WORD-LEVEL SIGN LANGUAGE RECOGNITION FROM VIDEOS

by

Al Amin Hosain
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Computer Science

Committee:

_____	Dr. Jana Košecká, Dissertation Co-Director
_____	Dr. Huzefa Rangwala, Dissertation Co-Director
_____	Dr. Parth Pathak, Committee Member
_____	Dr. Vivian Motti, Committee Member
_____	Dr. David Rosenblum, Department Chair
_____	Dr. Kenneth S. Ball, Dean, Volgenau School of Engineering

Date: _____ Fall Semester 2021
George Mason University
Fairfax, VA

Word-Level Sign Language Recognition From Videos

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Al Amin Hosain
Bachelor of Science
Chittagong University of Engineering and Technology, 2012

Co-Directors: Dr. Jana Košecká, Professor; Dr. Huzefa Rangwala, Professor
Department of Computer Science

Fall Semester 2021
George Mason University
Fairfax, VA

Copyright © 2021 by Al Amin Hosain
All Rights Reserved

Dedication

I dedicate this dissertation to my parents and sisters.

Acknowledgments

I would like to express my deepest gratitude to my advisors Dr. Huzefa Rangwala and Dr. Jana Košecká. Without their continuous support and guidance, I would not be able to complete my degree. I feel fortunate that Dr. Rangwala introduced me to the exciting area of sign language research. Thereafter, he provided me with invaluable technical and logistical supports. His suggestions helped to improve my writing and presentation skills. I learned a lot during the countless discussion with Dr. Košecká. She taught me how to become a better reader and how to ask the right questions in a research problem. She always encouraged me to set my sights on the most generic solution to a problem. I am also thankful to her for the valuable feedback and suggestions on my writings. I would like to also thank my committee members Dr. Parth Pathak and Dr. Vivian Motti for their guidance and helpful feedbacks towards my dissertation and presentation.

I am thankful to my collaborator Panneer Selvam Santhalingam and all of my lab mates for all those useful and fun discussion times. I am grateful to the office staffs at GMU Computer Science Department, for helping me with countless queries in the course of my program. I have worked here as a Teaching Assistant (TA) for a good amount of time. Thanks to all the Professors I have had the pleasure of working with, especially to Dr. Elizabeth White for helping me with valuable suggestions towards being a better TA. I would like to end with a heartfelt appreciation for all the teachers in my life.

Table of Contents

List of Tables

List of Figures

Abstract

WORD-LEVEL SIGN LANGUAGE RECOGNITION FROM VIDEOS

Al Amin Hosain, PhD

George Mason University, 2021

Dissertation Directors: Dr. Jana Košecká, Dr. Huzefa Rangwala

Sign language is the primary form of communication among Deaf and Hard of Hearing (DHH) individuals. Due to the absence of speaking capability, voice-controlled assistants such as Apple Siri or Amazon Alexa are not readily available to a DHH individual. An automated sign language recognizer can work as an interface between a DHH individual and the voice-controlled digital devices. Recognizing world-level sign gestures is the first step of an automated sign language recognition system. These gestures are characterized by fast, highly articulate motion of the upper body, including arm movements with complex hand shapes. The primary challenge of a world-level sign language recognizer (WLSLR) is to capture the hand shapes and their motion components. Additional challenges arise due to the resolution of the available video, differences in the gesture speed, and large variations in the gesture performing style across individual subjects. In this dissertation, we study different methods with the goal of improving video-based WLSLR systems.

Towards this goal, we introduced a multi-modal American Sign Language (ASL) dataset, GMU-ASL51. This publicly available dataset features multiple modalities and $13,107$ word-level ASL sign videos. We implemented machine learning methods using only video input

and a fusion of videos and body pose data. Usually, word-level sign videos have a varying number of frames, roughly ranging from 10 to 200, based on the source and type of the sign videos. To utilize the frame-wise representation of hand shapes, we implemented Recurrent Neural Network (RNN) models using per-frame hand-shape features extracted from a pre-trained Convolutional Neural Network (CNN). To further improve hand-shape representation, we proposed a hand-shape annotation method. This method can quickly annotate hand-shape images and simultaneously train a CNN model. We later used this model as a hand-shape feature extractor for the downstream sign recognition task.

Most of the information in sign language is conveyed using hand-arm movements. To prioritize the hand-arm related features, we proposed a pose guided feature localizing method from 3D feature maps of a 3D CNN model. This method can track the location of hands in a feature map space and extract representative features for hands in a sign video. To further leverage the idea of hand representation, we developed a graph-based hand modeling. This formulation sees the hands as graphs and attempts to model the finger structures using Graph Convolutional Network (GCN). When added with existing models, in an ensemble manner, the graph modeling yielded extra recognition performances.

In an attempt to build an interface between DHH individuals and voice assistants, this dissertation presents different building blocks of a video-based WLSLR. These range from developing a multi-modal dataset to improving state-of-the-art video classification models. We demonstrate the roles of hand shapes and pose data in several contexts of sign video modeling. We anticipate that the data and the insights emerged from this work will help to advance the research towards an automated sign language interpreter.

Chapter 1: Introduction

1.1 Motivation

According to the World Federation of the Deaf, 70 million deaf and speech-impaired people use sign languages in their day to day communication [2]. This vast community is collectively referred to as Deaf and Hard-of-Hearing (DHH). There are at least 300 sign languages in use around the world today [3]. Thoughts in sign languages are primarily conveyed using hand gestures, with occasional head and facial expressions. Due to the hearing and speech impairment, the DHH community faces many hardships in daily life. One such problem is the incapacity of using voice controlled personal digital assistants (PDA) such as Apple Siri, Amazon Alexa and Google Home. While the use of these devices is gradually becoming ubiquitous [4], due to the hearing and speaking disability, the DHH population can't benefit from it. Our goal is to alleviate this limitation by developing different components of a sign language recognition system. Such a system can work as an interface between a DHH individual and a digital system by translating sign gestures to text and vice versa. In this regard, we aim to improve different aspects of video based automated sign language recognition (SLR).

1.2 Sign Language Recognition (SLR)

The SLR task can be categorized into word-level sign language recognition (WLSLR) and sentence-level or continuous sign language recognition (CSLR). Figure 1.1 shows examples of these tasks. In the top image, we see three examples of the sign word `TIME` and in the bottom image, we observe a sentence-level sign video. The sentence-level video is composed

Figure 1.1: Three examples of word level sign class TIME (on top) taken from WLASL dataset [1]; and a sentence level sign video sample translating HOW LONG HAVE YOU LEARNED ASL (on the bottom) taken from www.handspeak.com.

of word-level signs and finger-spelling segments in a sequence. A finger-spelling segment spells a sequence of letters using hand shapes representing letters in the alphabet.

From the recognition perspective, an automated WLSLR system aims to recognize isolated sign words corresponding to particular upper body gestures, using hand and arm movement. Since a video is a sequence of image frames, this is analogous to a sequence classification problem. The top example in Figure 1.1 shows three samples for an example sign word `TIME`. These show a progression of various hand shapes with arm motion. These also demonstrate the appearance challenges such as different backgrounds and attires; variation in the execution of a sign class; lighting condition; and camera distances. The underlying machine learning (ML) system has to learn the hand-shape patterns and the arm motion for videos from a given sign class; it must also learn to differentiate among patterns from different classes. The objective in a CSLR system is to translate a sign video into an English sentence. The video usually contains sequences of sign words and finger-spelled segments. The goal of a finger-spelling task is to parse a continuous stream of hand shapes

corresponding to letters. The bottom example in Figure 1.1 shows a sample sentence-level sign video, which shows a sequence of word-level signs and a finger-spelling segment. Solving the WLSLR problem is a vital prerequisite to solve the CSLR problem. Considering the importance of a word-level sign recognizer, this dissertation focuses on improving different aspects of a WLSLR system.

1.3 Problem Statement

The goal of a WLSLR system is to identify the sign class when a video is given as input. Each of the classes represents a word. Formally, we assume access to a set of N pairs $\{V_i, W_i\}_1^N$ where V_i is a video and W_i is the corresponding class, and our goal is to learn a function ϕ such that $W_i = \phi(V_i)$. As an input to a WLSLR system, a video V_i can be seen as a sequence of features $\{f_1, f_2, ..., f_T\}$ where T is the number of frames in the video; and f_is are frame features such as RGB or pose based features. Learning the function ϕ is typically addressed as a sequence classification problem. We also followed a similar approach. In details, input to all our proposed methods is a sign video or a sequence representation extracted from the video; and the output is the corresponding class, which represents a word. Once the ϕ is learned, the function can predict the class label of an unseen sign video with reasonable accuracy. Improving this prediction accuracy is the ultimate goal of the learning process.

1.4 Thesis Organization

This thesis includes seven chapters excluding the current chapter. Chapter 2 presents the background methods that we used in our works and also discusses the prior works on sign language recognition. Chapters 3 through 7 describe our contributions for improving word-level sign video recognition. Chapter 8 presents conclusion and future research directions.

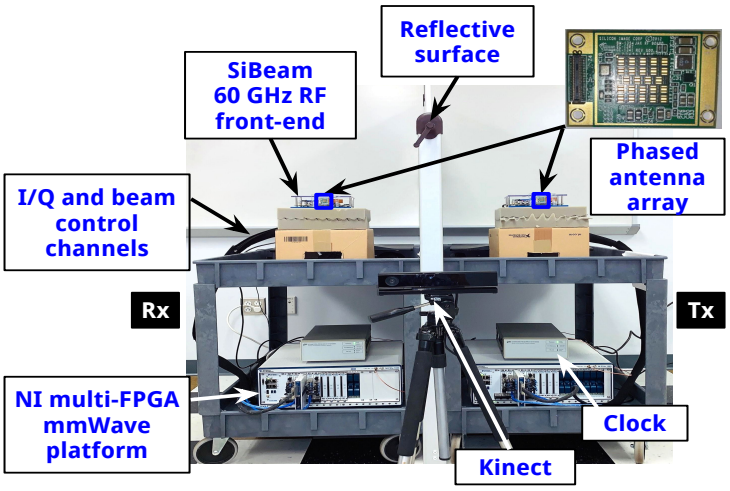


Figure 1.2: Multi-modal data collection system.

1.5 Contribution Summary

In Figure 1.1, the top image shows three examples of the word level sign for the English word `TIME`. Although taken from the same sign class, we observe noticeable execution variations: the orientation of the left arm, the bending motion of the right index finger, background variation such as clean background versus cluttered background, and different lighting conditions. A successful machine learning (ML) based recognizer has to deal with thousands of such word classes, possibly coming from hundreds of different sources [1, 5]. This gives us a hint about the required scale and modeling capacity of the underlying ML model. In addition to that, the model has to be trained on as much data as possible for a better generalization ability. To tackle these challenges, our contribution focuses on building sign language dataset and developing ML methods for word level sign recognition (WLSLR). Here we provide a summary of our contributions presented in this dissertation.

1.5.1 Multi Modal Benchmark Dataset

Chapter 3 details the construction of an American Sign Language (ASL) dataset. The previous datasets were neither public nor suitable for data driven deep learning (DL) based

methods [6–9]. In this work, our objectives were to build a multi-modal ASL dataset, demonstrate feasibility of DL methods and release the benchmark publicly.[1] We considered the modalities of the RGB video, depth, skeletal body key-points (3D body pose), and 60 GHz millimeter-wave wireless signals. Figure 1.2 shows the setup we used for the data collection system. Upon the completion of the dataset, we implemented baseline ML approaches to assess the modeling challenges in our dataset. We also proposed Recurrent Neural Network (RNN) models using pose data and fusion model using both RGB and pose data. Our experimental results showed, using only the pose data, a reasonable recognition accuracy of 81% can be obtained. This established the viability of the RNN based approach using the pose data.

1.5.2 Deep Hand Feature Based SLR

Most of the existing WLSLR datasets have only sign video-level annotation [1, 5, 10, 11]. To be more specific, for a given sign video we know the class label of the video but not the class label of the video frames. The underlying ML models are trained to capture both the per frame hand-shape features and the temporal context using this video-level ground truth or class label. This limits the extent to which an ML model can learn from individual frames in a sign video. We implemented a method that takes a sequence of per-frame features as input to learn the temporal dynamics in a sign video. We extracted these feature representation from a Convolutional Neural Network (CNN) that is pre-trained on 1 million hand-shape images [12]. We evaluated multiple RNN models using extracted hand representation along with pose information. Chapter 4 details this method.[2]

1.5.3 Learning Hand Shapes for SLR

To learn better hand-shape representation than a pre-trained source, we proposed a semi-automatic approach for annotating all of the frames in a dataset.[3] With a little manual

[1] https://github.com/amin07/GMU-ASL51
[2] https://github.com/amin07/DeepFt-Based-ASL-Rec
[3] https://github.com/amin07/FineHand

effort, this method can learn robust hand-shape features, which significantly improves the downstream SLR task. We started with a small set of manually annotated frames, and iteratively expanded the number of annotated frames using CNN prediction and re-training. Upon the availability of adequate hand-shape annotations, we developed Long Short Term Memory (LSTM) based fusion networks that outperformed the baseline methods mentioned in Chapter 3 and 4. This method is detailed in Chapter 5.

1.5.4 Enhancing SLR using Pose Guided Pooling

While a 2D CNN learns form an image input, a 3D CNN can directly learn from a video input. This allows 3D CNN models to learn in an end to end manner from large scale sign video datasets [5, 13]. Although modeling at large scales, these approaches disregard the importance of hand-shapes in a sign video. To better represent the hand shapes in sign video modeling, we proposed an enhanced 3D CNN model that can look more closely at the hand region. We utilized body-pose information to localize feature maps associated with important hand locations such as elbows and palms. Our experimental results show that these localized features are more robust than the basic 3D CNN features in sign video modeling. We found also these better transferable to related domains. We describe this work in Chapter 6.[4]

1.5.5 Improving SLR using Representative Frames

The 3D CNN learns the spatial and the temporal features in videos simultaneously [1, 5, 13–15]. Hence, two sign classes with similar hand-arm motion and subtle hand-shape differences are difficult to distinguish. This is due to the fact that, 3D CNN takes input and processes the video as 3D tensors of different 3D resolutions through the layers of the network. This way, it learns *spatial-temporal* correlation of different video classes. We included a separate per-frame spatial modeling on the top of existing *spatial-temporal* features. We based this extra spatial source on a 2D CNN and a Graph Convolutional Networks (GCN). Our

[4] https://github.com/amin07/Pose-Pooling-3dConvNet

experiments showed better performance with this added feature stream. In addition to that, this graph based hand feature can disambiguate very similar looking sign video classes. We describe this work in Chapter 7.

Chapter 2: Background

The WLSLR task can be formulated as a supervised machine learning (ML) problem where a sign gesture video and the corresponding word label form a pair of training examples. The goal is to train an ML model such that it can identify the word class label of unseen gesture videos. Primarily, an ML algorithm must tackle two challenges regarding a gesture video: modeling the per-frame spatial appearances and capturing the temporal component across the sequence of frames. Together, these are known as *spatial-temporal* modeling. A commonly used approach before the deep-learning (DL) era was to first extract some engineered features and then utilize the Hidden Markov Models (HMM) to capture the temporal component [8, 9, 16]. Deep-learning based methods have replaced this approach using different combinations of Artificial Neural Networks (ANN). For example, in case of an RGB video input, the frame-level spatial modeling can be efficiently tackled using Convolutional Neural Networks (CNN) and the temporal part can be modeled with Recurrent Neural Networks (RNN). Another approach is to use 3D convolution directly on video inputs [6, 13–15, 17]. The 3D CNN models learn the spatial and the temporal features simultaneously.

To train an ML model in a supervised setting, a collection of training data is required. A video-based WLSLR system primarily depends on raw RGB input. However, some derived representations from RGB such as optical flow and body pose location also provide complementary input representation. Traditionally, 3D body pose, otherwise known as skeletal data, has been collected using motion capture systems [18–20]. Another approach, commonly referred as pose estimation, trains CNN models to extract body poses directly from RGB video input [21–24].

The aforementioned ML techniques and input representation serve as the building blocks of the implemented methods in this thesis. In the following section, we provide more detailed

background in the context of video-based sign gesture modeling.

2.1 Spatial Modeling

In the context of video modeling, the spatial feature refers to per-frame feature. Specifically for a sign video, we are interested in hand shape and arm orientation at each frame. Feature engineering refers to the idea of extracting useful representation from raw input data using various processing methods. For example, in the case of an RGB video frame, one could extract features like edges and corners that are useful for a specific downstream task. In gesture recognition, some examples of robust features from RGB video frames are Histogram of Oriented Gradient (HOG) features and SIFT features [9, 16, 25, 26]. Similarly, hand key-point based spatial features such as unit-vectors, angle between joints or motion features are useful in gesture recognition [10, 27]. As opposed to engineered features, Neural Networks take raw data as input and learn useful features from the data. In this way, Neural Networks can skip the feature engineering part.

2.1.1 Artificial Neural Networks

The most basic Artificial Neural Network (ANN) consists of an input layer, an output layer and possibly multiple hidden layers. Layers are composed of nodes, also referred as neurons. Usually a neuron at a layer is connected to another neuron at a subsequent layer using a weight parameter. Training the network refers to learning these parameters, using class labels of input data, in an iterative fashion. This process is known as back-propagation algorithm [28]. The training process stops when the error reduces to a considerable amount.

2.1.2 Convolutional Neural Networks

A popular variant of neural networks is the Convolutional Neural Network (CNN) that works on image-like input. As shown in the top-left diagram of Figure 2.1, the CNN expects an image or a matrix. The CNN looks at the image input using smaller patches, in a sliding

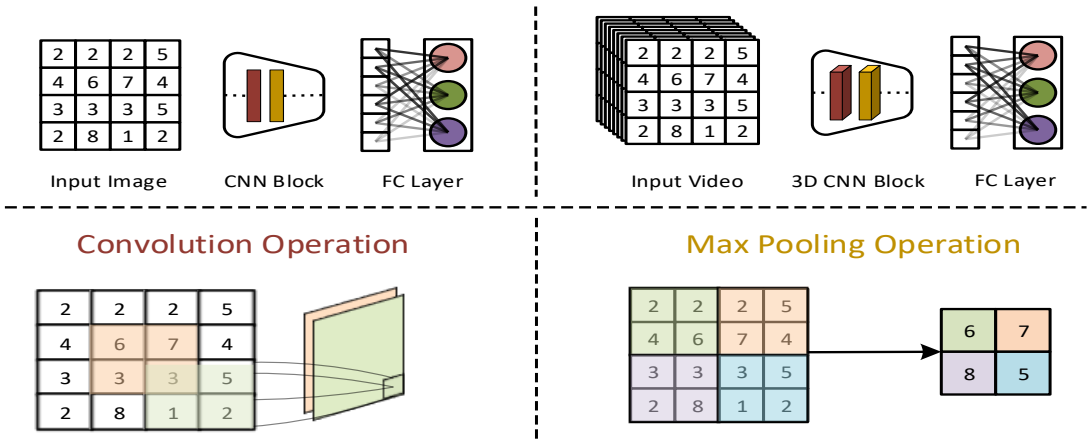


Figure 2.1: Schematic diagrams of different types of CNNs. Top-left schematic shows a basic CNN while top right shows a 3D CNN. Bottom-left image shows the convolution operation and the bottom-right shows an example of maximum pooling. Different colored square patches symbolize kernels.

window manner. These patches, also known as kernels, calculate a dot product with the input windows they slide over. The dot product values for one kernel at a layer yield one feature map for the next layer to process. This is known as the convolutional operation. The ultimate goal is to learn the optimal kernel values using the back-propagation algorithm. The convolution operation is usually followed by a non-linear activation function, which enables the CNN to learn highly nonlinear relationships in the data. Some of the popular choices for the activation function are Sigmoid function, Rectified Linear Unit (ReLU), Leaky ReLU, and Hyperbolic Tangent (tanh) function. There can be a sampling or pooling operation followed by the activation function. The purpose is to aggregate highly active regions in a 2D feature map. The pooling operation does not have any parameters to learn and reduces the resolution of the feature map from one layer to another. The bottom-left and bottom-right diagrams in Figure 2.1 depict a convolution and a maximum pooling operation respectively.

The aforementioned operations – convolution, activation and pooling – form an arrangement of operations. Usually a standard CNN has several layers of such arrangement. Finally, for a given input image, the network outputs a representation vector that can be used in subsequent tasks. The most common use is to send it to a fully connected (FC) layer for a classification purpose. The FC layer is a linear ANN where the input is the representation vector and the output is a score vector. The size of the score vector is the number of classes in the dataset. The raw scores in this vector are converted into prediction scores using the Softmax function.

$$\sigma(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \tag{2.1}$$

This function, as shown in Equation 2.1, takes the raw scores, x_i, as input and outputs a probability distribution over all the classes in the dataset.

3D Convolution In a standard CNN layer, each kernel convolves over the stack of 2D feature maps from the previous layer and produces a feature map for the next layer to process. The operation is given by Equation 2.2 where $F^l_{i,j}$ denotes the value of a feature map at l^{th} layer at location (i, j). The symbol $\odot$ represents the dot product between a kernel W and the associated feature map patch in the previous layer. An index (p, q) refers to a neighboring location centered at (i, j); $N(i, j)$ refers to the set of all neighboring indices. The size of $N(i, j)$ depends on the kernel size. For example, in case of a 3×3 kernel, there are 9 location elements in $N(i, j)$ uniformly centered at (i, j).

$$F^l_{i,j} = \sum_{(p,q)\in N(i,j)} W_{p,q} \odot F^{l-1}_{p,q} \tag{2.2}$$

$$F^l_{i,j,k} = \sum_{(p,q,r)\in N(i,j,k)} W_{p,q,r} \odot F^{l-1}_{p,q,r} \tag{2.3}$$

Imagine that we have a sequence of images or a video, as input. While a standard CNN learns useful features from a 2D image-like input, it faces difficulty when the input is a sequence of images, or a video. To solve this problem, 3D convolution was introduced in [14]. The key difference is that kernels are 3D and sub-sampling (pooling) layers work across three dimensions. The extra dimension, compared to the 2D convolution, represents the temporal axis. Equation 2.3 shows a 3D convolution operation. In this case, from each filter we get a 3D feature map and $F_{i,j,k}$ denotes a value at (i, j, k) location after the convolution operation. The dot product is between two 3D matrices (also referred to as tensors). The top-right depiction in Figure 2.1 shows an example schematic of a 3D CNN.

2.1.3 Graph Neural Networks

Image input can be seen as a graph where each pixel location is a node and the adjacent pixels are the neighbors. In this case, the neighborhood relationship is uniform, meaning that, given a distance, each pixel has same number of neighbors. However, if the uniform neighborhood is not guaranteed, the uniform kernel-based convolution operation is not straightforward. In other words, each node in the graph can have varying number of neighbors. Let's refer to this type of graph as irregular graph for this work.

