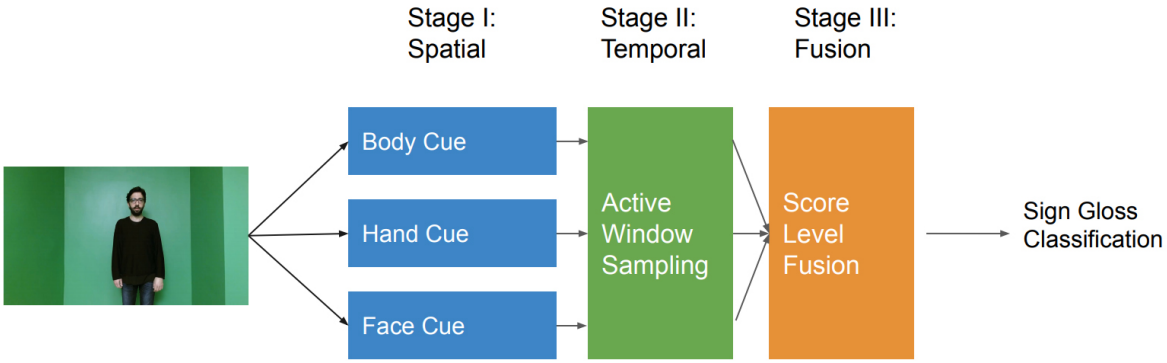

Figure 6.1. Our sampling and fusion pipeline is visualised. We divide our approach into three stages, spatial sampling, temporal sampling and score-level fusion.

and then score fusion, as shown in Figure 6.1.

We firstly share the spatial hand sampling results in Table 6.1. We can see that the mixed setting generates the best accuracy results. We suspect that this benefit occurs because of mixed data allows the network to differentiate between the one-handed and two-handed signs easily. We can see that both hand setting is the second-best setting. Even with both hands not being active in all of the videos, it is still beneficial to provide both hand data at the training phase. After declaring the best hand setup, we proceed with our sampling pipelines.

Table 6.1. Test results of the different hand crop settings and resulting accuracy values.

Hand Cue Setting	Accuracy(%)
Single Hand	79.13
Both Hands	85.81
Mixed	**86.25**

We describe sampling settings by starting with the spatial sampling applied to the cue regions described before, body, hand, and the face. Regions are cropped and then given to the network as input data. Then we proceed with our standard training

pipeline. Spatial sampling results are shared as the left column of Table 6.2. We can see that each spatial sampling setting builds on the expertise of the assigned cue region and generates different results. We can also see that the hand cue model performs better than the body or the face cue.

We follow up by applying temporal sampling on the active window region. We firstly process each video in the Bosphorus Sign 22k dataset to extract the hand movement regions, as described in the previous section. After gathering the active window data, we follow up with a uniform sampling in the active window region then feed the data into the mixed convolutional network. Our temporal sampling results perform better than the spatial sampling variants, as shown in middle section of Table 6.2.

We combine each cue model by using score level fusion variants, as described in the previous section. We explore the fusion in two successful settings. The standard fusion operation is applied by averaging the softmax outputs, then evaluating the prediction. The weighted fusion alternative is applied by weighing each cue model with its respective performance on the validation set, then normalizing the score vector and applying prediction. Our empirical results in bottom two rows of Table 6.2 show that the weighted model performs better than the standard fusion setup.

6.3. Spatio-Temporal Multi-Cue Fusion

So far, we have used the spatial and temporal cues separately. We also explore the setting where we use both sampling methods at once and observe the accuracy gains over every single setting. This is applied by using the spatial cue models and temporal cue models, a total of six models, in the final fusion algorithm. The usage of the weighted fusion approach provides the best results with the %94.94 accuracy. Results are provided in the rightmost column of Table 6.2.

Table 6.2. Classification accuracy results of the sampling and fusion settings. Three different settings are provided in the table. From left to right, (1) Single cue spatial sampling results, (2) Active Window Based Temporal Sampling applied to each crop, and (3) Spatial & Temporal settings are combined in one setting. Note that the bottom two rows include the fusion result of the above three models in each setting.

	Spatial		Temporal		S&T Combined	
Setting	Acc@1	Acc@5	Acc@1	Acc@5	Acc@1	Acc@5
Body	75.73	93.88	81.83	96.02	86.91	98.17
Hand	86.25	97.61	88.70	97.59	91.73	98.72
Face	24.27	44.45	37.00	57.89	39.12	59.33
Fusion	90.63	98.92	93.88	**99.65**	94.47	**99.78**
Weighted Fusion	**92.18**	**99.27**	**94.03**	99.56	**94.94**	99.76

With our Spatio-Temporal Sampling setting, we have successfully outperformed against the baseline method with %16.09 accuracy margin. We also compare our results with the best classification result using a handcrafted method [13]. Despite the handcrafted method involving more complex feature engineering and usage of more than ten times spatial resolution (640×640), our mixed convolutional network method outperforms it. We have provided the comparison with the baseline Bosphorus Sign 22k benchmarks [5] in Table 6.3.