Irregular graphs often arise in different domains such as the user-product graph in e-commerce, an user interaction graph in social media, a molecule graph in chemistry, and the body key-point graph in activity gesture modeling. To tackle such irregular graph modeling, Graph Convolutional Network (GCN) was invented [29–31]. GCN treats the input data as a graph and allows node-level feature representation while taking the graph structure into account. It keeps track of the neighboring nodes using an adjacency matrix input along with the node-level inputs. Figure 2.2 shows the fundamental operations in a GCN layer. We observe the adjacency matrix input along with the node-level feature inputs (as a visual diagram). Then a linear transformation of the node feature occurs. The coefficients in this transformation are the trainable parameters of the network. This is followed by an aggregation operation where each node receives responses from the neighboring nodes, using

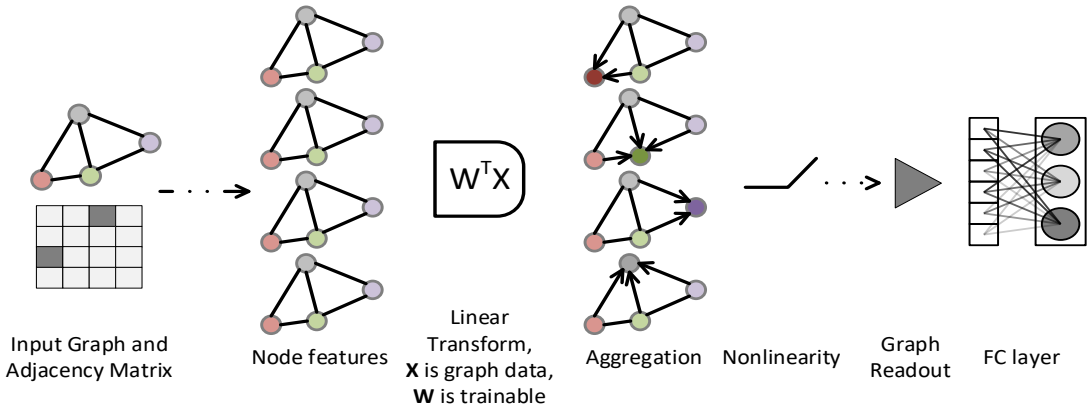


Figure 2.2: Graph Neural Network. One layer of operations is shown where each of the four graphs means each node's operation. In the aggregation step, each node uses information from the neighboring nodes using the input adjacency matrix. The whitish cells in the adjacency matrix represent neighboring connections.

neighborhood relationship in the input adjacency matrix. Finally, if a graph-level representation is required by the task at hand, a readout operation is applied on the learned nodes' representations. A readout is basically an aggregation of all the nodes' representations. Finally a non-linear activation function is applied.

$$F_i^l = \sigma(W_1 F_i^{l-1} + W_2 \sum_{j \in N(i)} F_j^{l-1}) \tag{2.4}$$

Equation 2.4 outlines the basic graph convolution operation. Here F_i^l represents feature representation of i^{th} node at l^{th} layer; $N(i)$ represents all neighboring nodes of i^{th} node; σ is the sigmoid non-linearity; and W_1 and W_2 are learnable parameters.

2.2 Temporal Modeling

The goal of the sign video modeling is to learn a feature representation for the whole input video. The spatial feature modeling only captures per-frame patterns in a sign video. Hence, the sequence of spatial features from the video frames needs to be further modeled

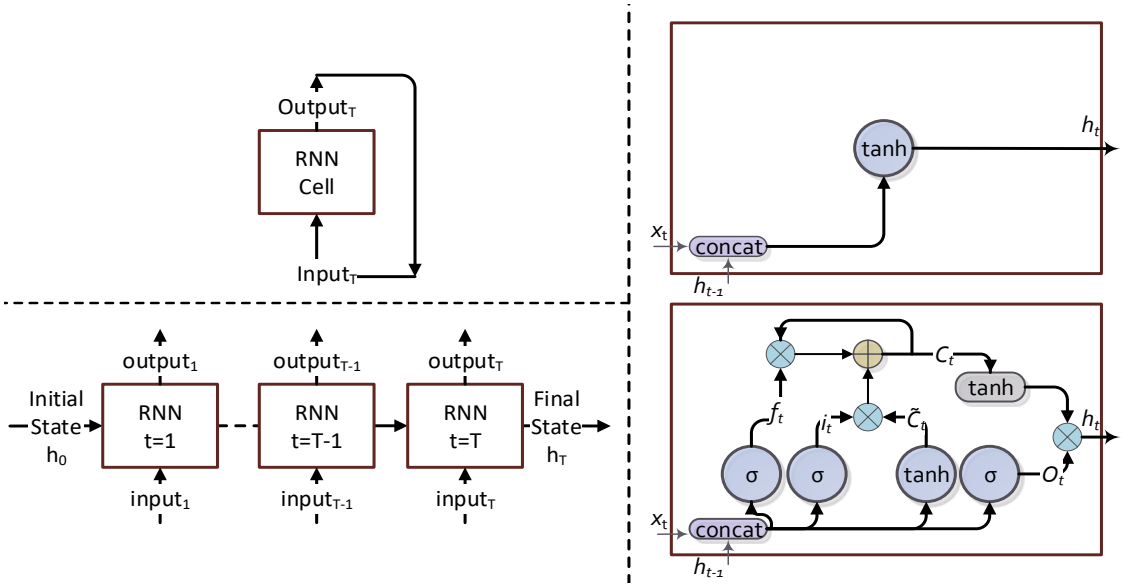


Figure 2.3: Different components of Recurrent Neural Networks (RNN). On the left side, an RNN network is shown in the loop form (top) and the unrolled form (bottom). On the right side, the internals of a simple RNN cell is shown on the top and an variant LSTM cell is shown in the bottom.

temporally. In this section, we outline ML models that are popularly used for temporal learning.

2.2.1 Recurrent Neural Networks

Recurrent Neural Networks (RNN) have shown success in modeling sequential patterns in data [32]. Such temporal patterns are often visible in sequential modeling problems such as text modeling, time-series modeling, event forecasting and video understanding. The RNN captures the sequential pattern in data by maintaining an internal state. Starting with an initial state, the RNN takes the sequence as input, at different time steps. At each time step, the state is improved using the current input. At the end, the RNN state represents an encoded summary of the input sequence and can be used in downstream tasks. The basic RNN has problems dealing with long term dependencies in data due to the vanishing gradient problem [33]. Some solutions to the vanishing gradient problem involve careful

initialization of network parameters or early stopping [34]. But the most effective solution is to modify the RNN architecture in such a way that there exists a memory state (cell state) at every time step; and to let the network decide what to remember and what to forget based on training data. This architecture is referred to as long short term memory (LSTM) network [35].

Long Short Term Memory While the basic RNN is a direct transformation of the previous state and the current input, the LSTM maintains an additional memory state and has mechanisms to update and use that memory. This is achieved by deploying four separate neural networks, also called gates. The bottom right diagram in Figure 2.3 depicts a cell of an LSTM network which shows input at the current time step x_t and the previous state h_{t-1} enter into the cell, and get concatenated. The forget gate processes it to remove unnecessary information, and outputs f_t which gets multiplied with the previously stored memory C_{t-1} and produces a refined memory for the current time. Meanwhile, the input and update gate process the concatenated input and convert it into a candidate memory for the current time step by element–wise multiplication. The refined memory and the proposed candidate memory of the current step are added to produce the final memory for the current step. This addition could render the output to be out of scale. To avoid that, a squashing function (hyperbolic tan) is used, which scales the elements of the output vector into a fixed range. Finally o_t, the output from output gate gets multiplied with the squashing function and produces the current time step output. The forget, input, update and output gates are represented by four circles and symbolized as f_t, i_t, $\tilde{C}_t$ and o_t, respectively.

$$
\begin{aligned}
f_t &= \sigma(W_f \times concat(h_{t-1}, x_t)), \quad i_t = \sigma(W_i \times concat(h_{t-1}, x_t)) \\
\tilde{C}_t &= tanh(W_{\tilde{C}} \times concat(h_{t-1}, x_t)), \quad C_t = (f_t \otimes C_{t-1}) \oplus (i_t \otimes \tilde{C}_t) \\
o_t &= \sigma(W_o \times concat(h_{t-1}, x_t)), \quad h_t = o_t \otimes tanh(C_t)
\end{aligned}
\tag{2.5}
$$

Equation 2.5 shows LSTM functions; where $\oplus$ and $\otimes$ represent element wise addition and multiplication respectively; $\times$ represents matrix multiplication, *concat* process means a concatenation of its input. σ and $tanh$ represents sigmoid and hyperbolic tan non-linearity respectively. The bottom-left diagram in Figure 2.3 shows how an RNN cell is processed through a network.

Gated Recurrent Units The Gated Recurrent Unit (GRU), another variant of RNN, was also proposed to tackle the vanishing gradient problem [36]. Instead of having an internal memory state and an output state like LSTM, the GRU has a single state and merges the input and the forget gate into one gate. This simpler version performs equally with the LSTM [37]. The authors in [37–39] attempted to identify the best variation with extensive experimental studies. It was found that, in general, the LSTM performs better than any variants and the GRU is compatible to it.

Attention Mechanism In general, the attention in machine learning (ML) refers to the idea of giving importance to some portion of the input. In a sequence modeling task, the input at each time step is not equally important. For example, all the frames in a sign video are not equally important to understand the sign. Hence it is ideal to treat the important parts with higher priority. When this prioritization is learned from the data, using back-propagation, the whole idea is referred as attention. The ultimate goal is to learn a set of scores that represents the weights between an input step and an output step [40, 41]. Another type of attention, know as the self-attention, attempts to enrich the representation of an input sequence by picking the most relevant parts from the same sequence [42].

2.3 Body Pose Data

The body-pose data refers to the coordinate location of vital body joints such as the wrist, elbow, the head, neck, of a human body in an image. This is also known as the joint key-point or the skeletal data. In the context of a sign video, the frame sequence of body

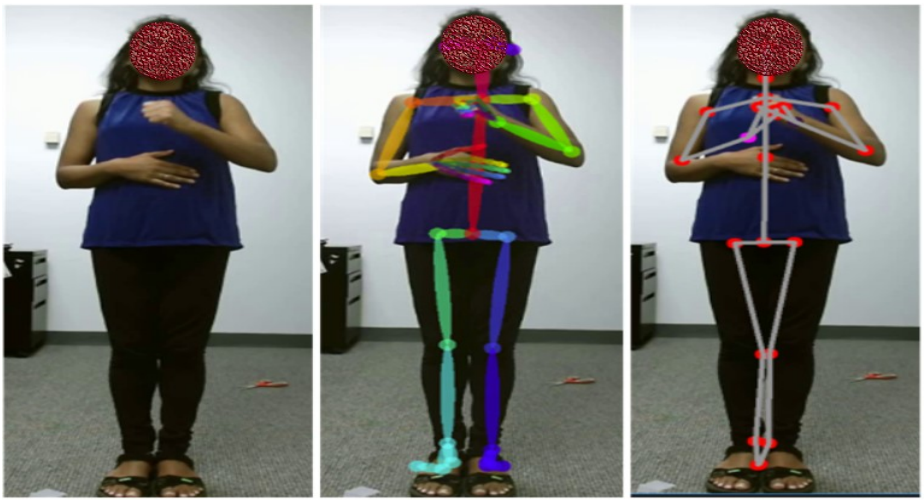


Figure 2.4: Pose data example. Left image shows an example frame from an ASL sign video. Middle image shows estimated poses from RGB using pose estimation methods. Right image shows pose collected with Kinect V2 sensor.

key-points forms a multi-dimensional time-series. This data provides high-level control over the human subjects in a video and suitable for modeling the human motion pattern. Pose data can be 2D or 3D based on the process of obtaining the data. While commercial depth sensors can provide 3D body poses using depth perception mechanisms [43], the 2D poses can also be obtained using ML based pose estimation methods [23]. Given an input image, a pose estimation method attempts to infer the information about some pre-selected locations of an entity in the image. The entity is task specific and some examples can be human, animals or objects. For example in case of human pose estimation, these locations are vital body joints. For object poses, one might be interested about the corners. For this thesis, we are only interested in human poses. Figure 2.4 shows examples of human body poses.

Sensor Based Pose Commercial depth sensors such as Microsoft Kinect, Intel RealSense, and LeapMotion provide RGB-D imaging of a scene [43, 44]. These devices use various depth perception techniques such as depth from stereo and depth from focus to estimate the distance of different objects in the scene respect to the camera. Along with these

techniques, machine learning is used to infer 3D coordinates of different body locations. One of the most widely used depth sensors is Microsoft Kinect. Along with the RGB and depth images, it provides 3D coordinate locations for 25 vital body joints [43].

Pose From RGB Image This approach formulates the pose estimation problem as a supervised machine learning tasks. With the availability of enough training data, deep learning based methods can be trained to infer the pose from unseen images of human [23, 45, 46]. The traditional methods follow either the top-down or the bottom-up approach. The top-down approach first detects any human subject in an image frame and then parses different body joint locations [47–49]. On the other hand, the bottom-up approach detects the body parts directly and associate the body parts to form a human skeleton [21–23, 50]. Figure 2.4 shows an example sign video and two types of poses: pose from RGB (middle) and pose from sensors (right).

2.4 Input Representation Fusion

In machine learning, the fusion strategy refers to the idea of using multiple input sources instead of a single source. The underlying assumption is to learn complementary information from different representation sources. For example, in the case of sign video understanding task, some input representations are the RGB, the depth, optical flows and body poses. The raw RGB data can provide the color, texture and shape information of arms and hands. On the other hand, the pose key-points and optical flow are great source of understanding the motion in a sign video. When using multiple input sources, the obvious question is how to aggregate responses from different sources. Two main approaches are the early fusion and the late fusion [51]. The early fusion feeds the aggregated input to the learning methods, while the late fusion concatenates the output representations of learning models for different input representations. In addition to that, there can be other variations of fusion developed for specific modeling task. For example, one might want to prioritize one input source over another or learn the prioritization from data.

2.5 Related Work on Activity Recognition

Activity recognition refers to the problem of understanding human motion in a scene. The specific formulation of an activity recognition problem depends on many factors: types of the scene such as indoor, outdoor or semi-outdoor; types of sensor such as camera, IMU or wireless signals; types of activities such as cooking activity, sports activity; and types of movements such as hand-only gestures or whole-body movements. Here we review the approaches related to video based activity recognition in indoor and outdoor scenes.

2.5.1 Video Based Approaches

Popular video-based action recognition approaches utilize modeled features from either or both the RGB and the body-pose (skeletal) data. For certain actions, the appearance information in single frame unambigously determines the actions, for others the motion is the discriminant cues. While the history of the modelling approaches is rich, we focus the review on more recent methods. To capture both the spatial and temporal signals in the video, Simonyan *et al.* [51] explored different ways of fusing the spatial and temporal information from a two stream approach, using an appearance and a flow convolutional networks. Similar multi-stream architecture was introduced in [52] for hand gesture recognition. This architecture is based on 2D convolution and sparse fusion of scores, from different channels of input streams where some of the channels are focused on hands. Later approaches explored the idea of 3D convolution for joint learning of *spatial-temporal* features [13, 17]. Authors in Inflated 3D ConvNet (I3D) [13] network expanded pre-trained convolutional kernels from 2D to 3D. The expanded 3D kernels allowed the I3D network to have better weight initialization and to bootstrap the video modeling. Some approaches focused on temporal modeling by either learning sparse frame sampling [53] or learning hierarchical features [54]; some focused on generating temporal candidate proposals [55]. Using unsupervised techniques is also a well studied area in gesture or activity recognition. These methods typically try to capture the temporal order similarities of full or sub-activities of similar kinds [56,57],

bypassing the need for more detailed labelling or ground truth annotation.

2.5.2 Pose Based Approaches

Activity Recognition using body pose (or skeletal) data is also a well studied problem [18, 58, 59]. Shahroudy *et al.* released a large-scale dataset for human activity recognition [18] and proposed an extension of long short term memory (LSTM) model which leverages group motion of several body joints to recognize human activity from skeletal data. A different adaptation of the LSTM model was proposed by Liu *et al.* where spatial interactions among joints was considered in addition to the temporal dynamics [58]. Veeriah *et al.* [60] proposed an LSTM network to capture the salient motion pattern of body joints. This method takes into account the derivative of motion states associated with different body joints. Some researchers treated the whole body as a hierarchical configuration of different body parts and proposed a hierarchical RNN to recognize human activities [59]. Several attention based models were proposed for human activity analysis [61, 62]. For example, Song *et al.* [61] used a spatial attention among joints' representation, and a temporal attention on time steps to model activity recognition problem. Liu *et al.* [62] used a global context memory to improve the recognition accuracy. Some approaches used skeletal sequences of body joints to develop new representations which capture *spatio-temporal* cues in videos [63, 64]. The goal of these methods is to generate an image-like representation using the pose data so that a pre-trained CNN model can be used in a sign modeling task. Some researchers attempted to use pose to extract features from ConvNet [65, 66]. These approaches deviate from the traditional use of poses of modeling sign videos. Instead, these methods use pose to localize a position in a feature map to extract features from. The feature map is usually generated from an RGB input video using a CNN network.

The graph convolutional network (GCN) was proposed to use the robustness of neural networks on graph structured data [29–31, 67, 68]. In other words, GCN allows convolutional modeling while taking the graph structure – node feature representation and neighbor relationship – in the input data into account. The body key-point or pose data, a representation

of human body parts using explicit graph structures, naturally fits the requirements of GCN modeling in human activity understanding. Most of the works towards this direction represented the whole body as a skeletal graph and implemented temporally aware GCN models [1, 69, 70]. These methods are primarily on body joints such as arm, wrist, torso and such more or less 25 joints. For example, authors in [70] proposed ST-GCN model – a *spatial-temporal* graph formulation across video frames – to model action videos from skeletal pose inputs. This formulation attempts to capture the *spatial-temporal* dynamics simultaneously using pose inputs. Li *et al.* proposed another pose-based baseline using temporal graph convolution to model ASL word-level signs [1]. Motivated by these applications, we aspire to apply the GCN modeling on hand shape patterns. Our idea is to use GCN on finger poses to complement existing RGB based *spatial-temporal* models' performance. Hence, we add this new feature stream with the existing features, in an ensemble manner, and improve the overall recognition performance.

2.6 Related Work on Sign Language Recognition

In this section, we describe some prior works on two types of sign language recognition task: word-level sign recognition (WLSLR) and sentence-level or continuous sign language recognition (CSLR).

2.6.1 Word Level Approaches

Most sign language recognition systems use RGB video data as input. These approaches model sequential interactions among video frames using Hidden Markov Models (HMM). Zafrullah *et al.* [8] used colored gloves (worn on hands) during data collection and developed an HMM based framework for ASL phrase verification. They also used hand crafted features from Kinect skeletal data and hand worn Accelerometer data [27]. Huang *et al.* [6] demonstrated the effectiveness of using Convolutional neural network (CNN) with RGB video data

for sign language recognition. The 3D CNN has been used to extract *spatio-temporal* features from videos [14]. Similar architecture was implemented for Italian gestures [71]. Sun *et al.* [72] hypothesized that all the RGB frames in a video are not equally important and assigned a binary latent variable to each frame in training videos for indicating the importance of a frame within a latent Support Vector Machine (SVM) model. Zaki *et al.* [16] proposed two new features with existing hand crafted features and developed the system using HMM based approach. Some researchers have used appearance-based features and divided the approach into sub units of RGB and tracking data, with a HMM model for recognition [9]. Most of the aforementioned approaches worked on smaller scale datasets collected in the laboratory settings. Besides, these datasets were not publicly available. Compared to these, in one of our contributions in this thesis, we developed a multi-modal American Sign Language (ASL) dataset and publicly released the dataset for research purpose.

Hand segmentation or recognition is also a well studied problem related to sign video recognition [12, 73, 74]. Some of these methods concentrated on hand detection rather than modeling hand shapes [74], some had different viewpoints such as egocentric views [73]. Koller *et al.* [12] trained a CNN for hand-shape modeling in an semi supervised manner. In general, these approaches attempt to learn per-frame hand representation. Since sign language is hugely depended on hands, this representation can be useful in the context of sign video modeling. Motivated by the potential of per-frame hand-shape learning, we envision to utilize this in sign video learning. However, traditional sign video datasets contain only video-level class labels, and the underlying ML model is dependent on this labels for learning frame features. To directly learn per-frame hand features in a supervised manner, we need frame-level class labels. In one of our contribution, we tackled this problem by implementing a per-frame hand-shape learning method that produced better representation than the other comparative methods.

2.6.2 Sentence Level Approaches

For sentence level parsing or CSLR task, most of the state-of-the-art methods are built upon the RWTH-PHOENIX-Weather corpus [75]. The corpus contains weather forecasts simultaneously interpreted into sign language. Videos in the corpus were recorded from German public TV and manually annotated using glosses on the sentence level. Koller *et al.* [12] trained a 22 layer deep convolutional neural network (CNN) with more than 1 million images from this corpus dataset. The data is weakly labeled with only video level annotation. The CNN model, which estimates the likelihood of hand-shape classes, is trained using the Expectation Maximization (EM) algorithm, jointly with Hidden Markov Model (HMM) for parsing sign gestures. Several later approaches focused on temporal modeling of sign sentences with the help of Connectionist Temporal Classification (CTC) loss [76–79]. For example, Cui *et al.* [76] proposed a method of staged optimization, where first an alignment proposal is learned for sign sequences and then those proposals were used as a stronger supervision in the final task. Pu *et al.* [77] proposed dilated convolutional kernels, also followed a pseudo-label based training, for capturing temporal dynamics. Guo *et al.* [79] also followed a similar paradigm with 2D convolutional pyramid features. Recently, Pu *et al.* [80] proposed a combination of 3D ConvNet and RNN encoder-decoder, with an alternative iterative training technique, to model continuous sign sentences.

2.7 Related Work on American Sign Language Recognition

There are several works in the literature that are directly related to the world-level recognition problem or WLSLR for American Sign Language (ASL). Early proposed works in this direction were linguistic property driven [11, 81–83]. In other words, these works focused on the inherent part of the language and did not exploit much of the machine learning (ML) techniques. Later approaches featured small datasets or restricted laboratory environment settings [6, 8, 27, 84]. More recently, two large scale word-level datasets were introduced to the community [1, 5]. Both of these datasets feature adequate sign variation to train large

scale deep-learning based models for sign language recognition. For example MS-ASL [5] contains more than $20,000$ sign samples of $1,000$ sign classes, while WLASL [1] features almost same number of sign samples with $2,00$ sign classes in total. Both works proposed several pose-based and RGB-based ML baselines for demonstrating the challenges and usefulness of the models. In both cases, 3D ConvNet based model was the best performing model. In one of our contributions, we show that the recognition accuracy can be improved using arm-hand related features for the sign video modeling task.

Chapter 3: Sign Language Recognition Analysis using Multimodal Data

Learning *spatio-temporal* dynamics, for a video dataset with large number of sign classes, requires enough training samples. This observation led us to build a dataset large enough to validate deep-learning based ideas for world level sign language recognition (WLSLR). In this chapter, we describe the proposed dataset GMU-ASL51 and related machine learning (ML) approaches along with experimental results. The work presented in this chapter has been published in the 2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Washington D.C., USA.

Most of the existing systems use RGB video data for American Sign Language (ASL) recognition task[6, 14, 72]. An ASL sign is performed by a combination of hand gestures, facial expressions and postures of the body. Sequential motion of specific body locations such as hand-tip, neck and arm provide informative cues about a sign. These location data can be obtained using commercial depth sensors like Microsoft Kinect and Intel Realsense. Kinect can use the depth information of a person to capture 3D coordinates of his/her body location across a video. This sequence of 3D body location is referred to as the skeletal data [43]. To the best of our knowledge, during this work, there was no publicly available skeletal dataset in literature for ASL recognition.

With skeletal data, an ASL sign can be seen as a sequence of 3D coordinates or a 3D time series [59]. Recurrent neural networks (RNN) are commonly used for modeling such sequential time series like data [32]. In this work, we investigated the impact of RGB video data in sign video recognition performance, especially when combined with the skeletal data. We also proposed a combined RNN network with a simple spatial data augmentation technique.

In summary, the contributions of this work are:

1. We proposed an RNN architecture with a novel spatial data augmentation technique;
2. We proposed an architecture that uses both the RGB and the skeletal data to improve recognition accuracy;
3. We introduced and publicly released a multi-modal dataset, titled as GMU-ASL51, for ASL word recognition.

3.1 The Proposed Dataset

ASL recognition with skeletal data has received little attention, resulting in a scarcity of public datasets. In the literature, there exists one public dataset for ASL recognition [7]. This dataset has 9,800 sign video samples from 6 subjects and more than 3,300 sign classes. The number of samples per class was small to train deep-learning based models. On the top of that, the samples were collected in a restricted setting, consisting of black attires and backgrounds. In contrast, our proposed GMU-ASL51 dataset has 13,107 samples from 51 word-level sign classes and 12 distinct subjects. These subjects are of different heights, builds and signing (using sign language) experiences. Figure 3.1 shows a T-SNE representation of a subset of samples from GMU-ASL51. The T-SNE was performed on the output vectors from a trained RNN model for each sign example in the subset. The used RNN model, AI-LSTM, is described in Section 3.2.2.

3.1.1 Collection Protocol

Figure 1.2 shows different components of our multi-modal data collection system. The RGB video data and the skeletal data were collected using a Microsoft Kinect depth camera, marked as `Kinect` in Figure 1.2. The camera was positioned in front of a human subject while performing a certain sign gesture. We trained each subject for a specific sign class, following online ASL tutoring websites, before collecting sign video samples for it. For each of the 51 sign classes, we collected 24 samples continuously. To gather individual samples

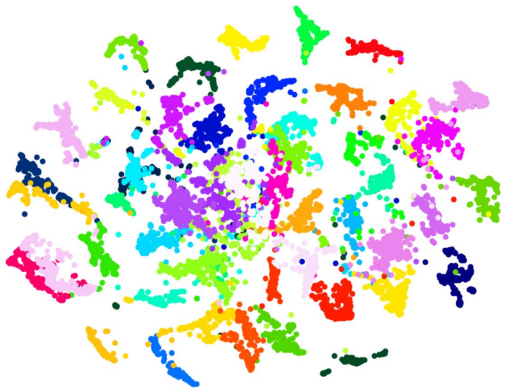


Figure 3.1: T-SNE representation of 11824 data samples from 51 different sign classes. Best viewed in color.

from the continuous data, segmentation marks were interleaved through a user interface. This was later used to segment individual samples. These samples were further segmented using motion measure of wrist joint co-ordinates for a video. For some subjects, we could not collect the samples for all the classes because of availability constraints. Consequently, we were able to collect 13,107 sign samples in total. We kept the camera distance between the subject and the camera in the range of 10 to 15 feet. We also assumed the subject is standing and front-facing towards the camera. For the lightning condition, we assumed an indoor lighting with occasional sunlight entering into the room. All of the above choices were made to imitate a house-like setup where a deaf individual can control a digital device with sign commands. Figure 3.2 illustrates, at the top, the distribution of the duration (frame counts) of videos in our dataset. At the bottom, the figure shows the distribution of the number of samples per gesture class in the GMU-ASL51 dataset.

3.1.2 Data Modality

All of our experiments on ASL recognition were done with the RGB video data and/or the skeletal data. Skeletal data is a multivariate time series input where each body key-point acts as a variable. Each of these key-points has 3D coordinate data at each time step or

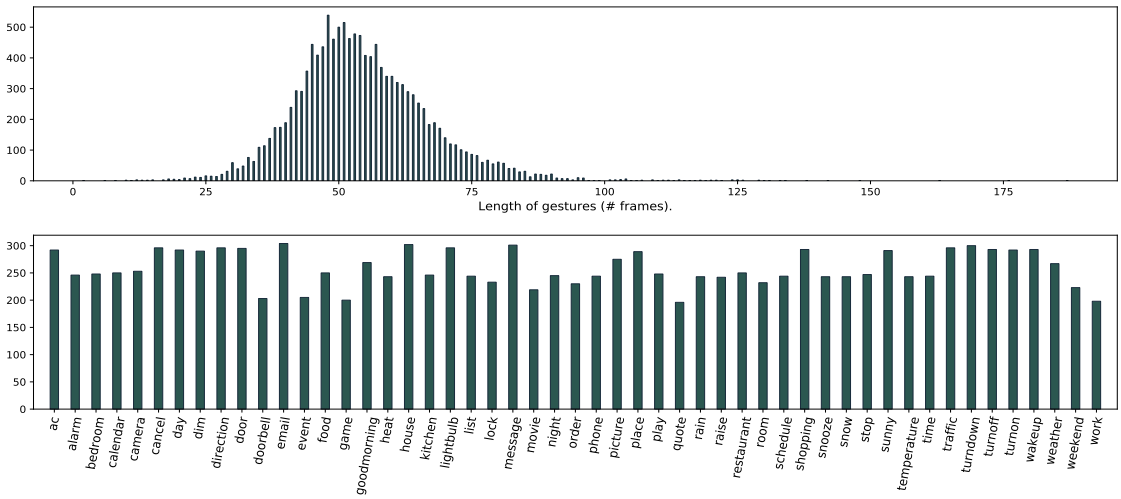


Figure 3.2: Top: Distribution of duration (frame count) of sign videos in the GMU-ASL51 dataset; Bottom: Count of sign videos from each of the sign classes (class label are best viewed zoomed in).

video frame. The skeletal data is also known as pose data and we use these two terms interchangeably. The skeletal data provides motion trajectory of different body parts such as wrist, elbow and shoulder (total 25 such body parts) over the video frames. This process is known as skeletal tracking. Skeletal data provides high-level motion of different body parts. These are useful for capturing motion features associated with different types of gestures. However, for better modeling of sign language, hand shapes are crucial, as different signs may have similar motion but different hand shapes and orientations. Figure 3.3 presents one such example where the sign pair `Alarm` and `Doorbell` have exact same motion pattern according to skeletal data but have different hand shapes. We observe similar situation for sign pairs such as `Kitchen`/`Room`, `Time`/`Movie`, `Quote`/`Camera`, `Lock`/`Stop` and many more. We hypothesize that the hand shape is useful in situations where skeletal data has similar dynamic motion patterns for different sign classes. Due to this fact, we extracted and used hand-shape features from RGB video data.

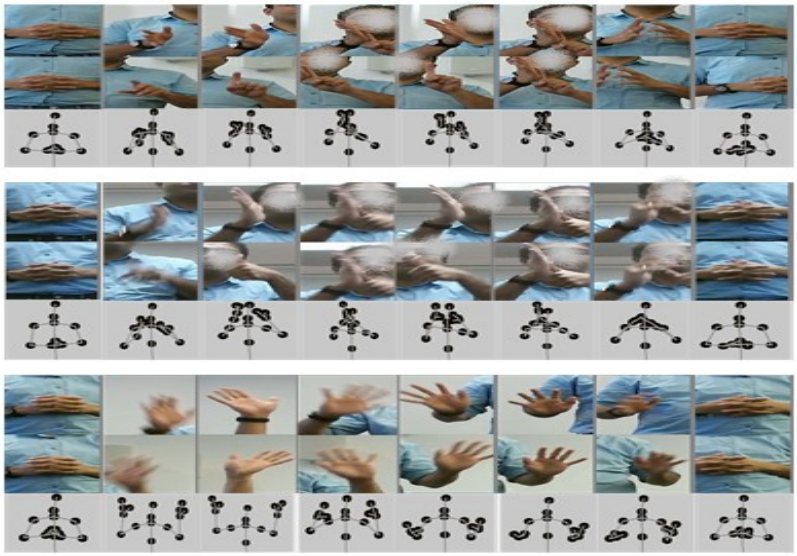


Figure 3.3: Visualization of hand shapes and skeletal joints of three sign classes. Top panel shows the sign `Alarm` and middle panel shows the sign `Doorbell`. For each sign, first two rows are the left and right hand image patches and third row is the skeletal configuration. We can see for `Alarm` and `Doorbell`, the skeletal motion is almost similar but each has different hand shapes. Bottom panel shows another sign `Weather` which has quite distinguishable skeletal motion from top two.

3.2 Our Approach

Inspired by the success of deep-learning (DL) based approaches in computer vision [85], we applied different DL architectures to model sign languages from the RGB and the skeletal input. In traditional image classification or object detection models, neural networks learn hierarchical spatial features from data. However, sign video recognition requires temporal context learning on the top of spatial feature learning from each video frame. To model the temporal context, we based our implementation on Recurrent Neural Network (RNN).

3.2.1 RNN Based Modeling

As described in section 2.2.1, the RNNs are commonly used for capturing temporal dynamics in data. The skeletal or body pose data, obtained from Kinect sensor, represents a sign video using a sequence of 3D coordinates. Each item in this sequence refers to the pose

representation of a human body in the corresponding video frame. As seen from the skeletal representation in Figure 3.3, these pose sequences are readily available as input to the RNN networks. Based on this observation, we propose two LSTM networks for modeling the sign videos using body pose data. The LSTM network is an upgraded version of the RNN network. The working principle of the LSTM netowrks is described in Section 2.2.1 in details.

3.2.2 Axis Independent LSTM

Given a sample skeletal data of $R^{T\times J\times 3}$, where T denotes time axis, J is the number of body joints and the last dimension is the 3D coordinates of each joint. We flatten all the dimension except the time and at each time step we can feed a vector of size $R^{3\times J}$ as input. However, we have empirically verified that learning a sequential pattern for each coordinate axis independently and combining them later shows stronger classification performance. One possible reason is noisy skeletal data. Often, the noise exists in a particular axis and the trajectories along other axes are good. Hence, decoupling the modeling across coordinate axes helps to represent the sign video using available noise free axes. Based on this, we train three different 2-layer LSTMs for data from x, y, and z coordinates separately; and concatenate their final embedding to produce a softmax output. In this setting, each separate LSTM receives data as $R^{T\times J}$ and final embedding size is $R^{3\times S}$ where S is the state size of LSTM cell. Figure 3.4 (a) shows the architecture where as a sample arrives, just before entering into the main network, data along each separate axis is split and enters into three different LSTM networks. The model concatenates the final state from each of the separate LSTM networks. The concatenated state is fed into the softmax layer for generating classification scores. This approach is referred by Axis Independent Architecture (AI-LSTM). Implementation details such as values of T and J are provided in the 'Experiments' section.

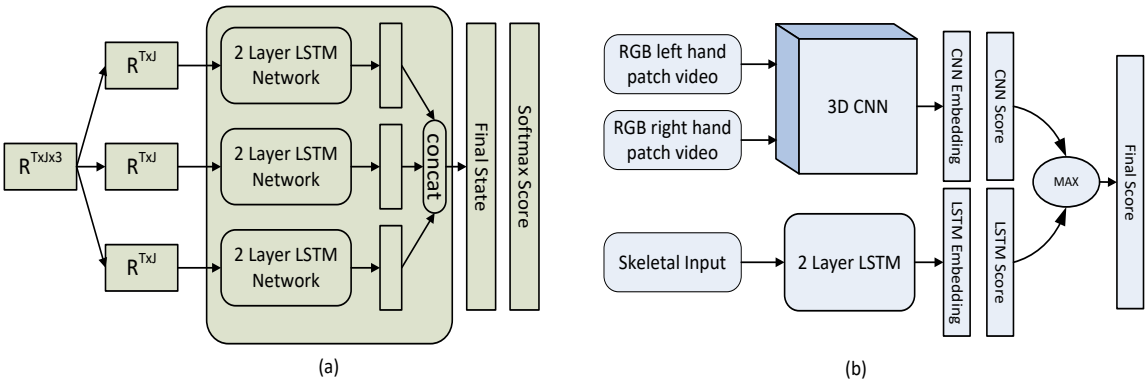


Figure 3.4: Proposed architectures. Fig (a): Axis independent LSTM network where data from each axis enters into different LSTM networks and at the end we take the concatenation of individual states. Fig (b): Combined architecture. Here 3D CNN symbolizes the architecture we presented in Figure 3.6. Here both CNN and LSTM network model data separately. At the end we take the maximum of probability scores produced by both network.

3.2.3 Spatial AI-LSTM

The AI-LSTM works by modeling temporal dynamics of body keypoint data over time. However, spatial interaction among the joints, at a specific time step, might exist. For example, for some sign classes, there might be a diagonal movement of right wrist with respect to the neck joints, while for some other classes, there is vertical or horizontal wrist movements. The AI-LSTM fails to capture any such interaction among joints in a given time. To incorporate the spatial relationship among the body joints, we propose a simple novel data augmentation technique for skeletal data. We do this by transferring the origin of the body skeleton. For each frame in a sign video, we use each wrist joints as origin and transform all other joints' data by subtracting that origin from them. In this way, the input to the LSTM become spatially aware. We refer to this model as Spatial AI-LSTM. This augmentation technique is depicted in Figure 3.5. A sample data of form $R^{T\times6\times3}$ results in a representation of $R^{T\times5\times3}$ after transferring the origin to the left wrist joint. After this

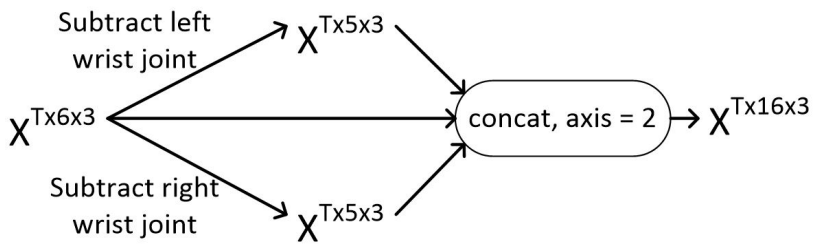


Figure 3.5: Spatial data augmentation.

augmentation process, each sample is a $R^{T\times16\times3}$ sized tensor. Hence, each separate LSTM networks in our Spatial AI-LSTM network receives an input of $R^{T\times16}$.

3.2.4 3D Convolution Based Modeling

Recalling from Section 2.1.2, the 3D convolution can tie the temporal context in a sign video. It achieve this by sliding 3D convolutional kernels in the *spatial-temporal* 3D space of a video. Each of these kernels learns rich *spatial-temporal* features. These features are purified layer by layer and in the end, a vector representation of the input video is generated. To utilize the modeling capacity of 3D CNN, we proposed to use it for sign video recognition. For our specific task, we proposed to use 3D CNN on the sequence of cropped hand patches. An example implementation is shown in Figure 3.6. This architecture has two parts: one for left hand patches and other for right hand patches. Each part has four 3D convolutional layers (second and fourth layers have following maximum pooling layers) followed by 2 fully connected layers. Final embedding from these two parts were concatenated and a classification score was produced using a softmax layer.

3.2.5 Combined Network

We hypothesize that some signs that have mostly similar skeletal motion pattern could be distinguishable using hand shape information. We proposed a combination of LSTM and 3D CNN networks. We call this Max CNN-LSTM network. Figure 3.4 (b) shows the Max CNN-LSTM. The details of the 3D CNN module is shown in Figure 3.6. This module produces

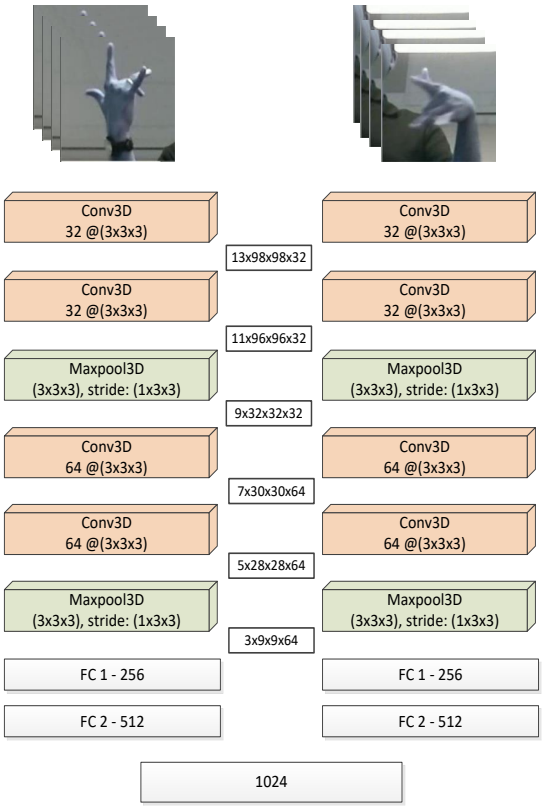


Figure 3.6: Used 3D CNN architecture for this work. It consists of four 3D convolutional layers and two fully connected layers at the end. There are two separate networks for left and right hands. Final embedding of these two networks are concatenated before producing softmax score. Feature map dimensions after each layer are shown in the middle.

one set of classification scores. The other AI-LSTM network, shown in Figure 4.3 (a), is fed with skeletal time series data. At the final time step, the LSTM state vector is taken and using a softmax layer another set of classification score is produced. The final classification score, as shown in Figure 4.3 (b), is calculated by taking element wise maximum of the output scores from the two networks. During the model parameter learning, using back–propagation, both networks are trained on their own score. The combined network acts like a model ensemble and some sign classes which are confused by the RNN network alone might have an improved recognition accuracy with this approach.

3.3 Experiments

A video sign differs in the execution speed and style in the same way as a spoken sentence differs across different individuals. Hence, each video sample from our dataset can have different frame lengths because of the style and speed variation among participating subjects. A frame length distribution is shown in Figure 3.2. Even a subject may execute the same sign at different speeds at different times. This makes the sign recognition problem more challenging. Further, neighboring frames in a video contain redundant information; and all joints will not have the equal amount of motion pattern in case of skeletal data. Hence, a pre-processing is necessary before using data for training the models. In this section, we describe the pre-processing mechanisms that were used in out experiments.

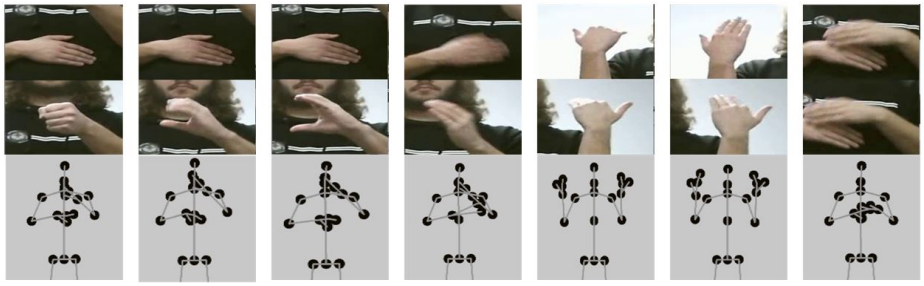


Figure 3.7: Seven sampled frames from a sign of class Air Condition. Top two panels show cropped hand patches while bottom panel shows body skeletal configuration of corresponding frames.

3.3.1 Skeletal Data Preparation

A sign video does not involve all the 25 joints' information provided by Kinect sensor. Mostly upper body joints are useful, especially joints involved with the two hands convey most information. Based on this, we consider only 6 joints – wrist, elbow and shoulder of both hands – as input to the LSTM network. Figure 3.7 shows an example where 7 frames were sampled from a sign video of class `Air Condition` and the bottom panel shows the

skeletal configuration across those 7 frames. From each sign video we sampled 20 number of frames uniformly and took joints' data associated with those frames. We took the sample frame rate as a hyper-parameter of the model and verified experimentally that picking 20 frames uniformly works best for skeletal data. For samples with less than 20 samples we converted them to 20 frame signs by interleaving existing frames uniformly. Thus skeletal data for each sample is of the form $R^{20\times6\times3}$.

3.3.2 Video Data Preparation

Since ASL involves specific hand shape patterns, we cropped both hand regions in each image frame. Using 2D coordinates of hand joints on a video frame as center, we did a 100×100 crop to generate hand crops. To reduce motion blur, we calculated velocity of joints at each video frame using skeletal coordinates and then sampled from frames which have less motion. We sampled 15 frames from each sign video resulting in a vector of $R^{15\times100\times100\times3}$ for each hand patch.

3.3.3 Training Details

To deal with the over-fitting problem, dropout was used for all networks, except convolutional layers with probability of 0.5. In addition to the dropout, L2 regularization was used for LSTM networks and for dense layers; β was set to 0.008 which controls the impact of regularization on the network. State size and number of layers of LSTM networks were 50 and 2, respectively. Learning rate for Max CNN-LSTM and LSTM networks were set to 0.00001 and 0.00005, respectively. We used Adam Optimizer for training our networks [86]. All networks were run for 300 epochs with a batch size of 64. We developed all of our models with Tensorflow 1.10 (python). Average time taken to train an AI-LSTM and an Spatial AI-LSTM were 25 and 30 minutes on an Intel(R) Core(TM) i5-7600 (3.50GHz) processor respectively. We trained 3D CNN and Max 3D CNN models on GPU (Tesla K80) and each model took around 20 hours to train.

3.3.4 Baseline Methods

To compare our proposed method with baseline approaches we implemented two algorithms: Support Vector Machine (SVM) and Random Forest (RF). To prepare the features from the skeletal data, we utilized following measures: Mean, Area, Skew, Kurtosis, Motion Energy, Range and Variance across the video frames. We have 6 upper body joints, 3 axes per joint and 7 aforementioned features for each sign video. This yielded a total of 126 ($7 \times 6 \times 3$) features per input sign video.

Table 3.1: Average cross-subject (CS) accuracy across all test subjects for different proposed architectures and baselines. Standard deviation across test subjects' accuracy is also shown.