Table 6.3. Comparison with the baseline approaches IDT and MC3_18 model.

Method	Acc-1	Acc-5
Baseline IDT [5]	88.53	-
Baseline MC3_18 [5]	78.85	94.76
Weighted Fusion - S&T Combined	**94.94**	**99.76**

6.4. Discussion and Analysis

There is a need to do a comparative analysis to investigate the contribution of each model in the fusion result. We adopt the Top N accuracy metric to measure how often network predictions land in the top N ranks. Top-N results are expected to approach 1 as the N value approaches to the 1. We have already provided the Top-1 and Top-5 accuracy in previous sections.

We report the Top-N accuracy graphs in two settings in Figure 6.2. We firstly include the comparison of the single cue model settings, shown on the left side. The hand model keeps its advantage with increasing N values. On the right side, we compare the multi cue settings by incrementally adding each cue to the hand setting. We can see the increasing accuracy trend of fusion results with the addition of each cue.

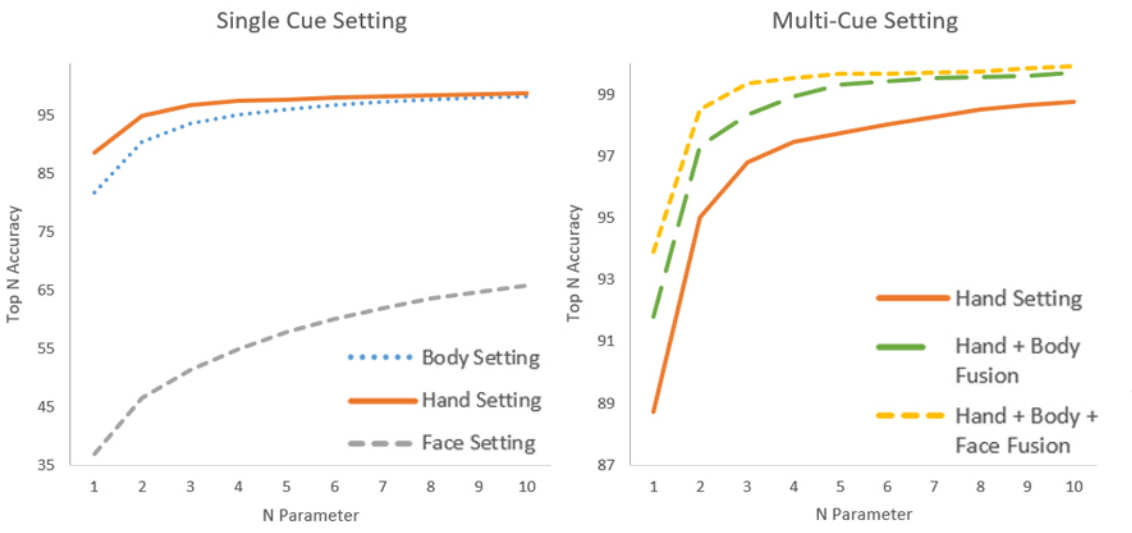

Figure 6.2. Comparison of the models in the Top-N accuracy setting. The horizontal axis denotes the increasing N value, and the vertical axis denotes the accuracy value. The first plot shows the Single Cue Setting comparison, and the second plot shows the Multi Cue Setting additive comparison. Despite the difference in single setting performance, each cue boosts the fusion results.

6.4.1. Spatial Ablation Study

We have investigated the cue with the most effect on the fusion result and generated an ablation analysis. We have applied fusion to each combination of two settings, then subtracted the result from the final fusion result to calculate the effect of the excluded cue to the fusion. In Table 6.4, we share the result of the calculation of the single cue effect. Hand cue results in the most effect on the final fusion with 9.22% accuracy followed with the body and face models, respectively.

Table 6.4. Effects of excluding individual cue units from the final fusion model. Using the different two cue settings and their performance, we infer to the excluded setting and its effect on the final mix.

Setting	Accuracy	Excluded Cue	Effect (%)
Body + Hand	91.80	Face	2.08
Body + Face	**84.66**	Hand	**9.22**
Hands + Face	88.70	Body	5.18

Our analysis continues on the gloss based setting, where we evaluate each cue performance on each gloss sign. Since single gloss is used, we have adopted the F1-Score evaluation metric, which is more sensitive to the false positive and false negative values, and more suitable for the single gloss evaluation.

Gloss Based Cue Comparison. We apply the gloss based evaluation on the hand and body cue models and share the sign glosses where the difference is most noticeable. We have also applied score fusion and share fusion performance to see if fusion can capture the successful cue feature. Results are shared in the Table 6.5.