Methods	Accuracy (CS)	Std. Deviation
SVM	62%	10%
Random Forest	66%	8%
3D CNN	52%	12%
AI-LSTM	73%	6%
Max CNN-LSTM	**75%**	7%
Spatial AI-LSTM	**81%**	6%

3.3.5 Experimental Results

Table 3.1 shows the comparative results among our proposed architectures and baselines. Overall, we used data from 12 subjects for our experiments which sum up to 13,107 sign gesture samples in total. To evaluate model performance on a specific subject, we adopted cross-subject evaluation criteria. Suppose, X is the test subject. We trained our networks with all sign samples except those are from subject X. We used subject X's data as test split to evaluate the performance of the networks. Table 3.1 shows the average test accuracy for all 12 subjects. We can see that 3D CNN network alone performs worse than simpler skeleton based baselines. But when coupled with AI-LSTM as Max CNN-LSTM, it shows an increase in recognition accuracy by 2% from AI-LSTM alone. This is because some of the

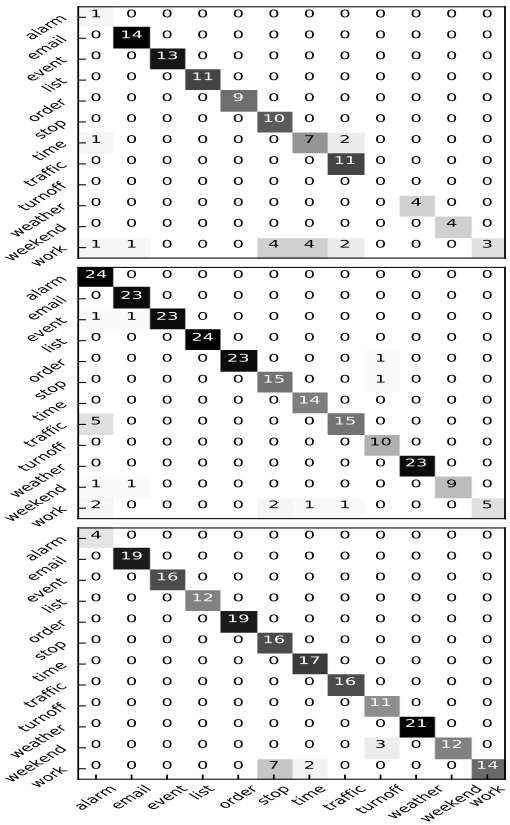


Figure 3.8: Confusion matrix for a subset of sign classes from a subject for AI-LSTM, Max CNN-LSTM and Spatial AI-LSTM from top to bottom respectively. Mentioned signs are a subset of 51 sign classes.

signs were confused by the AI-LSTM network because of similar skeletal motion pattern. Incorporating spatial relationship among joints led to a 2% improvement. The Spatial AI-LSTM was trained only on skeletal data but outperformed the combined network by 6%. Figure 3.8 shows three confusion matrices for a subset of 12 sign classes for a subject. The top matrix is for AI-LSTM network, middle is for Max CNN-LSTM and bottom one is for Spatial AI-LSTM. As seen from the Figure 3.3, the sign pairs `Alarm`/`Doorbell` are similar in skeletal motion but have different hand shapes. Since Max CNN-LSTM includes hand shapes, it can successfully recognize it while other two models struggles. Same is true for some other signs like `Email`, `Event`, `List`, `Order` and `Weather` . Some other signs are

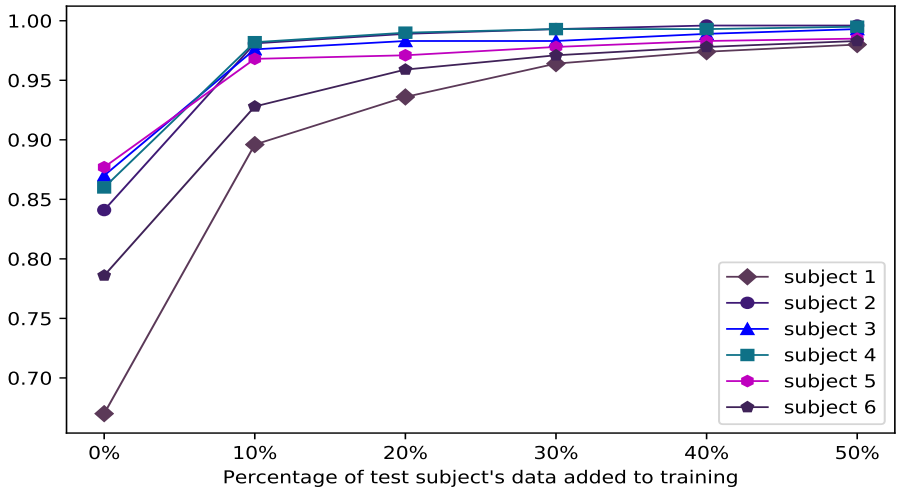


Figure 3.9: Effect of adding data to the training from test subject in Spatial AI-LSTM model. X axis is the fraction of test subject's data used in training. Y axis is the test accuracy.

better recognized by Spatial AI-LSTM network. It should be mentioned here that accuracy listed in Table 3.1 shows average accuracy across all test subjects, while Figure 3.8 presents confusion matrix for a single test subject. For this particular subject, overall test accuracies were 58%, 70% and 69% for AI-LSTM, Max CNN-LSTM and Spatial AI-LSTM network respectively.

3.3.6 Effect of Same Subject Data in Training

Cross-subject evaluation criteria, mentioned in Section 3.3.5, assumes absent of training data from the test subject. We want to know the impact of adding videos, to the training process, from a test subject. Usually, if a model sees video samples from the test subject during the training process, the test recognition accuracy should be improved. However, we want to know how much or what fraction of data is necessary for achieving a higher recognition accuracy, say greater or equal to 90%. This is important for assessing the practical usability of a recognition system. In other words, we want to know how quickly or with what amount of data, the current system can be adapted for a previously unseen

subject. To do that, we first pick a test subject and train a model for that subject using the data from all other subjects in our dataset. Then we retrain the model with some fraction of data from the test subject. We keep increasing the fraction of data being used from the test subject in the retraining process up to 50%. The other half of the test subject's data is used for testing the model. Figure 3.9 shows the effect of added training data from the test subject in the retraining process, started with 0% training data. We observe that adding data from a test subject increases recognition accuracy and similar improvement is observed for all the 6 subjects. It is interesting to observe that, adding even 10% of data from a test subject yields more than 90% recognition accuracy for all the 6 subjects shown in the Figure 3.9.

3.4 Key Takeaway

We presented deep-learning based approaches for ASL word sign recognition that use skeletal and video data as input. The best performing model captures the underlying temporal dynamics and identifies specific hand-shape patterns from video data to improve recognition performance. A new data augmentation technique was introduced that allowed the LSTM networks to capture the spatial dynamics among joints. Finally, a large public dataset for ASL recognition has been developed and released for public research. We anticipate the dataset will be helpful for advancing ASL recognition research.

Chapter 4: Body Pose and Deep Hand Shape Feature Based American Sign Language Recognition

In this chapter we investigate the effectiveness of frame-level hand shapes in the context of sign gesture modeling. A traditional word-level sign language recognition (WLSLR) system uses only video-level sign annotation to learn *spatial-temporal* dynamics in a video. We hypothesize that, using frame-level shape representation can help boosting recognition accuracy. To validate our initial thoughts, we proposed to utilize hand-shape features from a pre-trained CNN model. This pre-trained CNN model was trained on 1 million hand-shape images from a related domain. Our experimental results showed 8% improvement in the recognition accuracy using such hand-shape features. The work presented in this chapter has been published in the 2020 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Sydney, Australia.

American Sign Language (ASL) uses approximately 6,000 gestures for common words and finger-spelling for more obscure words and proper nouns [87]. An ASL gestures is performed by a combination of hand gestures, facial expressions and postures of the body. Motion of specific body locations (such as hand-tip, neck and arm) provide informative cues about a sign. Early ASL recognition methods considered video [88] and sentence level parsing with carefully engineered environments to simplify the hand detection and tracking using traditional image features. The followup advancements included creation of larger datasets [7] with uniform background and multiple views. More recent advances include use of RGB-D sensors as well as robust feature learning methods, based on Deep Convolutional Neural Networks (CNNs) [6, 27, 72]. These approaches consider smaller datasets (e.g. spelling gestures), uniform backgrounds and with the exceptions of few consider only

subject-specific models and evaluations. In order to overcome these challenges, we proposed to use the skeletal data to aid the ASL gesture recognition. Skeletal data provides multi-variate time series 2D of 3D coordinates of body joints, factors out the appearance of the speaker and naturally enables speaker-independent model for ASL recognition. The basic skeletal model will be augmented by hand-shape representation obtained from another hand-shape image recognition CNN model, trained on hand-shape patterns from a different source [12]. We leveraged this pre-trained CNN model to overcome the problem of learning frame-wise hand representation. In addition to that, for obtaining reliable human body pose estimates, we used recent advances in deep-learning based pose estimation from video [23, 24, 50], and compared performances with depth sensor based poses.

Given skeletal trajectories of body joints and extracted hand-shape features for all the frames in a sign video, we trained recurrent neural network (RNN) [32] to model sign dynamics and predict the sign gesture label. In summary, the contributions of this work are:

1. We proposed a novel RNN architecture for modelling sign videos, using the pose data and the hand-shape features extracted from pre-trained models;

2. We proposed a fusion method with two jointly trained RNNs that can learn prioritizing among different input sources;

3. We evaluated similar network architectures for the pose data obtained from a depth sensor and the pose extracted from RGB videos, to verify that competitive results can be achieved without the depth sensor.

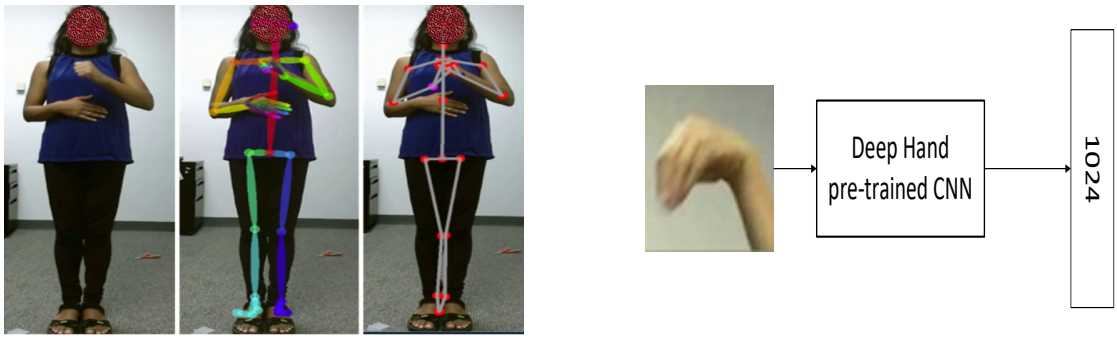
We performed all of our experiments using the GMU-ASL51 benchmark dataset [10]. We also performed an experiment to identify the subjects using two types of input data: the RGB and the skeletal pose data. Our experiments showed that the RGB input, for some subjects, yields 15% better performance than the skeletal pose input for the user identification problem.

4.1 Our Approach

In this section, we describe the methods for estimating pose of the upper body and hand-shape features from RGB videos. The time series of skeletal data and the hand-shape features will then be used to train a model for sign video recognition.

4.1.1 Pose Estimation

Pose estimation is the process of estimating 2D or 3D body joint locations (e.g. wrist, elbow) in single image [21–23, 47–50]. We discussed the pose estimation process in Section 2.3. For this work, we used the state-of-the-art 2D human body pose estimation approach: the OpenPose [23]. We used this model because, in addition to the body joints, it provides joint locations of the palm and the fingers. Figure 4.1 (a) shows an image frame from an



(a) (b)

Figure 4.1: Pose estimation examples (a) and hand-patch generation (b).

ASL sign gesture (left), corresponding estimated poses using OpenPose (middle) and pose generated by Kinect depth sensor [43] (right). In the rightmost frame of Figure 4.1 (a), we see that Kinect API tracks left hand joints instead of right hand by mistake while the OpenPose estimation, shown in the middle frame, is better. Hence, we observe that, the OpenPose not only gives fine grained finger joint locations but its estimation is also less

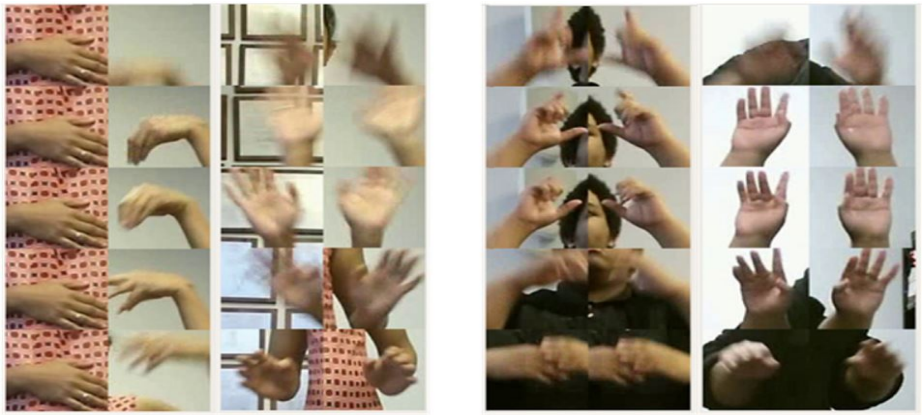


Figure 4.2: Illustration of motion blur. Leftmost two figures show sign 'sunny' and 'weather' performed by one subject. Rightmost two column show sign 'camera' and 'snow' performed by another subject. Each column represents hand patches of both hands from five sampled frames of the sign video.

noisy than the Kinect. This makes the OpenPose an excellent choice for pose estimation for sign language modeling.

4.1.2 Hand Feature Extraction

We cropped a rectangular patch around hands centered at the mean (x, y) location of the finger and palm joints obtained from pose estimation. These image crops often contain severe motion-blur because of the hand motion associated with the sign execution, as shown in Figure 4.2. In order to learn effective representations of hand shapes while accounting for these complex variations in hand appearances, large amount of training data is necessary. In addition to the data, frame-level hand annotation is also required to train supervised machine learning methods. Obtaining such amount of labeled training data is a cumbersome process. An alternative is to use models that are already trained on data from a related distribution. Recently, several deep learning models for hand segmentation, recognition and detection were proposed [12,73,74]. Most are however not suitable for our problem because of different factors such as nature of viewpoint and hand pose variation in these datasets. For example, [73] has egocentric – first person vision – viewpoint while [74] considers data

from online sources such as movie clips. We observe that 'Deep Hand' model is the most suitable model for our problem [12].

Deep Hand Model Koller *et al.* [12] trained a 22 layer deep convolutional neural network (CNN) with more than 1 million images, using videos from Danish and New Zealand sign language. The data is weakly labeled with only video level annotation. The CNN model for estimating the likelihood of hand shapes, was trained using Expectation Maximization (EM) algorithm, jointly with Hidden Markov Model (HMM) for parsing sign gestures. The network was trained to recognize 60 hand shape classes plus one additional class which determines the start or end of a video. We omitted the softmax classification layer of this network and used the embeddings computed by this pre-trained model as hand-crop patch representations. Figure 4.1 (b) shows a schematic diagram of the process of hand-shape representation generation. The representations obtained from the DeepHand model were used as input to our proposed RNN based sign language methods.

4.1.3 Recurrent Neural Network

For modeling sequential data, recurrent neural network (RNN) have yielded impressive results on a variety of sequence prediction tasks [32]. Vanilla RNN suffers from vanishing gradient problem which was solved by introducing Long Short Term Memory (LSTM) [35]. While the basic RNN is a direct transformation of the previous hidden state and the current input, the LSTM maintains an additional internal memory and has a mechanism to update and use that memory. This is achieved by deploying four separate neural networks also called gates. Section 2.2.1 details on this topic.

4.1.4 Proposed Architectures

Assume that we have an ASL sign video data $R^{F\times H\times W}$ where F, H, W are number of frames, height and width respectively. Using 2D locations of hand joints for each frame, we crop a rectangular window for each of the hands. Suppose, the dimension of our cropped

hand patches is 100×100. Hence, by cropping hand regions from all the frames, we convert our video data to $R^{F \times 2 \times 100 \times 100}$ where second dimension represents the two hands. As discussed in the Section 4.1.2, the pre-trained Deep Hand model produces an R^{1024} for each hand patch image. Thus, by feeding our cropped data for each frame in the video into this model we produce two embeddings e_1 and e_2, which are $R^{F \times 2 \times 1024}$, which is $R^{F \times 2048}$ after flattening last two dimensions.

For body joint locations (skeletal data), a video can be seen as $R^{F \times J \times D}$ where F, J, D are frame counts, joint counts and coordinate dimension respectively. In this setting, D can be 2 or 3 dimensional. In our case, skeletal data directly obtained from Kinect depth sensor [43] are 3D, while joint locations estimated from pose estimation algorithm are 2D. The extra dimension in case of depth sensor data is the depth information of the body joints. This depth refers to the distance between a joint and the camera origin. After flattening the last two dimensions, the joint location data becomes $R^{F \times (J \times D)}$. Hence, for each sign video we have hand-shape feature data of form $R^{F \times 2048}$ and skeletal data of form $R^{F \times (J \times D)}$. Here, frame numbers F varies across videos. Also, consecutive frames in a video contain similar information for a standard sample rate like 32 frame per seconds. Thus, it is usual to sample a fixed number of frames from each video. We first pick a sample rate say T and then divide the whole video into T consecutive windows. Then from each window we pick one frame randomly. Thus, a video data like $R^{F \times 2048}$ becomes $R^{T \times 2048}$ after sampling. Similarly, skeletal data $R^{F \times (J \times D)}$ becomes $R^{T \times (J \times D)}$. Appropriate T can be set as a hyper-parameter. We will provide details in the experiments section.

Fusion LSTM Figure 4.3 shows our proposed fusion architecture. We see the top LSTM network takes hand shape feature input. We call this RgbLSTM since its input is generated from RGB hand patches. Bottom LSTM takes body joint pose locations as inputs and we call this PoseLSTM. Final embedding from both layers are fused using concatenation and passed through a softmax layer for final prediction score. We call the whole network as FusionLSTM.

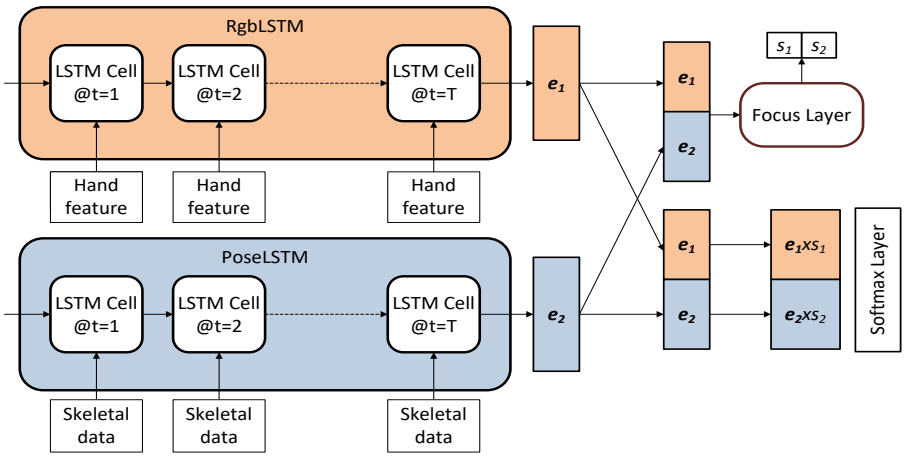


Figure 4.3: Overall architecture. Top is the LSTM network on RGB hand-shape feature while bottom is the LSTM on skeletal joint data. Final embedding vector from these networks are fused by concatenation. There is also a Focus Layer which learns to prioritize between the two feature representations depending on the data and annotation.

Plain concatenation during fusion works well for our proposed architecture. However, we found in the experiment that for some subjects RgbLSTM gives better performance while for other subjects PoseLSTM yields superior results. To get maximum benefit from both the networks, we proposed an attention mechanism that assigns priority to either one of the networks. The idea is to first feed the plain concatenated output from both networks to a neural network layer which we call 'Focus Layer' in Figure 4.3. Focus layer produced two scalar values s_1 and s_2. These scalars were used for scaling the final embedding vector from both networks e_1 and e_2. Embedding vector e_1 was multiplied with scalar s_1 and vector e_2 was multiplied with s_2. Then those multiplied embeddings were concatenated and a prediction score was produced as shown in Figure 4.3. We will show in experiment that, this mechanism improved overall recognition accuracy by 2% to 3%. Interestingly, for some subjects it yielded significant improvement, specifically when one of the sources was noisy.

4.2 Experiments

In this section first we briefly present GMU-ASL51 [10], the dataset we have used for all of the experiments shown in this work. We then present experiment details of our proposed architectures and all baseline methods.

4.2.1 Dataset

We used GMU-ASL51 for all of our proposed and baseline experiments. GMU-ASL51 has 51 word-level ASL signs performed by 12 subjects of different ages, gender and builds. Among the participating subjects 8 were male and 4 were female. Ages of the subjects range from 20 to 35 years. The dataset was collected using depth sensor and has two modalities: RGB videos and 3D skeletal body parts. Figure 3.2 shows the histogram of number of gesture samples per class and the histogram of the duration of videos in the GMU-ASL51 dataset. Details can be found in Section 3.1.

4.2.2 Baseline Methods

Convolutional Neural Network (CNN) with 3D convolutional kernel (3D CNN) has shown promising performance in classifying human activities in video [14]. That's why we chose 3D CNN on RGB hand patch videos for one of our baselines. It consists of four 3D convolutional layers and two fully connected layers at the end. There are two separate networks for left and right hands' patches. Final embedding of these two networks are concatenated before producing softmax score. We have also implemented other feature based baselines such as random forest and support vector machine using only 3D skeletal data. These baseline models utilize skeletal data in each axis for every joint to create following features per sample: Mean, Area, Skew, Kurtosis, Motion Energy, Range and Variance over the frames. We have 6 joints and 3 axes per joint and 7 features for each one of them giving us a total of 126 ($7 \times 6 \times 3$) features per sample [89].

4.2.3 Experiment Using Kinect Pose

For this experiment we used RGB video and 3D skeletal pose data as computed by Kinect sensor. We performed four types of experiments using these two types of data. Two of those experiments used each modality separately. The other two used simple concatenated fusion or designated Focus Layer as described in the section 4.1.4.

Most of the ASL signs do not involve all the 25 joints' information provided by Kinect sensor. The joints belong to the hands carry most of the information regarding sign language. Given that, we consider only 6 joints (wrist, elbow, shoulder) from both hands and use them as input to the LSTM network. In this way, for each video, our skeletal data of form $R^{T\times(J\times D)}$ becomes $R^{20\times(6\times 3)}$ with ($J = 6$ and $D = 3$). We use $T = 20$ for sampling rate for skeletal data. To crop hand patches from RGB video frames we use hand joint's 2D location and select a window size of 100×100. After cropping we feed this into Deep Hand model and prepare hand feature data as $R^{15\times 2048}$ as described in the section 4.1.4. Sampling rate used for hand data is $T = 15$ for each video.

4.2.4 Experiment Using RGB Estimated Pose

Data preparation using estimated pose is similar to what we have already discussed in the previous section. However, there are some differences. First, our pose data is of form $R^{20\times(48\times 2)}$ because we are using 48 joints and our pose is 2D in this case. Among 48 joints 6 are wrist, elbow and shoulder from both hands; the remaining 42 joints are finger and palm level joint location from both hands.

4.2.5 Comparative Methods

Finding a similar work for comparison with our method was difficult due to the scarcity of public ASL datasets. A closely related and well studied setup is generic isolated gesture recognition [52, 90–92]. In [52] authors propose FOANet gesture recognition architecture, that uses fusion of different channels on cropped hand patches and full body. Using different modalities (RGB, depth and flow) with those channels, this work sparsely fuses 12 channels

of inputs to model gestures. We tried to reproduce this work on our dataset as closely as possible. However, we only used 2 channels (RGB left/right hand patch) instead of full 12 channels to make a fair comparison with our experiments. In addition to that, there were some changes in terms of input data such as the crop size and the location feature. The FOANet uses a hand detection network to generate cropped hand patches which gives hand crops of different sizes. On the top of the RGB hand patch input, they use sizes of patches as location features. But our hand patches, cropped using 2D pose information of hand, are of the same size. Hence, we could not use location features from the sizes of the hand patches. However, we used corner coordinates of hand windows as location features.

4.2.6 Training Details

In this section, we describe implementation and hyperparameter details of presented models. We observe from experimental results that different T (sample rate) values for RGB features and pose data works well in terms of test accuracy. We used $T = 15$ for RGB hand feature data and $T = 20$ for pose data as described in section 4.2.4 and 4.2.3. We used Adam Optimizer for training our networks [86]. Initial learning rate was set to 0.00005. 2-Layer LSTM networks were used for all implementations. State sizes used for PoseLSTM and RgbLSTM were 100 and 1000 respectively. Size of the hidden layer in focus layer was 512. To deal with the over-fitting, dropout was used for all LSTM networks with probability 0.5. In addition to dropout, L2 regularization was used for dense layers; β was set to 0.008 which controls the effect of regularization on the network. Average cross entropy loss was used to update the network parameters for a batch. Given a true one hot encoded class label of $y_{i,j}$ and corresponding predicted score of $\hat{y}_{i,j}$ this loss is calculated as Equation 4.1.

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{i,j}\log\left(\hat{y}_{i,j}\right) \tag{4.1}$$

Table 4.1: Test accuracies across 12 subjects. In header row each subject is represented by S appended with subject number. Top three rows show results of our baselines. Next four rows show experiments with 3D pose and extracted hand shape features. Bottom four rows show results of using estimated pose and hand shape features.

		S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	S11	S12	Avg.
	3D CNN	0.62	0.63	0.60	0.60	0.56	0.56	0.62	0.45	0.39	0.51	0.46	0.17	0.51
	Random Forest	0.60	0.67	0.43	0.68	0.66	0.64	0.62	0.59	0.58	0.76	0.78	0.49	0.62
	SVM	0.61	0.75	0.46	0.73	0.68	0.73	0.69	0.59	0.66	0.76	0.67	0.61	0.66
	FoaNet[52] Style	0.18	0.22	0.094	0.19	0.11	0.28	0.15	0.17	0.09	0.12	0.11	0.05	0.15
3D Pose	PoseLSTM	0.81	0.88	0.68	0.84	0.88	0.85	0.85	0.78	0.81	0.83	0.76	**0.83**	0.82
3D Pose	RgbLSTM	0.81	0.82	0.85	0.89	0.81	0.89	0.93	0.69	0.83	0.72	0.81	0.37	0.78
3D Pose	FusionLSTM	0.90	0.93	0.88	0.93	0.90	**0.95**	**0.96**	0.84	0.91	**0.89**	**0.89**	0.67	0.88
3D Pose	FusionLSTM (Focus)	**0.92**	**0.93**	**0.88**	**0.94**	**0.90**	0.94	0.95	**0.85**	0.90	0.84	0.89	0.77	**0.89**
2D Pose	PoseLSTM	0.83	0.90	0.89	0.92	**0.90**	0.92	0.95	**0.88**	**0.94**	**0.93**	0.94	**0.63**	0.89
2D Pose	RgbLSTM	0.80	0.82	0.88	0.93	0.82	0.92	0.95	0.79	0.90	0.75	0.92	0.27	0.81
2D Pose	FusionLSTM	0.85	0.85	0.93	**0.95**	0.87	0.92	0.98	0.83	0.93	0.85	0.96	0.40	0.86
2D Pose	FusionLSTM (Focus)	**0.87**	**0.92**	**0.93**	0.93	0.88	**0.95**	**0.98**	0.84	0.93	0.91	**0.95**	0.56	**0.89**

Here, N is the size of the minibatch and C is the number of classes. For our experiments, these values are 64 and 51 respectively.

4.2.7 Results

Table 4.1 shows our experimental results. We evaluated each method in a cross-subject manner. For a particular test subject, we trained our model using data from all other subjects in the dataset. This cross-subject evaluation criteria mimics the practical scenario where the trained system is expected to be working on previously unseen subjects. Each column in Table 4.1 shows test accuracy of one subject in our dataset. First four rows shows the baseline results: 3D CNN, Random Forest, SVM and FoaNet style results respectively. Next four rows presents experiments with 3D Kinect pose and bottom four show experiments with estimated 2D pose. Hand shape features are common to both type of experiments (2D/3D pose).

Result shows that methods using 2D poses achieve similar performance to Kinect poses which is interesting because 2D pose methods depend only on RGB data. Also we observe

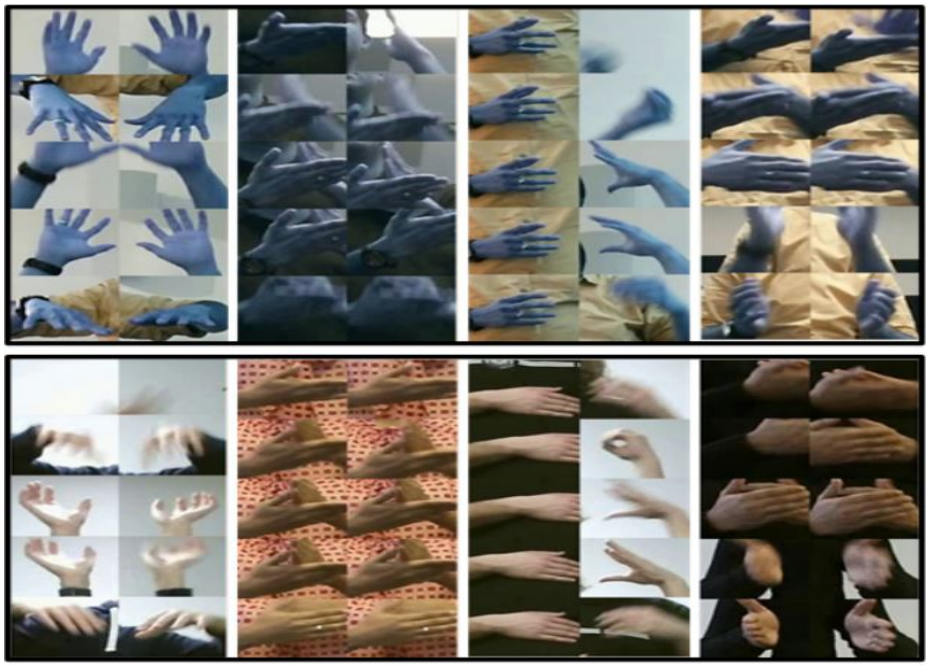


Figure 4.4: Illustration of different RGB distribution for subject 12. Top hand patches show four signs from S12 ('rain','stop','sunny','kitchen' in order). Bottom patches are examples of same corresponding signs from four other subjects from our dataset. This clearly shows that S12 has different bluish RGB color than all other subjects which explains bad results with RGB modality for S12 (difference in RGB is best viewed in color).

that, using fusion of both modalities increases the model performance to some extent (7% in case of 3D pose). The proposed focus layer improves the accuracy by 1% in 3D pose and by 3% in 2D pose. However, if we take a close look at the result of subject 12 (S12), we observe the actual benefits of the focus layer. Video data for S12 is different from other subjects due to a compression related color changes during data acquisition which is apparent by the poor recognition accuracy (RGB modality). Figure 4.4 shows how RGB data from S12 has different distribution than other subjects in the dataset. However, Kinect pose (3D Pose) data does not depend on RGB and we can see we have 83% test accuracy in PoseLSTM for S12 just as like other subjects. With focus layer which was proposed to learn priority among modalities, we can see that there is a 10% gain in accuracy for S12. In case of 2D pose, PoseLSTM has 20% lower performance than 3D pose version. This large

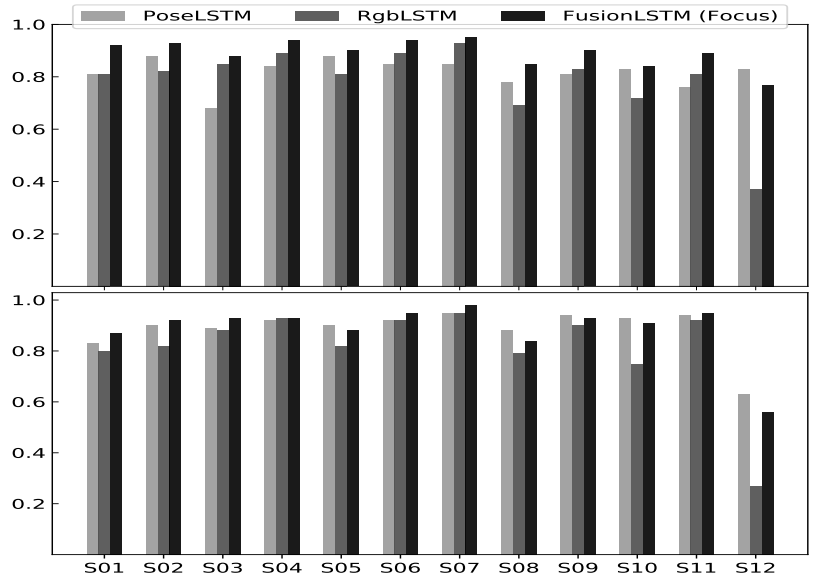


Figure 4.5: Test accuracy bar charts of different subjects(X-axis) in our dataset. Top figure shows the results using Kinect 3D poses while bottom shows results using estimated 2D poses.

decrease in accuracy is uncommon in other subjects because in all other cases PoseLSTM for 2D pose gives better accuracy than 3D pose (because of finger joints). Since 2D pose is estimated from RGB data which has different distribution for S12, we speculate that, this bad performance is expected. We have verified this by observing hand poses estimated by OpenPose model for S12. These poses are noisy and the model outputs very low confidence for S12's finger joints. However, we see a 14% increase in the accuracy in case of focus layer version using 2D pose for S12. This result supports that our proposed attention mechanism can significantly improve recognition performance in case of a noisy (or bad) source. Figure 4.5 shows the relative test accuracy of three implementations for using each of two types of pose data: 3D pose from depth sensor (top) and estimated 2D pose from RGB (bottom). We can see from the plots that, FusionLSTM with focus layer (dark gray) achieves best accuracy for each test subject. However, in some cases test accuracy is close for focus FusionLSTM and either one of single modal methods. For example S03 and S07 have similar test accuracy for fusion method and RGB only method in case of 3D pose

version (top plot in Figure 4.5). If we take a closer look we can see that accuracy from pose only method (PoseLSTM) is relatively low for these subjects and proposed focus fusion mechanism tries to obtain optimal performance out of it. It works as a decision maker deciding which modality should be used and which should be ignored. Similar phenomenon is observed for S05 where focus fusion method improves low performance of RgbLSTM. Hence, we observe that in general, proposed fusion method with focus mechanism can learn prioritizing among sources to produce best possible recognition results.

Also we observe that, FoaNet like implementation has really low recognition accuracy of 15%. This was surprising to us because, the FoaNet yielded high recognition accuracy for generic gesture recognition task. We can think of several reasons for this low performance. Original work uses 12 channels of different modalities (flow, depth, rgb) while we reproduce using only 2 channels (rgb). Other reasons are the dataset size and the evaluation criteria. We conclude that when available training dataset has small number of training samples, it is very difficult to train video models with millions of parameters. On the other, fine grained hand-shape representation with a simpler sequential modeling works far better in this case. Hence, frame-wise RGB hand-shape information is vital for ASL recognition.

4.2.8 User Identification

Here we investigate the ability of our representation for user identification. This is important for security reason. If we can identify an user using a video input, we can impose a security filtering on a system. Without this filtering, the system is exposed to the world and can be exploited by an intruder.

To set up the experiment, we modify the dataset so that for each gesture example, the subject becomes the label for all videos. We have 12 classes (12 subjects in GMU-ASL51) in this setting. First, we keep aside 5 gestures from each subject for each class for training purpose; the remaining gestures are summed up to 10,431 samples and used for evaluating user identification models. Then we trained models using increasing fraction of training data. We trained a first model using 1 video sample of each class from each subject. Then

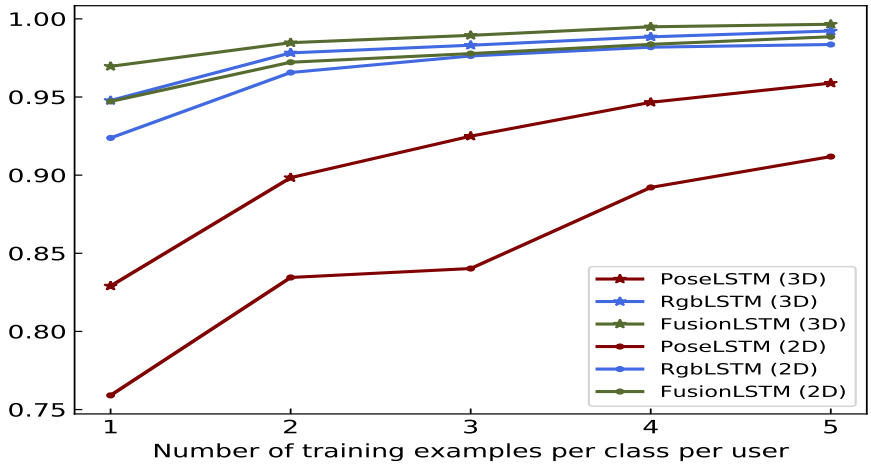


Figure 4.6: User identification results. X axis represents number of gesture video examples used for training from each sign class for each subject. Y axis denotes recognition accuracy on the rest of the gesture samples.

we used 2 videos of each class from each subject and so forth. One video per subject per class gives a total number of 535 videos for training. Figure 4.6 shows this process and the associated recognition results for models trained on different fraction of data. `X` axis represents the number of sign video examples per subject per class used as training. We observe that, using only 2 videos as training, our proposed architectures yields 98% user recognition accuracy. We also see that, PoseLSTMs did not perform as well in comparison with other methods. This is because, body joint data captures motion of different body parts which is not helpful for distinguishing specific person. On the contrary, hand data carries user specific shapes, colors and orientation. Hence, we observe more than 15% performance gain, in case of 1 training video for each class from each subject, in RgbLSTM which uses RGB hand crops. It is also interesting that, hand data used with joints' pose data, in FusionLSTM, yielded slight performance gain over other methods.

4.3 Key Takeaway

We demonstrated the effectiveness of skeletal 2D pose trajectories for ASL gesture recognition. The baseline model was further enhanced by incorporating hand-shape features along with a novel fusion mechanism. In both cases we used models pre-trained on large amounts of labeled human pose and hand-shape data. This enabled proposed networks to handle large variations in hands and upper body appearances. An extensive comparison of the proposed approach with 4 baseline approaches has been carried out. With the advances in human pose estimation from single image, the 2D skeletal gesture recognition was found to be competitive with the models using 3D joint trajectories. This is encouraging because, 2D skeletal data can be obtained without employing depth sensors in the system.

Chapter 5: FineHand: Learning Hand Shapes For American Sign Language Recognition

To recap from Chapter 4, we extracted hand-shape features from each frame using a pre-trained CNN model and used them to learn sign videos in the GMU-ASL51 dataset. Although, the CNN model was trained on hand-shape images from a different source, it improved sign modeling task by 8%. This motivated us to cultivate hand-shape representation for the GMU-ASL51 dataset. However, training a CNN for producing hand-shape representation requires annotated frame-wise hand-shape images. To solve this, we proposed a semi-automatic approach for cultivating annotated hand-shapes from the GMU-ASL51 dataset. This annotated data source is used to train a CNN based hand-shape recognizer. Our experimental results showed that the learned hand-shape representation improved the downstream sign video modeling task by a margin of 11%. Even using only RGB modality, our proposed method outperformed multi-modal fusion based approaches. This chapter describes the proposed methods with experimental results. The work presented in this chapter has been published in the 2020 IEEE International Conference on Automatic Face & Gesture Recognition (FG), Buenos Aires, Argentina.

An ASL sign is performed by a combination of hand gestures, facial expressions and postures of the body. Further, the sequential motion of specific body locations (such as hand-tip, neck and arm) provides informative cues about a sign. In this work, we consider the problem of classifying word-level ASL signs captured by short video snippets in unrestricted settings by multiple signers. We focus on learning effective hand shape representations, robust to changes of style, motion blur and illumination (see Figure 5.1). To localize the hands, we use recent advances in human pose estimation methods from video, which are effective in estimating 2D hand joint locations [23, 24, 50]. In order to train a

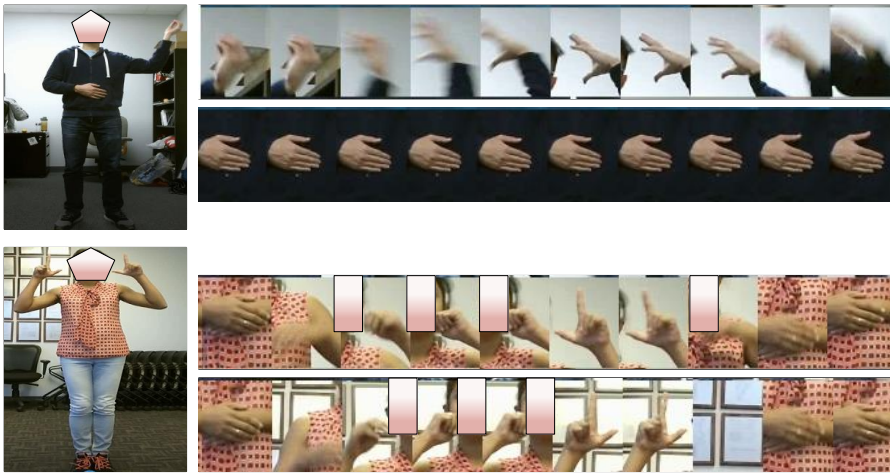


Figure 5.1: Subjects performing different ASL signs and corresponding hand shape pattern. Faces are masked for privacy concern (Note: Faces are masked in this dissertation for the same reason).

robust hand shape model, frame level annotation of hand shapes classes is required. Such annotation labelling is a tedious and time consuming process. For this reason, we first manually annotate hand-shape images from a small set of videos or, to be specific, 612 sign videos. Later, we train a CNN model on these annotated images and generate more annotated hand-shape images using the trained CNN. By iterating this process, we obtain a robust hand-shape recognizer CNN model. Finally, given a hand-shape image as input, we can extract the hand-shape features as the penultimate layer of the CNN network. These are subsequently used to learn sequential dynamics of sign video using recursive neural network (RNN) [32]. In summary, the contributions of this chapter are:

1. We proposed an iterative learning mechanism to train a deep CNN to model hand-shape representation;

2. We implemented sequential RNN models for video gesture recognition using the learned hand shape representation;

3. We evaluated our method by varying different factors such as fraction of hand-shape supervision and single hand vs both hands, and compared it with other multi-modal

methods for ASL gesture recognition;

4. We created and released hand-shape annotations for each frame of the GMU-ASL51 dataset. This will allow future research in the direction of frame-wise modeling for sign videos.

We will demonstrate superior performance of the proposed model on the GMU-ASL51 dataset [10] and compare it quantitatively with several baseline approaches which use different representations and different sensing modalities.

5.1 Our Approach

In this section, we first outline the dataset we use to experiment with our proposed sign learning approach. Then, we describe the hand-shape learning process of a CNN model. Hereafter, we refer to this model as the hand-shape model. Finally, we present the sign video classification model. This model utilizes frame-wise hand-shape representation generated by the CNN based hand-shape model.

5.1.1 The GMU-ASL51 Dataset

All of our hand-shape learning as well as ASL classification tasks were evaluated using GMU-ASL51 benchmark [10]. This dataset has 51 word level ASL signs performed by 12 subjects of different ages, gender and builds. The dataset was collected using depth sensor and has two modalities: RGB videos and 3D skeletal body parts. More detail can be found in the paper [10]. In this dataset, only video-level class label is available. One of our main contributions of this work is to systematically annotate per frame hand shape from each video. The dataset is also described in details in Section 3.1.

5.1.2 Hand Shape Learning

The goal of this part of our pipeline is to learn high-dimensional feature embeddings of hand-shape images which are representative for ASL gesture recognition. In this section

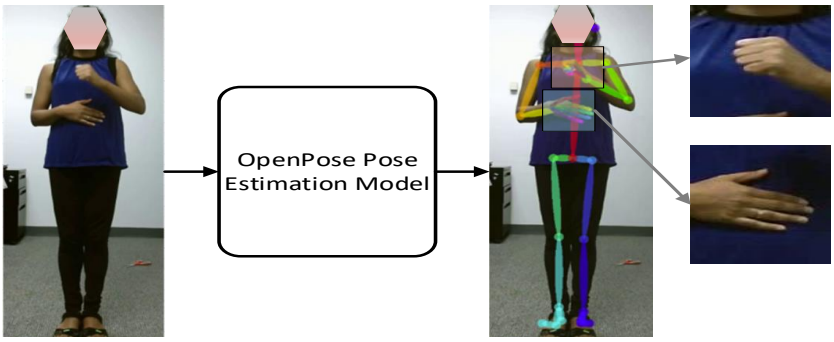


Figure 5.2: Pose estimation and hand cropping process.

we are first going to present how to crop hand patches using off-the-shelf pose estimation method, followed by an iterative hand-shape learning mechanism. Finally, we demonstrate qualitative results from our learned hand-shape model.

Hand Crops from Pose

The pose estimation from RGB is described in Section 2.3. For this work we have chosen the state-of-the-art 2D human body pose estimation approach OpenPose [23]. It is to be noted that we only use hand poses to crop a hand patch from each image. Figure 5.2 shows the whole process of estimating body poses from an RGB frame and cropping hand patches.