Table 6.5. F1-score comparison for the top ten sign glosses that hand sampling outperforms body sampling. (Sorted in the alphabetical order)

Sign Gloss	Hand	Body	Fusion	Sign Gloss	Hand	Body	Fusion
Aspirin	0.62	0.00	0.67	Internet_2	1.00	0.33	1.00
Deposit(v)_2	0.89	0.25	1.00	Noon	0.91	0.33	1.00
Exchange(v)	0.57	0.00	1.00	Shout(v)_2	0.91	0.33	0.91
Head	0.89	0.29	1.00	Sleep(v)	0.62	0.00	0.67
Identify(v)	0.89	0.00	1.00	Turn(v)	1.00	0.40	0.89

We select the one sample of the EXCHANGE(v) sign to provide a detailed analysis of each cue model. EXCHANGE(v) sign gloss stems from the B̈ozdurmakẅord in Turkish origin that corresponds to the monetary exchange. Sign performance is started by touching the index finger with the thumb finger to signal a monetary term, then two hands moved close to the chest region and moved back and forth to each other. Whereas the Hand cue model predicts the class successfully, the Body cue model makes a true prediction at the third try. As seen in Figure 6.3, we compare confused cases as follows:

- COST and SAVINGS glosses are also started performing with the monetary hand pose, followed by both hands standing in the chest region but differs in terms of hand shape and movement. This similarity stems from the semantic similarity between the sign meanings.
- LAST WEEK gloss does not involve a monetary term hand shape yet follows with both hands moving to each other near the chest region.
- GROSS INCOME starts with a monetary term but has multiple unique hand shape patterns.

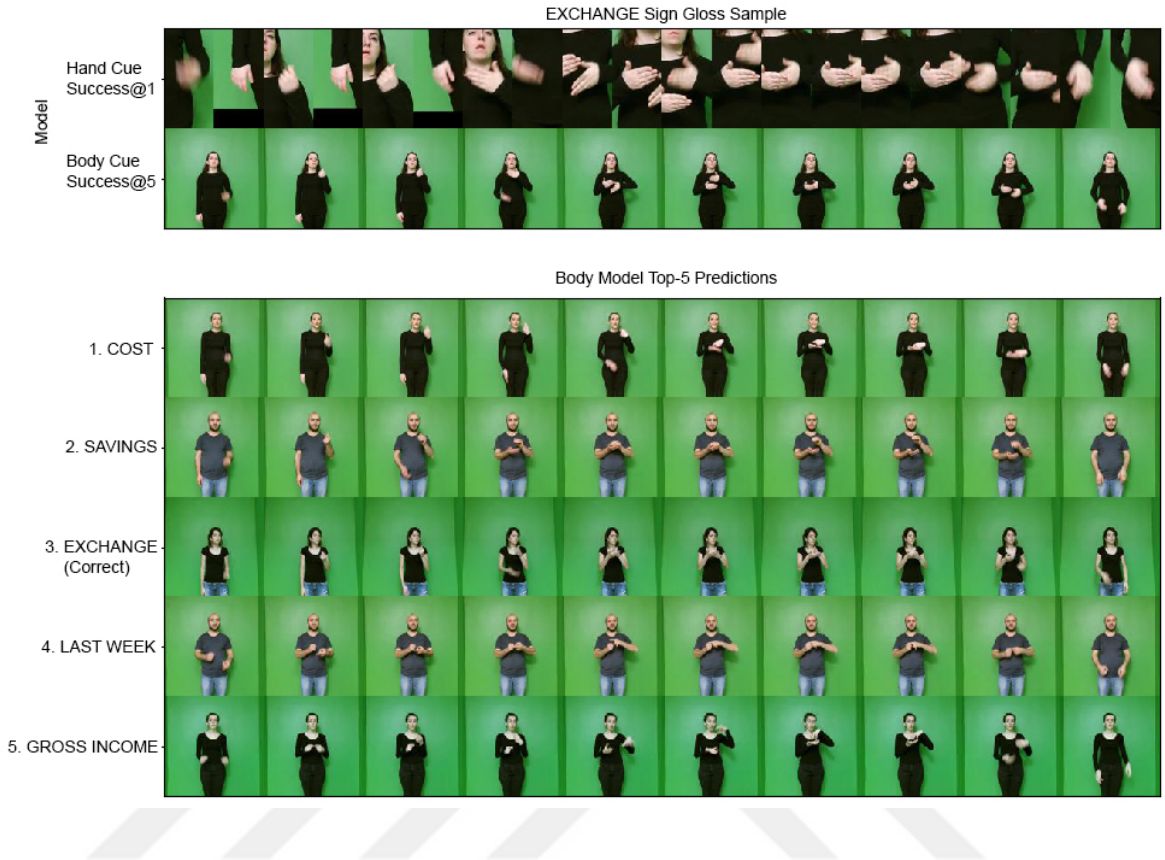

Figure 6.3. Class confusions of the EXCHANGE(v) sign gloss for the Body cue model.