Iterative Hand Shape Learning

The GMU-ASL51 dataset has $13,107$ sign videos. These videos are from 12 subjects and 51 word-level sign classes. The 51 classes are video-level, meaning that, each video has an associated class label with it. However, we want to learn frame-level representation of hand shapes appearing in each frame of a sign video. Since no frame-wise hand-shape annotation is available, training a model in a fully supervised manner is not possible. Hence, we implemented a method that automatically learns frame-wise hand-shape representation. Our approach depends on some early manual hand-shape annotation. To be more specific, we took one sign video from each combination of a subject and a sign class, which yielded a total of 612 (12×51) sign videos. We extracted hand-shape crops using pose data and

Table 5.1: Iterative hand-shape learning process. In the header row P, C and T symbolizes prediction, correct and total count respectively. Iter 1 is the manual annotation, hence all labels are correct. T column of final pass denotes the cumulative count of hand-shape samples for the class represented by rows. Iteration is abbreviated as Iter.

Class	Iter 1	Iter 2			Iter 3		
		P	C	T	P	C	T
C1	402	598	534	936	524	511	1447
C2	217	277	277	494	281	277	771
C3	69	88	73	142	102	96	238
C4	328	554	408	735	435	396	1131
C5	163	236	196	359	219	198	557

manually group them based on visual similarity into 41 classes. These examples were used to fine-tune a pre-trained ResNet-50 CNN architecture [93]. In the second iteration, we used the high-confidence predictions of our initial model to predict hand-shape crops from another 612 sign videos (different from the 612 set of first iteration). At this point we may have some incorrect predictions from initially trained model. We manually corrected the incorrect predictions. Although this fix is manual, it requires less time and labor than manually annotating each hand-shape image. After this phase, we had more annotated hand shape patches and we retrain our model. With additional iterations, the model becoame stronger in predicting hand-shape images. Similarly we can start a third iteration and so on. We performed 3 such iterations and ended up with 41 classes of hand shapes which are distributed among 51 sign gesture classes of GMU-ASL51 dataset. Table 5.1 shows the count of annotated hand shape samples for five classes and three iterations. It should be noted here, in three iterations the fraction of sign gesture data we used to train the model were 4.16%, 8.32% and 12.5% respectively. It should be also mentioned that, we used per frame hand shape annotation on this fraction of data which means that a video gesture sample could possibly generate several hand shape training examples. Effects of this incremental learning on sign classification is discussed in more detail in the result section. Among 41 hand shapes two are unusual: the garbage and the rest-position. Keeping those two classes

Figure 5.3: Sample hand shapes. Each row shows 12 randomly picked samples from the created dataset in the iterative hand shape learning phase.

is vital because, in a sign video, sometimes hands appear either blurry or are at the rest position. As shown in the bottom two rows in Figure 5.3, these hand shapes are uninformative for the sign video modeling task. Hence, if we have a way to learn these unnecessary hand shapes, then we can exclude them during sign language modeling and hence have robust feature representation. Figure 5.3 shows examples of several hand-shape classes picked from our cultivated 41-class hand-shape dataset. Each row represents one class. Twelve samples in a row are picked randomly from our created hand-shape dataset. In each row, we observe noticeable intra-class variations such as flipped hand shapes, backgrounds, hand-arm orientations, and blurs. Bottom two rows show the sample hand shapes from the class `garbage` and `rest-position` respectively. The ResNet CNN model we trained here is a module of the proposed FineHand sign video classification pipeline. After being trained on hand-shape data, parameters of this model can be freezed during temporal sign video learning.

Qualitative Results

The hand-shape CNN model is trained in an incremental fashion. Manual annotation in the first iteration and all the other predicted hand-shape images can be from either left or

Figure 5.4: Predicted hand shape classes from a trained ResNet-50. For each of three sign classes two samples are shown where Qy means query hand shape sequence and Ref means a reference sample of predicted label for each corresponding query hand patch from the training samples.

right hand. This means if we look at all the samples from a class, we will find examples from both hands. Of course, a shape will be horizontally flipped or rotated as we look at the left versus right hand. First, second and fourth rows in Figure 5.3 show such examples. Figure 5.4 presents predicted and reference examples patches using a trained hand shape model, four top rows show two samples (divided by thin black line) from the sign `play`. For each samples the second row shows the query (Qy) hand crops and the corresponding crops in the row above (Ref) represent a reference training sample from predicted class. This classification model is trained on a cross-subject manner, which means none of the hand crops from the test subject (query) is used during the training procedure. We observe, in most of the cases, hand-shape model learns useful feature representation which is invariant to rotations, scales and backgrounds. We will then keep the penultimate layer of the model as the high-dimensional representation of each hand crop.

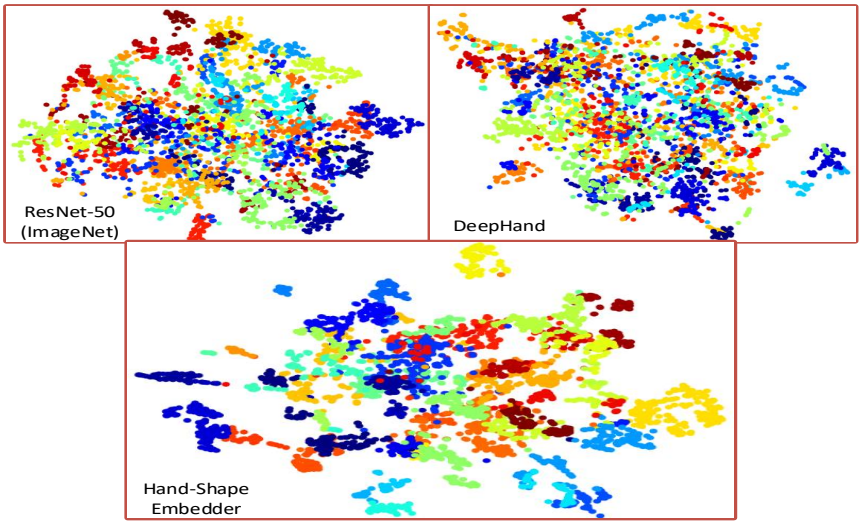


Figure 5.5: T-SNE visualization of hand-shape embeddings from different model sources for 4,033 samples. Top figures show representation obtained from ResNet-50 (ImageNet pre-trained) and DeepHand. Bottom figure shows embedding obtained from a trained (cross subject) ResNet model using hand-shape labels created using our proposed method.

Figure 5.5 shows the comparison among T-SNE representation of embeddings obtained by our model. It shows that hand-shape representation learned by our CNN embedder cluster better than embeddings produced by ResNet (ImageNet trained) [93] or DeepHand [12] models. This gives us the motivation behind using this effective representation in sign video classification task.

5.1.3 Temporal Sign Gesture Learning

As pointed out and shown in the previous section, learned hand-shape representation could be useful in classifying ASL gesture videos. However, the sequential dynamic in video data still needs to be modeled carefully for better classification accuracy. With a trained hand-shape CNN embedder, each video can be converted into a sequence of hand-shape representation vectors. We base our sequential modeling using these vectors as inputs to LSTM networks. Section 2.2.1 details the LSTM network.

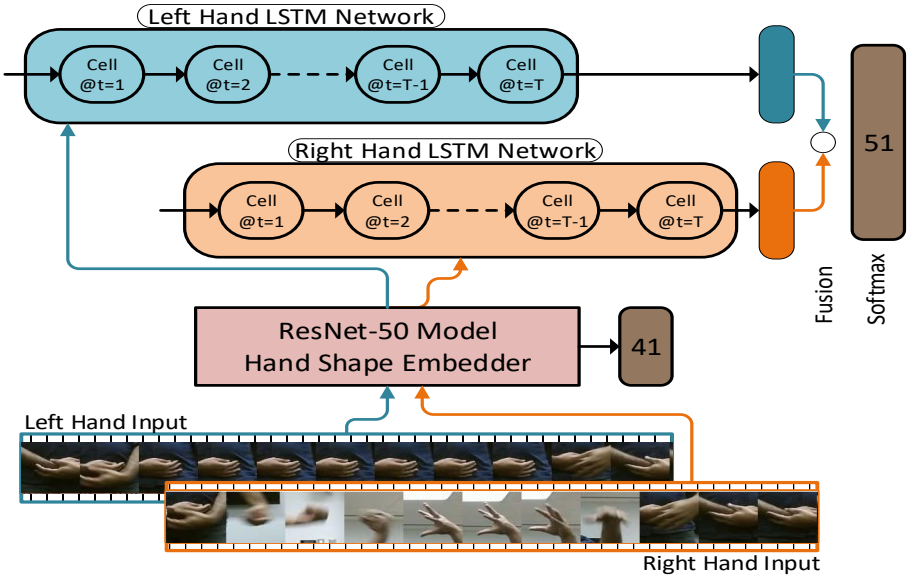


Figure 5.6: FineHand RNN model. The ResNet-50 model is trained separately first on 41 hand shape classes. After training, it provides representation for each hand-patch video which is then used in sequential LSTM classifier for 51 sign classes in the dataset.

FineHand Architecture

We proposed an LSTM based method for the sign classification task. For a sign video, input to this model is a sequence of vectors obtained from our hand-shape CNN. Assume that we have a sequence of cropped hand images $R^{F\times H\times W}$ where F, H, W are number of frames, height and width of hand crops respectively. Feeding the images to the hand-shape model gives 2048 dimensional embedding for each frame's crop and the sequence becomes $R^{F\times D}$, where $D = 2048$ for ResNet-50 CNN. We used this data to train an LSTM model where F is the number of temporal steps. To deal with varying video lengths in the dataset, we sampled T predetermined number of frames uniformly. Hence, for a given video, input to the LSTM network is $R^{T\times D}$. Finally we picked the hidden state at the end of last layer of LSTM network and took that as an encoded representation for each sequence. This final representation captures rich temporal as well as spatial hand-shape features and is fed to a fully connected neural network layer to produce prediction probability distribution

for sign gesture video classification. We trained two different networks for left and right hands. We fused the output of two networks at the end to produce final classification scores. Figure 6.2 shows the details of our proposed architecture. It shows that the left-hand and right-hand patches are input to the trained hand-shape embedder model and the generated representation is being used as input for the recurrent LSTM networks.

Training Details

Average cross entropy loss is used to update the network parameters for a batch. Given a true one hot encoded class label of $y_{i,j}$ and corresponding predicted score of $\hat{y}_{i,j}$ this loss is calculated as Equation 5.1.

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{i,j}\log\left(\hat{y}_{i,j}\right) \tag{5.1}$$

Here, N is the size of the minibatch and C is the number of classes. For our experiments, these values are 64 and 51 respectively. We performed grid searching for selecting the hyper-parameters of our model. These hyper-parameters are the number of layers in our network, state size and time steps of the input sequence. We found best validation accuracy is obtained with 2 layer networks, 512 as hidden state size and 20 (T) as the input sequence length. Size of the input dimension at each time step is the size of the representation produced by hand shape embedder, or 2048 to be specific in case of the ResNet-50 CNN. We used the Adam Optimizer for training our networks [86] with learning rate set to 0.0001.

5.2 Experiments

In this section, we are going to describe different methods we used in comparison with our proposed architecture.

5.2.1 Comparative Methods

3D Convolution Convolutional Neural Network (CNN) with 3D convolutional kernel (3D CNN) has shown promising performance in classifying human activities in video [14]. That's why we chose 3D CNN on RGB hand patch videos for one of our baselines. It consists of four 3D convolutional layers and two fully connected layers at the end. There are two separate networks for left and right hands' patches. Final embedding of these two networks are concatenated before producing softmax score.

Deep Hand Model The authors in [12] trained a 22 layer deep convolutional neural network (CNN) with more than 1 million images, from videos of Danish and New Zealand sign language. The data is weakly labeled with only video level annotation. The CNN model for estimating the likelihood of hand shapes, is trained using EM algorithm, jointly with Hidden Markov Model (HMM) for parsing sign gestures. The network is trained to recognize 60 hand shape classes plus one garbage class which determines the start or end of a video. We forgo the softmax classification layer of this network and use the embeddings computed by this pre-trained model as representations of hand-patch crops from our ASL videos. Given a hand-crop image as input, the final layer representation produces a 1024 sized vector as hand features. We compare the hand features obtained from this source with our proposed hand-shape learning method in the context of sign video recognition.

Kinect 3D Pose For this experiment we used RGB video and 3D skeletal pose data as computed by Kinect sensor [43]. We performed three types of experiments using these two types of data. Two of those experiments use each modality separately. The other one uses fusion strategy to get maximum out of both modalities. These models are referred as 3D version of PoseLSTM, RgbLSTM and FusionLSTM in the result section.

OpenPose This experiment is similar to the process described in last paragraph except only uses estimated poses from RGB videos instead of using 3D skeletal poses generated by Kinect sensor. Process of estimating poses from video is described in Section 2.3. Rationale

behind doing this experiment is to exclude the dependency on a depth sensor. Since, this poses are estimated from RGB video, this experiment depends solely on RGB modality. These models are referred as 2D version of PoseLSTM, RgbLSTM and FusionLSTM in the result section.

Comparison with Similar Work It was difficult to find a similar work which can be directly compared to our method. This is due to the nature of our set up which is isolated ASL word level sign recognition. We could not find any standard public dataset other than recently released GMU-ASL51 for this purpose. However, there is a public dataset on generic gestures [94]. Various deep learning based methods have been proposed to model generic isolated gestures [52, 90–92]. Narayana *et al.* proposed FOANet which uses fusion of different channels on cropped hand patches and full body [52]. Using different modalities (RGB, depth and flow) with those channels this work sparsely fuses 12 channels of inputs to model gestures. Details can be found in the paper [52]. We tried to reproduce this work on GMU-ASL51 as closely as possible. However, the exact re-implementation was not possible due to several factors such as number of data channels used, number of location features of hand patches and mechanism of cropping hand patches.

5.2.2 Results

Table 5.2 shows our experimental results. All of the mentioned models are evaluated in cross-subject manner. For a particular test subject, we trained our models using data from all other subjects in the dataset. This cross-subject evaluation criteria attempts to mimic a practical scenario, where we want to measure how a trained system responds to unknown subjects. Each column in Table 3.1 shows test accuracy of one subject in GMU-ASL51. First two rows shows the baseline results: 3D CNN and FoaNet style implementation. Next two blocks of three rows show experiments with 3D and 2D poses respectively. Finally, bottom row shows results of our proposed method (FineHand). Result shows that methods using 2D poses achieve almost similar performance to 3D Kinect poses (88% vs 86%) which

Table 5.2: Cross-subject test accuracies for 12 subjects. In header row each subject is represented by S appended with subject number. The bottom row shows the result of our proposed FineHand architecture. Other rows are different comparative methods. 3D labeled three rows show the results using DeepHand embedding with Kinect 3D pose. 2D labeled rows show the similar experiments with OpenPose poses.

		S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	S11	S12	Average
	3D CNN	0.62	0.63	0.60	0.60	0.56	0.56	0.62	0.45	0.39	0.51	0.46	0.17	0.51
	FoaNet[52] Style	0.18	0.22	0.10	0.19	0.11	0.28	0.15	0.17	0.09	0.12	0.11	0.05	0.15
3D	PoseLSTM	0.81	0.88	0.68	0.84	0.88	0.85	0.85	0.78	0.81	0.83	0.76	**0.83**	0.82
3D	RgbLSTM	0.81	0.82	0.85	0.89	0.81	0.89	0.93	0.69	0.83	0.72	0.81	0.37	0.78
3D	FusionLSTM	0.90	0.93	0.88	0.93	0.90	**0.95**	0.96	0.84	0.91	0.89	0.89	0.67	0.88
2D	PoseLSTM	0.83	0.90	0.89	0.92	0.90	0.92	0.95	0.88	**0.94**	0.93	0.94	0.63	0.89
2D	RgbLSTM	0.80	0.82	0.88	0.93	0.82	0.92	0.95	0.79	0.90	0.75	0.92	0.27	0.81
2D	FusionLSTM	0.85	0.85	0.93	**0.95**	0.87	0.92	0.98	0.83	0.93	0.85	0.96	0.40	0.86
	FineHand (ours)	**0.93**	**0.98**	**0.97**	0.91	**0.91**	0.93	**0.99**	**0.88**	0.91	**0.94**	**0.96**	**0.83**	**0.93**

is interesting because 2D pose methods depend only on RGB data. It should be noted that, unlike Kinect sensor, OpenPose provides finger joints. We presume, even though these poses lack depth information, finger joints help to achieve comparable performance with 3D Kinect poses.

From Table 3.1 we observe, our proposed method, FineHand outperforms top models using 3D and 2D poses by 5% and 7% respectively. It should be mentioned that, pose based models use pose data while FineHand model only uses RGB hand patches. Taking this into consideration, it is fair to compare FineHand with RGB only versions (RgbLSTM) of 3D and 2D implementation. In that case, FineHand outperforms those implementations by 15% and 11% respectively. This boost in the classification accuracy can be justified by the way FineHand learns hand shapes. Representation used for RgbLSTMs is taken from DeepHand, a pre-trained model on large amount of hand shapes data from a different class distribution as described in Section 5.2.1. On the other hand, FineHand embedder learns representation from annotated hand shapes which has proven to be crucial for this kind of 15% performance gain in classification. Our best method outperforms the work came with the dataset [10] by 12%.

Table 5.3: Average cross-subject sign recognition accuracy on different iterations of hand-shape learning

Iterations (% Train Data)	Accuracy
Iteration 0 (0.00%)	0.65
Iteration 1 (4.17%)	0.89
Iteration 2 (8.32%)	0.91
Iteration 3 (12.5%)	0.93

The FOANet style implementation on GMU-ASL51 has low performance (second row in Table 5.2) even though it is one of the top performing models for generic gestures. One possible reason is the number of data channels used. While original work uses 12 channels, in our implementation we use only 2 channels to make it comparable with our proposed work. Another reason is the training procedure. FOANet architecture proposed to capture sequential dynamics in a video gesture by using sliding window based approach where classification scores for a video gesture was computed by taking averages over all sliding window scores. Our method however, pre samples a fixed number of frames from a video and produces one set of prediction score per video gesture.

Effect of hand shape Learning Iterations In section 5.1.2, we briefly described how hand-shape learning CNN was trained in successive iterations. We hypothesize that increasing these iterations will boost up the sign classification accuracy. Table 5.3 shows the average recognition accuracy on using different fractions of data for training the embedder CNN. Here, 'Iteration 0' represents no hand-shape learning, meaning that, we use embedding representation from ImangeNet pre-trained CNN model as input to LSTM network for sign classification.

Both Hands vs Single Hand We are also interested to compare results obtaining from either using left or right hand and using them together. Usually some signs are dominated by single hand while others are double handed. It is obvious that, using both hands' information will increase the accuracy. However, we want to see how much improvement

Table 5.4: Average cross subject sign recognition accuracy for different scenarios of hand usage.

Input Types	Accuracy
Left Hand	0.65
Right Hand	0.90
Both Hands (Max)	0.86
Both Hands (Catenation)	0.92
Both Hands (Mean)	0.93

is possible using both hands. In case of using both hands, we also show if there is any best fusion mechanism. Table 5.4 outlines this results. We observe that, good accuracy is achieved using only right hand input. This is not surprising, because all of the subjects use right hand as the dominant hand in GMU-ASL51 dataset. However, using both hands we have 3% improved accuracy which suggests that, in some cases left hand is also important. Among the fusion strategies we found, averaging scores works best.

Table 5.5: Average cross-subject sign recognition accuracy for different learning mechanisms (joint vs separate).

Train Type	Accuracy
Separate Learning	0.93
Joint Learning	0.89

Effect of Joint Learning This section shows comparison between learning the whole network jointly and learning separately. Our default set up is separate learning where first we train the hand shape CNN model using annotated hand patch data, then freeze it during sequential learning of embedding produced by it on training video hand patches. In case of joint learning, we don't freeze the hand shape CNN network during sequential sign learning. We notice that joint learning worsen the performance of the whole network. We

anticipate, since the CNN is first trained on hand shape supervision, it might get confused when sign video-level gradient updates are done on its parameters. Also, during back propagation gradient has to travel backwards through the LSTM network before hitting hand shape CNN model. This causes derogatory updates on CNN parameters. Hence, produced embedding representation differs in each iteration, which could impact the LSTM learning negatively. Table 5.5 depicts this result.

5.3 Key Takeaway

We have demonstrated the effectiveness of learning hand-shape representation in the context of ASL sign video modeling. We showed qualitative results for the representation produced by our proposed hand-shape learning mechanism and compared with previous methods. We also verified, this representation can achieve superior sign classification accuracy than other sources of representation. Our proposed method is RGB-only but outperforms multi-modal (RGB and pose) approaches for sign language recognition. In addition to that, different factors of training such as single hand vs both hands, separate training vs joint training, and fractions of hand-shape data in training, were compared and explained.

Chapter 6: Hand Pose Guided 3D Pooling for Word-level Sign Language Recognition

This chapter focuses on body-pose guided feature extraction from Convolutional Neural Network (CNN). In a traditional word-level sign language recognition (WLSLR) setting, the pose data is used as input to a machine learning (ML) model such as CNN, and goes through linear transformations before producing the output scores associated with target classes. However, in this work, we proposed to use pose data to localize some specific body location in the feature-map space. It is to be noted that, the CNN produces feature maps by transforming the input RGB video while the pose inputs are used only to localize several pre-specified locations in the feature map. Our experiments showed 10% to 12% improvement, in some cases, in sign video recognition accuracy in a state of the art WLSLR benchmark dataset. The work presented in this chapter has been published in the 2021 IEEE Winter Conference on Applications of Computer Vision (WACV).

The basic components of a sign gesture are complex arm movements with articulated hand shapes and facial expression. Both the motion and the shape of the hands are the most discriminative components of individual gestures. From computer vision perspective, the word-level gesture recognition requires learning strong *spatio-temporal* representations from videos, capturing both the appearance of the hand as determined by its shape and pose, as well as motion of arms and hands. Several deep-learning based approaches were found to be effective in capturing *spatio-temporal* representations of action videos on commonly used action recognition benchmarks [13, 95]. The common building blocks of these models use combination of Deep Convolutional Neural Networks (ConvNet) for extracting spatial features and Recurrent Neural Networks (RNN) [96, 97] for temporal features, or used combination of 2D and 3D CNNs for fusing spatial and temporal cues [14, 97–99].

These methods take action videos as input and compute class prediction probabilities over the available classes or occasionally action localization depending on the labels available in the training set. Complementary approaches for action recognition exploit the existing techniques for human pose estimation [23] in individual frames and learn models on the top of these representations [64, 100]. Motion blur and limited resolution make the body pose estimation very brittle. The same factors affect most of the estimation of hand and finger joints, making these methods ineffective for capturing hand shape and pose. For both action recognition and word-level sign language recognition 3D ConvNets are currently one of the best performing models [1, 5, 13] enabling an end-to-end joint training of *spatial-temporal* component. These approaches treat a sign video as any other videos, disregarding the importance of hand shapes in sign videos. For the word-level gesture recognition, previous methods that shown the effectiveness of hand shapes for gesture recognition required training separate models for hand-shape classification, in supervised or weakly-supervised way [12, 101].