It is seen that except for the LAST WEEK gloss, other confused cases include semantic similarity that results in similarity in the performance. The hand model able to capture different hand movements trajectory and hand shape, whereas the Body model shows confusion. We suspect the difference in capturing the fine details occurs because of the high spatial region that the Hand cue model has access to during the training phase.



Figure 6.4. Class confusions of the IDENTIFY(v) sign gloss for the Body cue model.

To evaluate the success of the body model carefully, we inspect the IDENTIFY sign shared in Figure 6.4. We can see that the IDENTIFY sign is performed by moving on hand to the near the head region. Detailed movement occurs in the hand shape, which is applied by having the index finger in the upper pose compared to the other fingers.

By evaluating the difficult cases, we can see a similar body-hand relative position in the HEAD gloss. With a slightly different position, EAT, PSYCHOLOGY, and the PHONE gloss signs also involve the one hand dominantly using a few fingers up position. The model needs a detailed look to distinguish between these signs, which is achieved via the hand model's high resolution, which results in successful classification.

6.4.2. Fusion Model Evaluation

How does fusion perform when capturing different cue models? To inspect the effect, we share the confusion matrix for the shared ten classes for the hand, body cues, and their corresponding fusion result.

Due to size limitations, we share a confusion matrix in a different format with $N \times M$ rectangle. Rows represent the $N = 10$ classes that we already share in the Table 6.5. These ten glosses are also shared as the first ten columns. Therefore, correct classifications generate a diagonal starting from the $0, 0$ and to $10, 10$. The strength of the blue color tone represents the number of predictions for the given column label, where the sample is taken from the row label. After each cue model makes predictions, we include all of the false predictions as extra columns to the given confusion matrix. This representation provides an efficient way for us to share the confusion results for the given classes.The modified confusion matrix is shared in Figure 6.5.

Using the modified confusion matrix, we can evaluate the Fusion model's false predictions. Firstly, we can see that the SLEEP(v) gloss is confused with the LIE(v) gloss in all of the cases. Comparing their Turkish original glosses, we can see that SLEEP gloss is derived from the UYUMAK sign, and LIE(v) gloss is derived from the YATMAK sign. The YATMAK/LIE(v) sign is used as a preface to sleeping in Turkish origin, and can also be used as a synonym to sleep word, thus conveys very similar semantic meaning. The other confusions are ASPIRIN and ANTIBIOTICS_3, which also have similar signer performance and representing taking medicine. Finally, we can see that TURN(v) gloss is confused with the TURN(v)_v2 gloss, which is two different representations of the same gloss and acceptable in real-world applications.

We conclude that the Multi Cue Fusion model was able to capture the success of the hand cue method and even outperform both models in a few sign glosses. Despite the Body model's bad performance, the Fusion model manages to separate noisy examples.