Motivated by the effectiveness of hand shapes for gesture recognition, we proposed to improve 3D ConvNet model by additional predictions obtained by pose-guided pooling of 3D convolutional features maps at different layers and levels of resolution. We used the spatial locations of hands in the feature map space to guide the pooling. Location of hands can be reliably estimated by the state-of-the-art of pose estimation methods [23]. We learned to predict the word-level gestures by training additional classifiers using pose-guided pooling of 3D ConvNet feature-maps with different spatial supports. During the test time, we fused the class probability scores from classifiers that were trained using these pooled multi-scale features. Figure 6.1 shows an example of body poses mapped to corresponding locations in different feature layers of a 3D ConvNet. In summary, our contributions can be listed as follow,

1. We proposed a novel pose guided pooling mechanism for word-level ASL recognition;
2. We demonstrated the effectiveness of the idea of pooling localized features from multiple feature-map levels of a 3D convolutional network;

Figure 6.1: Top images show the variation in hand position in three different samples from a sign gesture of `city` class. Bottom images shows how the hand poses are mapped (for the middle sample) to corresponding activation maps for four randomly selected channels from a certain layer of a 3D ConvNet.

3. We evaluated the proposed architectures improving the state of the art results on word-level action recognition;

4. We demonstrated that our feature pooling mechanism produces features that are better transferable to datasets obtained from different sources.

6.1 Our Approach

In the word-level sign language recognition problem, we have a dataset on N training examples $\{V_i, W_i\}$ where V_i is a RGB video $\in \Re^{T \times H \times W \times 3}$ and W_i is a word level label, where $H, W, 3$ is the dimension of single frame of video and T is its length. Example gestures for the sign `city`, performed by three different signers is in Figure 6.1.

We first describe the baseline 3D convolutional neural network model, followed by our proposed pose guided pooling and fusion approach.

6.1.1 Inflated 3D ConvNet

Deep convolutional neural networks (ConvNet) for image classification proceed by learning layers of shared filter parameters that are used obtain spatial features maps by means of convolution. To model sequential nature of video data, convolution can be extended to 3D where weights of 3D filters can learned directly from spatio-temporal data [14]. However, training these models from scratch is quite challenging, due to the large number parameters added by the temporal dimension of convolutional filters. To mitigate this issue, I3D network [13] proposed to use already trained 2D convolutional filters and inflate them into 3D to initialize the training of 3D convolutional network. Inflating pre-trained 2D filters into 3D allowed the network to learn seamless *spatio-temporal* features and has become the state-of-the-art method in action recognition and world-level sign gesture recognition [1,13]. GoogleNet was proposed to find the optimal sparse local structure in data using readily available dense convolutional fiters [102]. The idea was motivated by the fact that, not every neuron, at each layer is equally responsible for the learning process of a ConvNet and optimal network structure can be approximated using the correlation statistics of highly activated neurons layer by layer [103]. The authors of I3D [13] inflated – or extended the 2D convolutional kernels to 3D – a version of GoogleNet and initialized 3D filters using 2D versions trained on image recognition task. This architecture is titled as Inflated Inception-V1 and we use this as one of our baselines as well as a starting point in our proposed model. Figure 6.2 shows the inception module as `Inc`. Details can be found in the original paper [13].

6.1.2 Proposed Method

In our approach, we proposed to augment a 3D ConvNet (CNN) model with body-pose guided pooling. We assume that the body pose estimates are available for each frame using an off-the-shelve pose estimation approach [23]. The video is passed through the layers of the 3D ConvNet generating spatio-temporal features maps with multiple channels at different levels of resolution. We then use the estimates of body joint locations to guide the

pooling of spatio-temporal feature maps to generate additional predictions. Details of our architecture is presented in this section.

We denote a sign gesture video input by $V^{T\times H\times W}$ where T is the video length in number of frames and H, W are the spatial dimensions. For a person performing a sign gesture in the video, the estimated body pose tensor of the person can be represented as $P^{T\times J\times 2}$ where T is the number of frames, J is the count of body locations and 2 for the (x, y) location coordinate of a body location on image space. The input layer and the subsequent layers of a 3D CNN reduces the spatial and temporal dimension of the input cubic video space. This reduction happens due to the presence of striding and pooling operations in a CNN architecture. Suppose we have a feature map at level k with dimensions $F^{T'\times H'\times W'}$. The spatial joint coordinates can be scaled down based on the ratios of heights and widths between the input video and the feature map. For the temporal dimension we uniformly sample T' frames from initial T, to match the temporal dimension of the feature maps. More specifically, we convert an input tensor index (t_v, x_v, y_v) to feature map index (t_f, x_f, y_f) using following equation,

$$\begin{gathered} s_x = \frac{H'}{H} \;\; s_y = \frac{W'}{W} \;\; s_t = \frac{T'}{T} \\ x_f = s_x \times x_v,\; y_f = s_y \times y_v,\; t_f = TemporalSample(T, T')[s_t \times t_v] \end{gathered} \tag{6.1}$$

Here, s_x and s_y denotes the scaling factor due to the spatial change of resolution for height and width dimension respectively. The function $TemporalSample$ divides the T temporal dimension into T' equal length windows and then picks the middle index from each window and finally returns a list of temporal indices. To get the corresponding temporal index of pose data to the feature map, $(s_t \times t_v)^{th}$ number from the sampled indices is selected. Given the computed joint locations in the feature-map space, we use these to guide the feature pooling to generate the new feature vector at a layer.

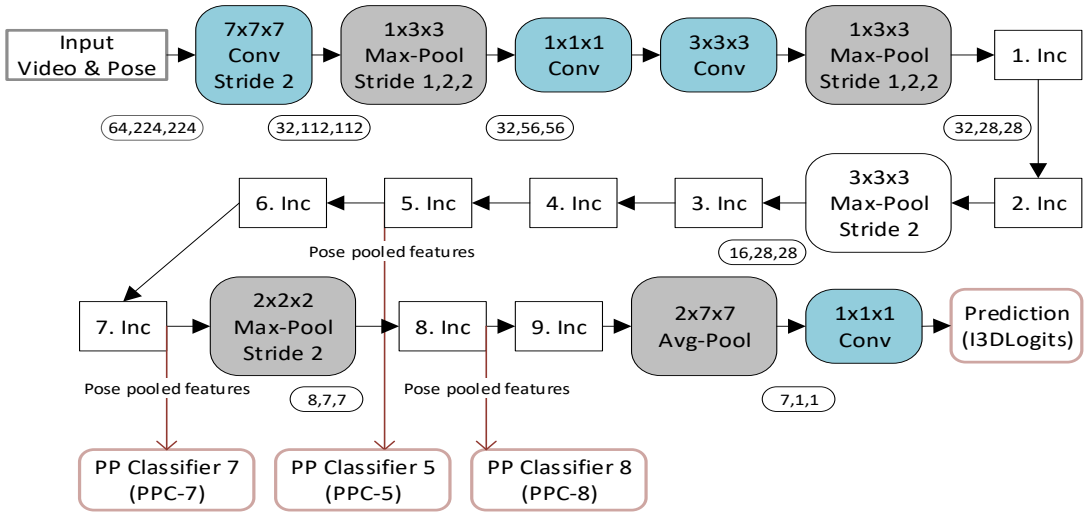


Figure 6.2: I3D Inception-v1 based sign video recognition pipeline. All inception blocks (Inc) are numbered for the convenience of description. Volume of output is labeled as "temporal,height,width" after any layer where it is being changed by the previous layer's sampling and convolution filters. Number of feature maps are not shown for simplicity in any output volume. Pose pooled classifier is shown for three output locations (PPC-5, PPC-7 & PPC-8), while it can be done from any output points in the network.

Pose Guided Pooling Classifier (PPC)

We pool features around each joint location and stack all joints' feature. Pooling from a feature map of $[f \times t \times h \times w]$ using j joint locations leads to a feature representation of size $[f \times t \times j]$. Using these features we train a separate linear softmax classifier described in more detail in the experiments section. The architecture learns multiple classifiers separately and during test time we fuse their prediction scores. Since these classifiers use features from different scales of 3D convolutional feature maps, they should carry complementary information. This fact is reflected in overall performance improvement using our pose pooling and score fusion mechanism. Figure 6.2 shows the overall architecture, comprised of an I3D backbone network with labelled inception modules. This figure shows, `PP Classifer 7 (PPC-7)` gets pose pooled features from the inception layer labeled 7, and `PPC-8 & PPC-5` from levels 8 and 5 respectively. We can extract features from any 3D feature map level

Table 6.1: Summary of the different subsets of WLASL dataset where mean is the average number of video samples per gloss. More details can be found in the paper [1].

Datasets	#Gloss	#Videos	#Mean	#Signers
WLASL100	100	2,038	20.4	97
WLASL300	300	5,117	17.1	109
WLASL1000	1,000	13,168	13.2	116
WLASL2000	2,000	21,083	10.5	119

and train a separate classifier. For the rest of the discussion, We refer to classifier from n^{th} inception module as `PPC-n`. We also consider the final prediction of baseline I3D network termed `I3DLogits` and experiment with different fusion strategies described in experiments section.

6.2 Experiments

6.2.1 Dataset

We performed all of our experiments using recently introduced WLASL dataset [1]. The dataset was curated from online ASL videos, primarily created for tutorial purposes. Being collected from different sources, the dataset contains unrestricted varieties in signing styles and background. The authors performed several manual and automated processing steps to create four subsets of data: WLASL100, WLASL300, WLASL1000 and WLASL2000. Table 6.1 shows the statistics of the dataset.

6.2.2 Preprocessing

We downloaded the data following the instructions provided with the dataset release. We run all the videos through OpenPose [23] and stored the estimated poses. Using the pose information on the image frame, we calculated a bounding box for each video. We ensured that both hands and the whole body are visible over all the frames of a video. Then we cropped each video using that bounding box. After cropping the videos, we adjusted

the poses according to the cropped region. Finally, each cropped video and corresponding adjusted pose formed a sample input to our network.

6.2.3 Implementation Details

For our pose pooling methods, we used only 2 joints to extract features from the intermediate feature maps of I3D network. These 2 joints are calculated using the mean joint location of all 21 finger and palm poses for each hand. Hence, a feature map of $[f \times t \times h \times w]$ is converted to $[f \times t \times 2]$. We empirically verified that, adding more joints, such as elbow or shoulder, in pooling does not improve results significantly. This can be understood by the fact that most of the variances explaining the data come from the hand regions. We used maximum pooling using $3 \times 3 \times 3$ kernel around each hand pose location. We also noticed, adding fully connected layer afterwards does not improve or deteriorate the performance. Hence, we decided not to use it. Once we extract pose localized features, training mechanism follows the original I3D [13]. We initialize the I3D network using pre-trained weights on Charades [95] activity dataset. Each video is resized into 256×256 and the poses are adjusted accordingly. We used two video level data augmentation techniques: random cropping using 224×224 spatial support and random horizontal flipping. Input video length is set to 64, with possible temporal augmentation in case of longer videos. We padded the videos less than 64 frames either in the beginning or end. Poses are adjusted appropriately in case of any augmentation of video data. We used the Adam optimizer with an initial learning rate of 0.001 and with 4 or 6 (for different subsets of data) mini-batch sizes. After fixing the hyper-parameters of a model, we trained the model using the training and the validation split and reported the results on the test split for each subset of the dataset. Average instance-level accuracy was used as the performance metric following the dataset release paper [1].

Table 6.2: Top-1, Top-5, Top-10 accuracy (%) achieved by each model (by row) on the four WLASL subsets. First row shows the results reported in [1]. Next two rows (PPC 7 & PPC 8) shows performance from two classifiers of our pose localized pooling. The `I3D Logits` result is the basic I3D classifier without any pose pooling mechanism. Here, Fusion-1 = PPC-7 + PPC-8, Fusion-2 = PPC-7 + PPC-8 + I3DLogits and Fusion-3 = PPC-5 + PPC-7 + PPC-8 + I3DLogits.

Method	WLASL100			WLASL300			WLASL1000			WLASL2000		
	Top-1	Top-5	Top-10	Top-1	Top-5	Top-10	Top-1	Top-5	Top-10	Top-1	Top-5	Top-10
I3D[1]	65.89	84.11	89.92	56.14	79.94	86.98	47.33	76.44	84.33	32.48	57.31	66.31
PPC-7	67.79	76.91	80.75	57.12	70.80	74.44	44.17	63.33	69.76	29.46	52.95	60.25
PPC-8	67.79	78.16	82.50	59.91	75.78	78.39	44.57	61.11	66.20	29.26	50.35	56.57
I3DLogits	68.70	86.66	89.58	57.62	78.63	82.46	48.29	68.25	73.17	33.18	60.04	68.87
Fusion-1	71.74	81.75	84.66	64.41	78.67	82.39	51.01	70.95	75.80	34.68	60.39	67.27
Fusion-2	74.16	**86.83**	**90.91**	67.79	**84.19**	**87.06**	55.71	77.90	83.77	38.57	**68.17**	75.71
Fusion-3	**75.67**	**86.00**	90.16	**68.30**	83.19	86.22	**56.68**	**79.85**	**84.71**	**38.84**	67.58	**75.71**

6.2.4 Evaluation Results

Table 6.2 shows the experimental results. The best accuracy is achieved by combining four sets of prediction scores from PPC-5, PPC-7, PPC-8 and I3D Logits. This version, titled as Fusion-3, outperforms the I3D implementation in [1] by approximately 10%, 12%, 9% and 6% in case of four subsets respectively. For the single layer classifiers, the result indicates that our pose localized classifiers (PPCs) perform competitively with the base I3D network. It should be mentioned that, we have a performance gain (68.70% vs 65.89%) using base I3D, shown as `I3DLogits`, over the same baseline's results reported in [1]. Although both of these are same implementation, we believe, the performance increase is due to the cropping pre-processing step. The results indicate that any single `PPC` classifier such as `PPC-7` outperforms the I3D baseline. This is interesting because, single `PPC-7` classifier uses less number of model parameters than the whole I3D. Hence, our pose guided pooled features achieves better performance using less memory and computation.

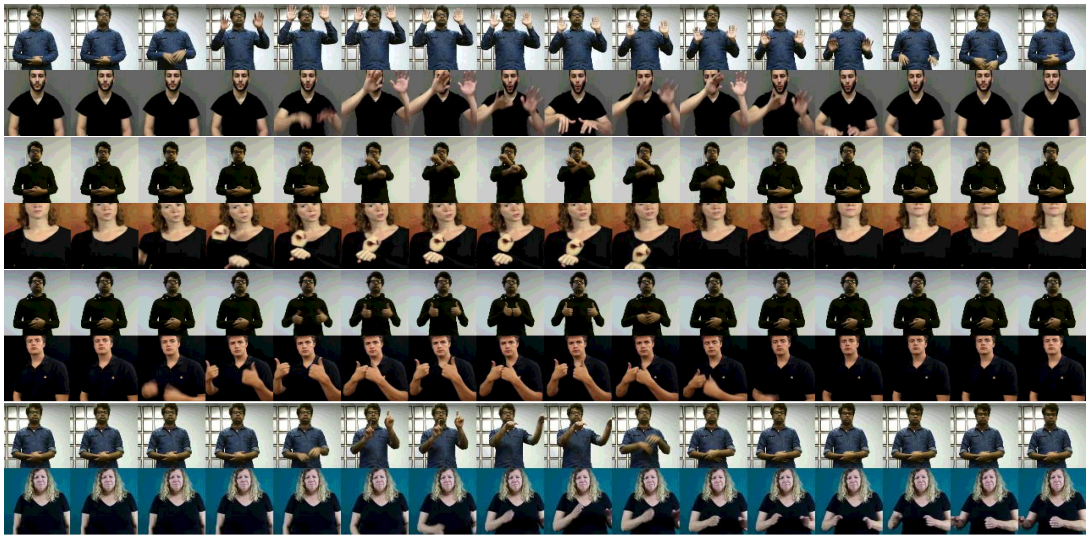
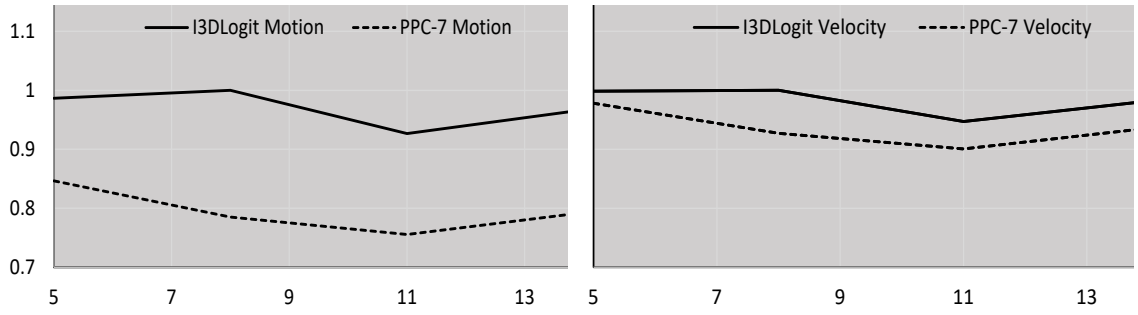


Figure 6.3: Examples of gesture samples from four classes. Each pair of sample shows one example from WLASL dataset (bottom in a pair) and other different dataset (top in a pair)

Complementary Feature Learning We wanted to verify that, improvement from fusing the prediction scores is not just an effect of model ensemble. In this regard, we also calculated the accuracy fusing scores from same branch but from separately trained model. For example if we fuse the scores from PPC-7 of two separately trained models, the top-1 accuracy the fusion achieves is 68.05% on WLASL100 subset. Fusing three models from PPC-7 gets 68.17%. Although, these are a bit better than result from a single PPC-7 (67.79%), but far worse than the fusion of PPC-7 & PPC-8 (`Fusion-1`, 71.74%). Similarly, fusing scores from two PPC-8 layers from two separately trained models gives 68.15%. This suggests that performance improvement is not merely coming from using multiple models at test time. Rather, different pose localized branches pick on different class specific features and complement each other to obtain better performance.



(a) Average hand motion

(b) Average hand velocity

Figure 6.4: Hand motion of top performing classes from PPC-7 branch and I3D logits.

6.2.5 Qualitative Findings

Our approach is built on the hypothesis that pooling features from different layers can be complementary for overall performance. We also found this in our experiment. Experimental findings revealed that the classes having less hand-arm motion get best feature from intermediate layers, while, the sign gestures with relatively heavy hand-arm motion get benefits from final layer of the I3D network. To run this experiment, first we picked top performing classes, the classes having higher fraction of samples correctly classified, from `PPC-7` branch. We calculated the hand motion using pose location of both hands for those picked classes. We repeated the same for top performing classes from the final logits (`I3DLogits` in Table 6.2) of the network. Figure 6.4 (a) shows the plot of hand motion where horizontal axis is the number of top classes we pick. We observe that the calculated average motion of best performing classes for `PPC-7` is always lower. This suggests that the classes having less hand-arm motion get useful features from `PPC-7` than `I3DLogits`. We believe, due to less motion, hand shapes of these classes are more visible to the network and the pose localized pooling helps to extract those informative shapes. Figure 6.4 (b) shows similar phenomenon, except we calculate average motion of a certain sized window. We calculate distance of the first and the last frame of each window and the window slides over the video with a stride of 1 frame. This is proportional to the average velocity of hands in

a sign sample and we observe the similar results as the motion case.

6.2.6 Representation Transfer

We conducted this experiment to validate the strength of pose-pooled features on unseen data samples from a different distribution than the WLASL dataset. We selected 12 com-

Method	Accuracy		
	Top-1	Top-5	Top-10
I3DLogits	55.63	75.67	82.25
PPC-8	59.21	89.48	88.63
Fusion-3	**66.70**	**91.80**	**98.11**

Table 6.3: Fine tuned results using only 0.4% of data in training. Method names have same meaning as in Table 6.2.

mon classes from the WLASL300 subset and the GMU-ASL51 dataset. We curated approximately 2900 samples from GMU-ASL51 dataset for those overlapping 12 classes. Figure 6.3 shows some examples from the both datasets. It is obvious from the examples that huge amount of difference in the distribution exists between two datasets. We tested our fusion scheme on these 2900 target samples from GMU-ASL51 dataset using a trained model on source WLASL300 dataset. Our fusion scheme (Fusion-3) achieved 21% recognition accuracy, while I3D only achieved 15.6%. This result is without any fine-tuning, meaning that, the network is not re-trained on any sign videos from the target GMU-ASL51 dataset. However, if we fine-tune the model using 12 sign videos, taking 1 sign video per class, the `Top-1` accuracy increased up to 66% in case our fusion approach. This result is 11% higher compared to I3D features without our proposed pose guided pooling. The `Top-5` and `Top-10` accuracy improvement are approximately 16%. Table 6.3 demonstrates this results. This verifies that the learned complementary representation using pose pooling are more resilient against unseen different distribution. Since the features are being pooled using

Table 6.4: Ablation studies of using pose as indexes while pooling activation from feature maps. Fusion methods bear similar meaning as Table 6.2, i.e. Fusion-1 = PPC-7 + PPC-8, Fusion-2 = PPC-7 + PPC-8 + I3DLogits and Fusion-3 = PPC-5 + PPC-7 + PPC-8 + I3DLogits.

Method	WLASL300					
	Without Pose Pooling			With Pose Pooling		
	Top-1	Top-5	Top-10	Top-1	Top-5	Top-10
PPC-7	**57.37**	76.33	82.05	57.12	70.80	74.44
PPC-8	55.59	73.08	77.55	**59.91**	75.78	78.39
I3DLogits	57.62	78.63	82.46	57.62	78.63	82.46
Fusion-1	60.12	80.05	85.25	**64.41**	78.67	82.39
Fusion-2	63.83	83.39	87.69	**67.79**	84.19	87.06
Fusion-3	64.45	83.86	87.58	**68.30**	83.19	86.22

pose localized indexes, our approach can extract better representation even from different distribution.

6.2.7 Ablation Studies

To validate the effectiveness of pose localized features, we implemented similar fusion experiments as we described in the result section, but without pose localized pooling. Instead we use traditional maximum pooling sub-sampling. In details, after extracting a feature map from any point of the network in Figure 6.2, we used maximum pooling over that feature map to produce a representation of the video. Table 6.4 outlines the results from this experiment. The result indicates, when feature is used from a single layer, pose pooling features perform equally. However, when scores from several layers are combined (any fusion case in Table 6.4), the pose pooling features provide around 4% performance gain in `top-1` accuracy. This suggests that, feature representation learned from pose pooling mechanism provides better complementary information across different layers.

6.3 Key Takeaway

In this work, we proposed a novel pose-guided pooling strategy for hand related feature extraction from a 3D ConvNet in the context of world level sign language recognition. We leveraged the body-pose information as guides to 3D feature maps. Using the body pose, we guided the 3D ConvNet to extract localized hand features. Our experiments showed that combining such features from different layers of the network can improve overall recognition accuracy. We also found these features to be better transferable than basic CNN features in modeling sign videos from a different source. In addition to that, qualitative findings revealed the effectiveness of multi-scale representation of sign videos in modeling varieties of sign classes.