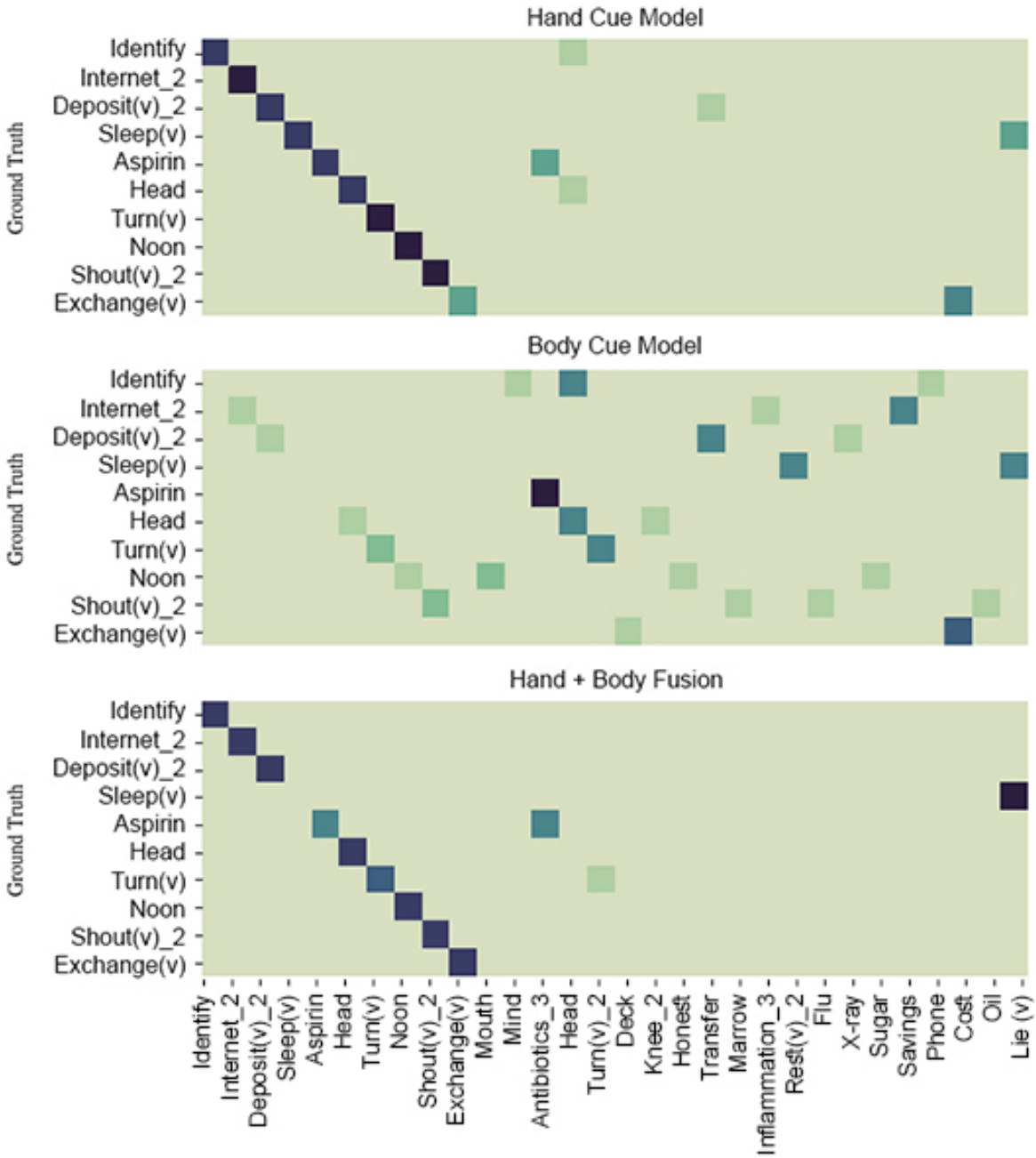

Figure 6.5. Modified confusion matrices of the top ten classes where the Hand model has an advantage over the Body Model. The rows represent the top ten classes, and the first ten columns also correspond to the same classes. The color strength represents the number of total predictions of a given column, where the row is the ground truth. The diagonal starting from $0, 0$ to $10, 10$ represents the true predictions. The rest of the column data represent other confused classes.

7. CONCLUSION

In this thesis, we worked on the Isolated Sign Language Recognition problem and proposed the score-level multi cue fusion method. We have focused on the important cues in the given data; hand movements, body alignment, and face properties. We have closely followed previous work for replicating the setting of the convolutional neural network-based SLR system for classifying between the 744 different isolated sign glosses.

Our contribution differs from the previous work [5] by our focus on both spatial and temporal cues. We have applied a preprocessing pipeline by carefully investigating the sampling operation and improving it by adding a cue-based sampling. We follow the SLR cue regions that are presented to be beneficial in previous work to decide to do croppings, such as body, single dominant hand, both hands, and face. We tailor each cue in our settings with the experimenting combinations of the given cues. Our final framework includes the baseline body setting, the face setting, and a mixed hand-setting, which combines the dominant hand, and both hand concatenation only when both hands are active. Our selected setting managed to perform better than the baseline.

We have investigated the multi cue fusion process. For each selected cue, we have trained 3D CNN and combined them through the score-level fusion. We have achieved the score-level fusion at the test level; after feeding the network the input data, we have managed to extract scores from the network after the softmax operation. We also tried to extract the impact of each cue setting from the validation set and then weighted the cues accordingly during the fusion operation. Our experiment results show that we have successfully outperformed the baseline results on the Bosphorus Sign 22k Turkish Isolated Sign Language dataset [5].

We have compared each cue and multi cue settings using the Top-N accuracy evaluation metric. Reported results show the significant advantage acquired with fusion via the addition of each model, which is a steady trend with an increasing N value. We have provided a gloss based comparison to the inspect how hand and body cues affect the final fusion result. We have demonstrated that each cue has generated its expertise during the training phase, which provides a benefit to the fusion and Sign Language Recognition.

As future work, a convolutional attention mechanism could also be implemented to explore intra-cue features and possibly boost the fusion results more. We hope that our score-level multi cue fusion approach can be used for other sign language based tasks such as the translation. In the Sign Language Translation task, it is shown that the state-of-the-art transformer model was able to capture temporal properties with the self-attention mechanism [71]. Future work on Sign Language Translation can use our cue extraction pipeline and use each cue model as a feature extractor to achieve better results.

Understanding sign languages is an essential task for generating robust SLR systems. One major obstacle is using language-dependent sets, which generates a very restricted domain for SLR researchers to proceed on. We hope that our work can help SLR progress to extract the language-independent cues to further help with uncovering more details, and close the gap between the different Sign Languages.

REFERENCES

1. Tran, D., H. Wang, L. Torresani, J. Ray, Y. LeCun and M. Paluri, "A Closer Look at Spatiotemporal Convolutions for Action Recognition", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6450–6459, 2018.

2. Li, D., C. Rodríguez, X. Yu and H. Li, "Word-level Deep Sign Language Recognition from Video: A New Large-scale Dataset and Methods Comparison", *arXiv preprint arXiv:1910.11006*, pp. 1–8, 2019.

3. Vaezi Joze, H. and O. Koller, "MS-ASL: A Large-Scale Data Set and Benchmark for Understanding American Sign Language", *The British Machine Vision Conference (BMVC)*, pp. 1–10, September 2019.

4. Zhang, J., W. Zhou, C. Xie, J. Pu and H. Li, "Chinese sign language recognition with adaptive HMM", *2016 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, 2016.

5. Özdemir, O., A. Kındıroğlu, N. Camgoz and L. Akarun, "BosphorusSign22k Sign Language Recognition Dataset", *Language Resources and Evaluation Conference (LREC)*, pp. 1–6, 2020.

6. Hanke, T., S.-E. Hong, S. König, G. Langer, R. Nishio and C. Rathmann, "Designing Elicitation Stimuli and Tasks for the DGS Corpus Project", *Theoretical Issues in Sign Language Research (TISLR)*, pp. 1–3, 2010.

7. Kadous, M., "Machine Recognition of Auslan Signs Using PowerGloves: Towards Large-Lexicon Recognition of Sign Language", pp. 1–9, 1970.

8. Gaolin Fang and Wen Gao, "A SRN/HMM system for signer-independent continuous sign language recognition", *Proceedings of Fifth IEEE International Conference on Automatic Face Gesture Recognition*, pp. 312–317, 2002.

9. Farhadi, A., D. Forsyth and R. White, "Transfer Learning in Sign Language", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8, 2007.

10. Parton, B., "Sign Language Recognition and Translation: A Multidisciplined Approach From the Field of Artificial Intelligence", *Journal of Deaf Studies and Deaf Education*, Vol. 11, pp. 94–101, 02 2006.

11. Liwicki, S. and M. Everingham, "Automatic recognition of fingerspelled words in British Sign Language", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 50–57, 2009.

12. Holden, E.-J. and R. Owens, "Visual Sign Language Recognition.", *Lecture Notes in Computer Science: Multi-Image Analysis*, pp. 270–288, 2000.

13. Wang, H., A. Kläser, C. Schmid and L. Cheng-Lin, "Action Recognition by Dense Trajectories", *IEEE Conference on Computer Vision Pattern Recognition*, pp. 3169–3176, 2011.

14. Soomro, K., A. Zamir and M. Shah, "UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild", *arXiv preprint arXiv:1212.0402*, pp. 1–6, 2012.

15. Kuehne, H., H. Jhuang, E. Garrote, T. Poggio and T. Serre, "HMDB51: A Large Video Database for Human Motion Recognition", *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2556–2563, 2011.

16. Hernandez-Rebollar, J. L., R. W. Lindeman and N. Kyriakopoulos, "A multi-class pattern recognition system for practical finger spelling translation", *Proceedings of the Fourth IEEE International Conference on Multimodal Interfaces*, pp. 185–190, 2002.

17. Zhang, L.-G., Y. Chen, G. Fang, X. Chen and W. Gao, "A vision-based sign language recognition system using tied-mixture density HMM", *Proceedings of the*

6th International Conference on Multimodal Interfaces, pp. 198–204, 2004.

18. Liang, R.-H. and M. Ouhyoung, "A Sign Language Recognition System Using Hidden Markov Model and Context Sensitive Search", *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST)*, pp. 59–66, 1996.

19. Vogler, C. and D. Metaxas, "Parallel Hidden Markov Models for American Sign Language Recognition", *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, Vol. 1, pp. 116–122, 1999.

20. Jong-Sung Kim, Won Jang and Zeungnam Bien, "A dynamic gesture recognition system for the Korean sign language (KSL)", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, Vol. 26, No. 2, pp. 354–359, 1996.

21. Vogler, C. and D. Metaxas, "Handshapes and Movements: Multiple-Channel American Sign Language Recognition", *Lecture Notes in Computer Science*, Vol. 2915, pp. 247–258, 2003.

22. Cooper, H., B. Holt and R. Bowden, "Sign Language Recognition", *Visual Analysis of Humans*, pp. 539–562, 2011.

23. Lang, S., M. Block-Berlitz and R. Rojas, "Sign Language Recognition Using Kinect", *International Conference on Artificial Intelligence and Soft Computing*, pp. 394–402, 2012.

24. Hore, S., S. Chatterjee, V. Santhi, N. Dey, A. S. Ashour, V. Balas and S. Fuqian, "Indian Sign Language Recognition using Optimized Neural Networks", *Advances in Intelligent Systems and Computing*, Vol. 455, pp. 553–563, 2016.

25. Kumar, P., H. Gauba, P. Roy and D. Dogra, "A Multimodal Framework for Sensor based Sign Language Recognition", *Neurocomputing*, Vol. 259, pp. 21–38, 2017.

26. Koller, O., S. Zargaran, H. Ney and R. Bowden, "Deep Sign: Enabling Robust

Statistical Continuous Sign Language Recognition via Hybrid CNN-HMMs", *International Journal of Computer Vision*, Vol. 126, pp. 1311–1325, 2018.

27. Suri, K. and R. Gupta, "Continuous sign language recognition from wearable IMUs using deep capsule networks and game theory", *Computers & Electrical Engineering*, Vol. 78, pp. 493–503, 2019.

28. Neto, G., G. Junior, J. Almeida and A. Paiva, "Sign Language Recognition Based on 3D Convolutional Neural Networks", *International Conference Image Analysis and Recognition*, pp. 399–407, 2018.

29. Eepuri, K., P. Kishore, T. Maddala and D. Anil Kumar, "3D Sign Language Recognition with Joint Distance and Angular Coded Color Topographical Descriptor on a 2 – Stream CNN", *Neurocomputing*, Vol. 372, pp. 40–54, 2019.

30. He, K., X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 7, pp. 770–778, 2015.

31. Szegedy, C., V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2016.

32. Alpaydin, E., *Introduction to Machine Learning*, MIT Press, 2nd edn., 2010.

33. Goodfellow, I., Y. Bengio and A. Courville, *Deep Learning*, MIT Press, 2016.

34. Salimans, T., I. Goodfellow, W. Zaremba, V. Cheung, A. Radford and X. Chen, "Improved Techniques for Training GANs", *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 2234–2242, 2016.

35. Kingma, D. and M. Welling, "Auto-Encoding Variational Bayes", *International Conference on Learning Representations (ICLR)*, pp. 1–8, 2013.

36. Kullback, S., *Information Theory and Statistics*, Dover Publications, 1997.

37. Rumelhart, D., G. Hinton and R. Williams, "Learning Internal Representation by Back-Propagation Errors", *Nature*, Vol. 323, pp. 533–536, 1986.

38. Kingma, D. and J. Ba, "Adam: A Method for Stochastic Optimization", *International Conference on Learning Representations (ICLR)*, pp. 1–10, 2014.

39. Girosi, F., M. Jones and T. Poggio, "Regularization Theory and Neural Networks Architectures", *Neural Computation*, Vol. 7, pp. 219–269, 1995.

40. Opitz, D. and R. Maclin, "Popular Ensemble Methods: An Empirical Study", *Journal of Artificial Intelligence Research*, Vol. 11, pp. 169–198, 1999.

41. Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", *Journal of Machine Learning Research*, Vol. 15, pp. 1929–1958, 2014.

42. LeCun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel, "Backpropagation Applied to Handwritten Zip Code Recognition", *Neural Computation*, Vol. 1, No. 4, pp. 541–551, 1989.

43. Gatys, L., A. Ecker and M. Bethge, "A Neural Algorithm of Artistic Style", *arXiv preprint arXiv:1508.06576*, pp. 1–12, 2015.

44. Ioffe, S. and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", *International Conference on Machine Learning (ICML)*, Vol. 37, pp. 448–456, 2015.

45. Ulyanov, D., A. Vedaldi and V. Lempitsky, "Instance Normalization: The Missing Ingredient for Fast Stylization", *arXiv preprint arXiv:1607.08022*, pp. 1–4, 2016.

46. Karpathy, A., G. Toderici, S. Shetty, T. Leung, R. Sukthankar and L. Fei-Fei,

"Large-Scale Video Classification with Convolutional Neural Networks", *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1725–1732, 2014.

47. 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, 2017.

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

49. Galvin, B., B. McCane, K. L. Novins, D. Mason and S. Mills, "Recovering Motion Fields: An Evaluation of Eight Optical Flow Algorithms", *British Machine Vision Conference (BMVC)*, pp. 1–10, 1998.

50. Dalal, N., B. Triggs and C. Schmid, "Human Detection Using Oriented Histograms of Flow and Appearance", *Proceedings of IEEE European Conference on Computer Vision*, Vol. 3952, pp. 428–441, 2006.

51. Laptev, I., M. Marszalek, C. Schmid and B. Rozenfeld, "Learning realistic human actions from movies", *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8, 2008.

52. Gaidon, A., Z. Harchaoui and C. Schmid, "Activity Representation with Motion Hierarchies", *International Journal of Computer Vision*, Vol. 107, No. 3, p. 219–238, May 2014.

53. Lowe, D., "Distinctive Image Features from Scale-Invariant Keypoints", *International Journal of Computer Vision*, Vol. 60, pp. 91–110, 2004.

54. Simonyan, K. and A. Zisserman, "Two-Stream Convolutional Networks for Action Recognition in Videos", *Advances in Neural Information Processing Systems*,

Vol. 1, pp. 568–576, 2014.

55. Ji, S., W. Xu, M. Yang and K. Yu, "3D Convolutional Neural Networks for Human Action Recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 35, pp. 495–502, 2010.

56. Kamnitsas, K., C. Ledig, V. Newcombe, J. Simpson, A. Kane, D. Menon, D. Rueckert and B. Glocker, "Efficient Multi-Scale 3D CNN with fully connected CRF for Accurate Brain Lesion Segmentation", *Medical Image Analysis*, Vol. 36, pp. 61–78, 2016.

57. Molchanov, P., S. Gupta, K. Kim and J. Kautz, "Hand gesture recognition with 3D convolutional neural networks", *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1–7, 2015.

58. Maturana, D. and S. Scherer, "VoxNet: A 3D Convolutional Neural Network for real-time object recognition", *International Conference on Intelligent Robots and Systems (IROS)*, pp. 922–928, 2015.

59. Tran, D., H. Wang, L. Torresani and M. Feiszli, "Video Classification with Channel-Separated Convolutional Networks", *IEEE International Conference on Computer Vision (ICCV)*, pp. 5552–5559, 2019.

60. Ghadiyaram, D., D. Tran and D. Mahajan, "Large-Scale Weakly-Supervised Pre-Training for Video Action Recognition", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12038–12047, 2019.

61. Wu, C.-Y., R. Girshick, K. He, C. Feichtenhofer and P. Krähenbühl, "A Multigrid Method for Efficiently Training Video Models", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 153–160, 2019.

62. Feichtenhofer, C., H. Fan, J. Malik and K. He, "SlowFast Networks for Video Recognition", *IEEE International Conference on Computer Vision (ICCV)*, pp.

6202–6209, 2018.

63. Zhou, H., W. Zhou, Y. Zhou and H. Li, "Spatial-Temporal Multi-Cue Network for Continuous Sign Language Recognition", *AAAI Conference on Artificial Intelligence*, pp. 13009–13016, 2020.

64. Cao, Z., G. Hidalgo Martinez, T. Simon, S. Wei and Y. A. Sheikh, "OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–12, 2019.

65. Zhang, Z., "Microsoft Kinect Sensor and Its Effect", *IEEE Multimedia - IEEEMM*, Vol. 19, pp. 4–10, 2012.

66. Camgoz, N. C., S. Hadfield, O. Koller, H. Ney and R. Bowden, "Neural Sign Language Translation", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7784–7791, 2018.

67. 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, 12 2015.

68. Chai, X., H. Wanga, M. Zhoub, G. Wub, H. Lic and X. Chena, *DEVISIGN: Dataset and Evaluation for 3D Sign Language Recognition*, Tech. rep., Beijing, 2015.

69. Rabiner, L. and B. Juang, "An introduction to hidden Markov models", *IEEE ASSP Magazine*, Vol. 3, No. 1, pp. 4–16, 1986.

70. Camgoz, N. C., S. Hadfield, O. Koller and R. Bowden, "SubUNets: End-to-end Hand Shape and Continuous Sign Language Recognition", *IEEE International Conference on Computer Vision (ICCV)*, pp. 3075–3084, 2017.

71. Camgoz, N., O. Koller, S. Hadfield and R. Bowden, "Sign Language Transformers: Joint End-to-end Sign Language Recognition and Translation", *IEEE Conference*

on Computer Vision and Pattern Recognition (CVPR), pp. 10020–10030, 2020.

72. Chen, T. and C. Guestrin, "XGBoost: A Scalable Tree Boosting System", *International Conference on Knowledge Discovery and Data Mining*, p. 785–794, 2016.

73. Hinton, G., O. Vinyals and J. Dean, "Distilling the Knowledge in a Neural Network", *arXiv preprint arXiv:1503.02531*, pp. 1–8, 2015.

74. Alpaydin, E., "Multiple neural networks and weighted voting", *Proceedings of The 11th IAPR International Conference on Pattern Recognition*, Vol. 2, pp. 29–32, 1992.

75. Xie, Q., E. Hovy, M.-T. Luong and Q. Le, "Self-training with Noisy Student improves ImageNet classification", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10687–10698, 2019.

76. Plappert, M., R. Houthooft, P. Dhariwal, S. Sidor, R. Chen, X. Chen, T. Asfour, P. Abbeel and M. Andrychowicz, "Parameter Space Noise for Exploration", *International Conference on Learning Representations*, pp. 1–9, 2017.

77. Ulyanov, D., A. Vedaldi and V. Lempitsky, "Deep Image Prior", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9446–9454, 2017.

78. He, K., X. Zhang, S. Ren and J. Sun, "Identity Mappings in Deep Residual Networks", *European Conference on Computer Vision (ECCV)*, pp. 630–645, 2016.