Chapter 7: Improving Isolated Sign Recognition Using Representative Frames

A word-level sign video is a sequence of one or multiple articulate hand shapes performed with distinguishable arm-wrist motion. From the feature modeling perspective, isolated sign words can be categorized into two general types. One type of sign has large hand-motion while the other type mostly depends on the hand-shape patterns. The former type requires robust temporal motion modeling and the later requires the spatial pattern modeling. Often, most of the signs require both type of modeling. We observed in our previous works, the success of a sign video recognition system depends on the modeling of this joint *spatial-temporal* features. In this chapter, we investigate this fact more closely and we will show that adding a separate spatial modeling can disambiguate almost similar looking sign video classes.

Isolated sign word recognition from video is a gesture modeling problem where the gesture is performed using complex upper body motion, facial expression and occasional head movements. As discussed earlier, a combination of temporal and spatial modeling is necessary for a robust sign recognition model. We hypothesize that most of the time only one of these two modeling is enough for capturing the underlying pattern of a sign. For example, Figure 7.1 shows two such sign examples where the top sign has a distinguishing motion pattern of both hands and the bottom sign has a distinctive hand shape. It can be also noticed from the bottom example, only a single distinctive hand shape is necessary for modeling the whole sign class. In other words, we can represent the whole sign video with few similar looking hand shape frames. This observation motivated us to leverage a few sparse representative video frames to model the whole sign video.

On the other hand, the motion modeling requires dense temporal or high frame-rate

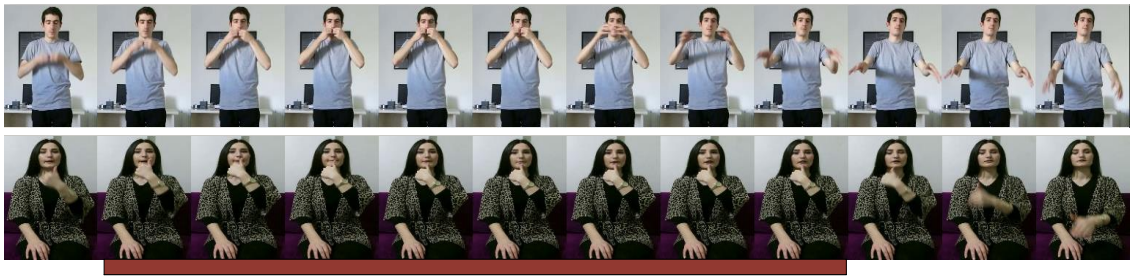


Figure 7.1: The top example shows a sign with heavy hand-motion pattern; while the bottom example shows where hand shape gives more information than hand motion. In the bottom sign, we observe, across the bottom red line, a single hand shape represents the sign.

inputs to learn hand-motion representation in a sign video. Highly complex 3D convolutional model such as I3D learns this *spatial-temporal* features simultaneously with dense RGB inputs. We hypothesize that adding another stream of representation using sparsely chosen representative frames can improve the recognition accuracy. In this regard, we proposed to add Graph Convolutional Network (GCN) and 2D CNN based sparse modeling with existing dense I3D model, in an ensemble manner. We based all of our experiments on the AUTSL dataset, a recently released word-level Turkish sign dataset [104]. Our results showed that, adding sparse feature streams to the sign recognition method adds complementary information, and improves the overall accuracy. In summary, our contribution can be listed as follows,

1. We proposed a sparse modeling method using few representative frames to disambiguate similar looking sign video classes;

2. We implemented GCN and 2D CNN based models using sparsely chosen representative frames from a sign video;

3. We demonstrated the effects of adding sparse modeling and showed comparisons with competitive approaches.

7.1 Our Approach

For the word-level sign language recognition problem, we have a dataset of N training examples $\{V_i, W_i\}$, where V_i is a RGB video $\in \Re^{T \times H \times W \times 3}$ and W_i is a word-level sign label; $H, W, 3$ are the height, width and number of channels of a single RGB video frame; and T is the video length. Two sample examples of V_i are shown in figure 7.1. In addition to RGB video inputs, our method also takes pose inputs. Body and hand poses can be seen as a time series representation of different joint location on the body and hands. For each video represented as a $V_i \in \Re^{T \times H \times W \times 3}$, we can represent the pose data as a $P_i \in \Re^{T \times J \times 2}$ where J is the number of key location in a person's body and 2 is for representing the (x, y) location of corresponding key-point on image frame space, considering the top-left corner as the origin.

To model the temporal motion representation, we use dense input frames. In other words, to model a sign video, we pick a contiguous window of video frames and form input to our models. Using this type of dense input, we train RGB based and pose based models to capture the motion component in the sign videos. On the other hand, to capture the spatial representation in a sign video, we leverage the idea of representative frames. In this case, we use a small number of selected representative frames in a sign video to model the spatial component. We train different models using dense and sparse input frames. During the test time, we use score fusion to utilize representation from both dense and sparse modeling.

7.1.1 Dense Frame Modeling

The purpose of the dense modeling is to capture the motion dynamics in a sign video. The underlying machine learning (ML) model must learn the motion representation of both hands. Since the hand motion can change abruptly, a contiguous window of input frames is necessary to better model the motion dynamics. Hence, we can pick a window of T' contiguous frames from a video $V_i \in \Re^{T \times H \times W \times 3}$. Here we assume, $T' <= T$ and

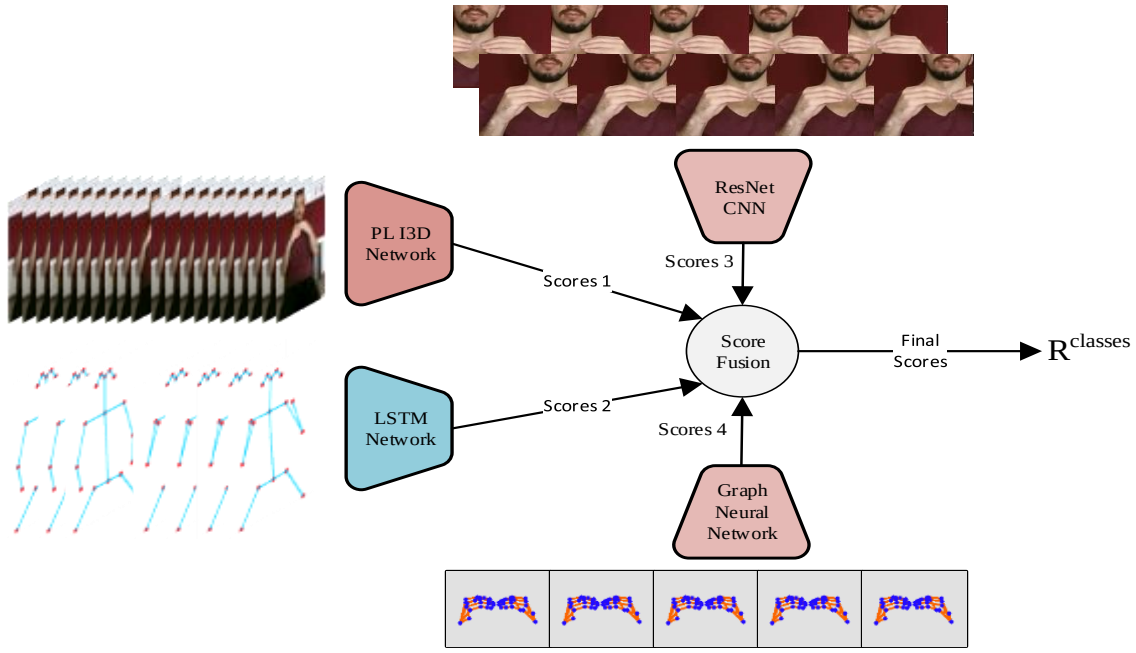


Figure 7.2: Overall fusion architecture. On the left side, two dense networks - one on raw RGB input and other on pose inputs - are shown. From the top, sparsely chosen cropped hand patches are input to image classification CNN (ResNet-34) and in the bottom hand graph are fed into the GNN based network. Scores from all of the four models are fused during test time to achieve final prediction scores over all the classes.

$T - T' < \delta$. If the δ is large, there is a chance of missing some informative part in the input video. We pick T' such that δ value only allows skipping frames from the beginning or ending region of an input video. These frames are similar for all the videos in an isolated sign recognition task and can be safely ignored. Such a δ value can be set by examining the mean and standard deviation of video lengths in the dataset.

To model the motion from such dense input streams, we utilized two types of machine learning models. The first type of model is based on 3D convolution [13]. Titled as Pose Localized I3D or PL-I3D in short, it is an enhanced version of I3D network for sign language recognition purpose [15]. Observing the better modeling and feature transfer results than basic I3D, we use pose localized PL-I3D version for modeling motion from raw RGB inputs. The PL-I3D method is described in details in Chapter 6. The second type of model works

on body pose inputs. To recap, the pose input provides joint coordinates of specific body and hand joints across the image frames in a sign video. Such pose data can be obtained from raw RGB input using off-the-self pose estimation methods [23, 50]. Since the body poses from arms and wrists are accurate high level coordinate representation of those joints, these can be an useful source for the motion modeling. We use a long short term memory (LSTM) network for modeling motion from pose data. Figure 7.2 shows our proposed ensemble architecture. We see the dense modeling input form the left side - I3D network on raw RGB and LSTM networks on body poses. These two streams, which titled as *Scores 1* and *Scores 2*, are part of the fusion architecture.

7.1.2 Sparse Frame Modeling

The purpose of sparse frame modeling is to model a sign video using a few representative frames as input. This is particularly useful because most of the isolated signs can be explained by a single representative hand-shape pattern. In some cases, there are two or more hand-shape patterns appearing in a particular order. However, even in those cases, there is always a primary hand-shape and the others work as auxiliary shapes. The bottom example in Figure 7.1 shows a sign video example that depends on only one type of hand shape across the whole video. Hence, it is natural to take that shape as a vital representation of the whole sign video.

Finding the representative frames from a sign video requires selecting the frames with clear distinguishable hand-shape patterns with less motion blurs. Since lesser blur means clearer hand-shape images, we can pick image frames based on the arm-hand motion measures in video frames. This requires a frame wise arm-hand motion estimation. However, there is a negative correlation between pose confidence and hand motion. Pose confidence is readily available from any pose estimation method described in Section 2.3. It is a measure of algorithm's confidence in estimating a pose location. High confidence pose is generated when the algorithm sees clear hand shapes in a RGB frame. On the contrary, when there is a higher motion in the hand region, the pose estimation algorithm sees blur and generates

Figure 7.3: Hand graph structures and examples. On the right side, the template hand graph from OpenPose is shown. On the left, some representative frames from random sign videos, with high pose confidence are shown.

low confidence poses. This phenomenon is described in more details in the Data Preparation section and depicted in Figure 7.4. A straightforward way to estimate motion in each frame is to calculate the overall confidence of hand poses. The higher the aggregated pose confidence the lower the motion in that frame. We followed this procedure and picked top 5 confident frames from a sign video. The experiment section details this.

Once we have 5 representative frames from a sign video, the next part is to utilize appropriate ML models for the learning task. Similar to the dense modeling case, we have two types of inputs: the raw RGB frames and the hand poses. For the RGB source, we can formulate this as an image classification problem and utilize any off-the-self 2D CNN architecture [93]. For the pose data, the hand can be seen as a graph with a certain number of nodes and links between them. Figure 7.3 shows examples of such formulation of the hand poses, following the hand-graph structure used in OpenPose [23]. This formulation uses 21 nodes and 20 links among the nodes to represent a hand. Graph Convolutional Networks (GCN) are excellent choice for modeling such graph-structured data [29]. It allows node-level feature learning on the graph while taking the graph neighborhood structure into account. The working principle of GCN is described in details in Section 2.1.3. Figure 7.2 shows the two types of sparse streams as *Scores 3* and *Scores 4*.

GCN on Hand Pose The pose key-points on whole body and hands can be seen as a graph structure. Edges of this graph are the bone connections between two neighboring joints. Figure 7.3 shows this graph for finger joints. Most of the body joints such as arms and wrist joints capture high-level motion information related to the body movements. On the contrary, the finger joints, as shown in Figure 7.3, capture the structural shape pattern of the hands. These patterns feature distinguishable finger orientation and bends using the finger pose locations. These hand-graph patterns motivate us to incorporate the structures into sign video modeling. Such graph-structured input data naturally requires a modeling where the structure is preserved. To facilitate this type of modeling, Graph Convolutional Network (GCN) was introduced [29]. Figure 7.3 shows 21 joints for a hand-graph representation and we can denote the node indices by a set $V = \{v_i \mid 0 <= v_i <= 20\}$. The graph adjacency matrix can be represented A_{ij} where $A_{ij} = 1$ if there nodes v_i and v_j are neighbors, otherwise $A_{ij} = 0$.

$$x_i^l = \sigma(W_1 x_i^{l-1} + W_2 \sum_{\forall j | A_{ij}=1} x_j^{l-1}) \tag{7.1}$$

Having this graph formulation, we can represent the graph convolutional operation using Equation 7.1 where W_1 and W_2 are the learnable model parameters. The σ represents a non-linearity operation. The x_i^l represents feature representation of i^{th} node at l^{th} layer of the GCN network. The details can be found in GraphConv method implementation in PyTorch-geometric [105].

7.2 Experiments

In this section, we provide a snapshot of the dataset we base our experiments on. Then we discuss the data preparation and training details. Finally, we demonstrate the results and the comparisons with several state-of-the-art methods. We also outline the effect of adding sparse modeling to the previous ensemble.

Table 7.1: AUTSL dataset summary.

Property	Count
Number of signs	226
Number of signers	43
Total samples	36,302
Train samples	28,142
Validation samples	4,418
Test samples	3,742
Average samples per sign	169.6
Number of different backgrounds	20
Modalities	RGB, depth, skeleton

7.2.1 Dataset

We performed all of our experiments in this chapter using AUTSL dataset: Ankara University Turkish Sign Language dataset [104]. This is an isolated word level Turkish sign gesture dataset. The dataset features various challenging background with 226 sign classes. A short summary is shown in the Table 7.1. We used only RGB modality from the dataset; we also used pose data, extracted from the RGB modality.

7.2.2 Dense Data Preparation

The median length of videos in the training set of AUTSL dataset is 61 frames [106]. We picked 64 contiguous raw RGB frames for an input to our PL-I3D network. From a T frame video, we picked a starting frame randomly from first T minus 64 ($T-64$) frames and then selected a contiguous window of 64 frames for the input. Since the initial frame is being selected randomly, this allows some temporal data augmentation. When T is less than 64, we randomly prepend the starting frame or append the ending frame to make the video a 64 frame input. Other types of data augmentation, such as random cropping and horizontal flips are also utilized while training the network.

For the motion LSTM part, we prepared input pose sequences in similar fashion except we used 32 frame inputs instead of 64. In this case, after picking a 64 contiguous window

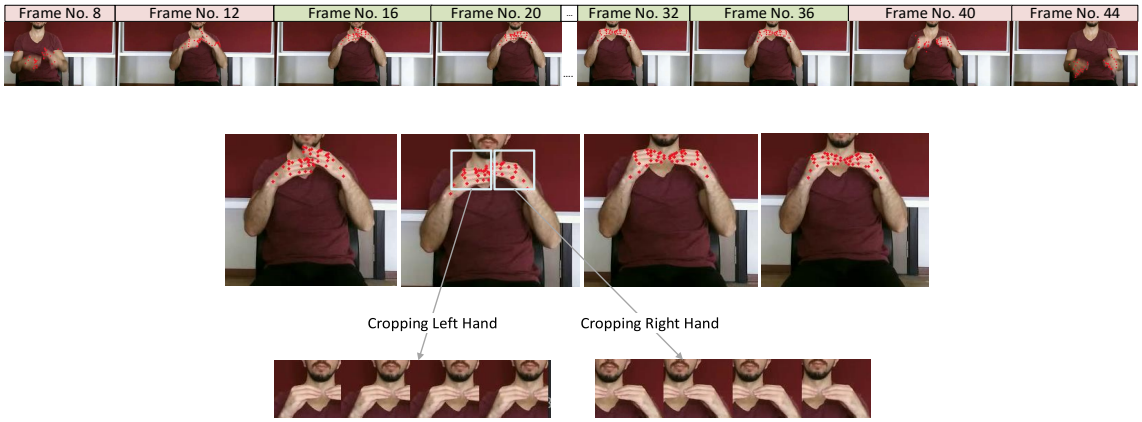


Figure 7.4: Data preparation for the sparse modeling. In the top row, 8 frames from a sample sign video is shown with pose depicted on hands. It can be seen that frame no 8, 12, 40 and 44 have blurry hand shapes that generates low confidence brittle pose. Thus, we should ignore them when selecting representative frames. The middle row displays four good frames (zoomed) based on hand pose confidence. These good pose representation (nodes' XY position) are input to the graph neural network. Bottom row shows the cropped hand patches from original frames. These are the input to ResNet-34 CNN model.

as described above, we uniformly sampled some frame inputs. We experimented with 16, 32 and 64 frame inputs. We found 32 frame input works similarly as 64 frame and used 32 during final model training.

7.2.3 Sparse Data Preparation

The purpose of the sparse modeling is to use a few representative frames from an input video. We utilized the estimated pose confidence as a measure of motion blur in the video frames and picked 5 frames with highest aggregated hand pose confidence. We did this for left and right hand separately and prepared the input for our 2D CNN classifier and Graph Convolutional Network (GCN). Figure 7.4 shows how the pose confidence relates to blurry image frames and how good frames can be chosen based on the pose confidence. To be more specific, we sorted the frames using aggregate pose confidences and picked the top 5 frames for input to the sparse modeling. However, due to less motion, beginning and ending frames of a sign video tend to have very high confidence. These frames are similar for all

sign videos in isolated signs. To prevent these frames from being selected as high confidence frames, we skipped 40% of the total frames from both ends before sorting. We assume here, informative part of a sign lies in the middle part of a video. This assumption makes sense in case of isolated sign videos, because there is an arm lifting and resting motion in the beginning and end of any isolated sign. Also, our motivation of sparse modeling was to use few frames to capture spatial pattern. Hence, skipping some frames from both ends should not interfere with the learning process.

7.2.4 Training Details

For the PL-I3D model, we followed similar data preparation and batch sizes mentioned in Section 6.2.3. For the dense pose-input LSTM models, we used a 2 layer bi-directional network and used the concatenated forward and backward final states as a sign sample representation. The state size and input embedding layer were of size 256 and 128 respectively. We also noticed, with 3 or 4 layer networks there is no significant performance gain. For the GCN model, we used a 2 layer network with initial embedding layer of 128 and with channel size of 256 in both layer. For input at each node, we used (x, y) position coordinates information on the image frame and the unit vector representation of each node. In total, this gives each node a 4 sized vector input. We pool 128 sized feature vector from each of the node to represent the graph input of hands. For a total of 21 hand joints at each hand, this yields a 2688 sized representation. For image CNN model, we take 100×100 hand crops and resize them to 224×224 with occasional horizontal-flip data augmentation. All models were trained using cross entropy loss described in the Equation 4.1. We used Adam optimizer for optimizing our models using an initial learning rate of 0.001 [86]. Due to 64 frame RGB input, we could not use batch size of more than 6 for the PL-I3D model due to GPU memory. For other models, we used batch sizes of 8 and 16. We developed our model using PyTorch and PyTorch-geometric and trained our model on Nvidia GeForce RTX 2080Ti GPU.

Table 7.2: Sign recognition accuracy on the AUTSL test split for different models in separation and fusion. First four rows shows performance on four models - two densely input and two sparse input models. The following rows shows different combination of these separate methods.

Model ID	Model Name	Model Description	Accuracy
M1	PL-I3D	Pose localized I3D	92.03%
M2	Motion-LSTM	LSTM network on body poses	86.45%
M3	Patch-CNN	ResNet34 on cropped hand patches	63.65%
M4	Hand-GNN	Graph Neural Network on hand poses	54.50%
Fusion1	M1 + M2	Dense input fusion	93.14%
Fusion2	M3 + M4	Sparse input fusion	69.26%
Fusion3	M2 + M4	Pose based fusion	89.17%
Fusion4	M1 + M3	RGB based fusion	92.94%
Fusion5	M1 + M2 + M3 + M4	All fusion	**95.83%**

7.2.5 Evaluation Results

Table 7.2 shows the results on test split of AUTSL dataset. First four rows show the recognition accuracy from each of our implemented models. The following rows show results on different fusion combination. The result indicates that PL-I3D works best as a single source classifier. This is expected because of powerful modeling capacity and the hand related feature extraction mechanism of PL-I3D network. However, we can observe that adding other sources improves the accuracy. For example, fusion of motion LSTM and PL-I3D, identified as Fusion1, improves accuracy by 1.11%. This can be seen as the overall performance from dense input models. On the other hand, the sparse input fusion, using only 5 representative frames, identified as Fusion2, yields 69.26%. Although this performance is far behind the dense fusion, it contributes to the overall performance when we use all the representation sources. It improves the overall performance by 2.70%, which converts to 102 test videos. In other words, adding 5 frame sparse modeling can disambiguate 102 test videos that were incorrectly predicted without sparse modeling. Another interesting result is Fusion3 or pose based fusion which is a combination of pose based motion LSTM and sparse input GCN.

Table 7.3: Comparison with different methods.

Method	Representation Used	Accuracy
SAM-SLR	Joint keypoints, bone, bone-motion, joint-motion, RGB, optical flow	98.42%
USTC-SLR	Not available	97.62%
wenbinwuee	Not available	96.55%
Baseline [104]	RGB	49.22%
VTN-PF [106]	Joint keypoints, RGB	92.92%
Gruber *et al.* [107]	Joint keypoints, RGB	95.46%
Enriquez *et al.* [108]	Joint keypoints, bone, bone-motion, joint-motion, RGB	96.15%
Ours	Joint keypoints, RGB	95.83%

This fusion has quite a less number of model parameters, 1.3M vs 12.06M, compared to PL-I3D version and yet it performs with comparable accuracy. This type of faster and lighter model could be useful for collecting isolated signs from long continuous sign repositories. This also indicates the potential of pose based sign video modeling with lighter networks.

7.2.6 Comparisons

In this section, we compare our results with several state-of-the-art methods. Although the AUTSL [104] dataset was released in the year of 2020, most of the recent works were introduced in a competition featured in CVPR'21 workshop [109]. Table 7.3 details the comparison among different methods and our proposed method. First three rows show the results, taken from the competition leaderboard, obtained by top 3 teams. The top performing team proposed an ensemble based method using whole body pose keypoints, pose features and 3d CNN on RGB and optical flow streams [110]. For joint keypoint data, the authors used four input streams: joint location, bone, joint motion and bone motion. Compared to this process, our motion LSTM used only the location information. As shown in table 7.3, the details of the second and third places in the competition are not available. The authors in [106], as shown in the 5^{th} row, proposed a multi-modal approach, based on Video Transformer Network (VTN) [111], using joint keypoint and RGB inputs. Our

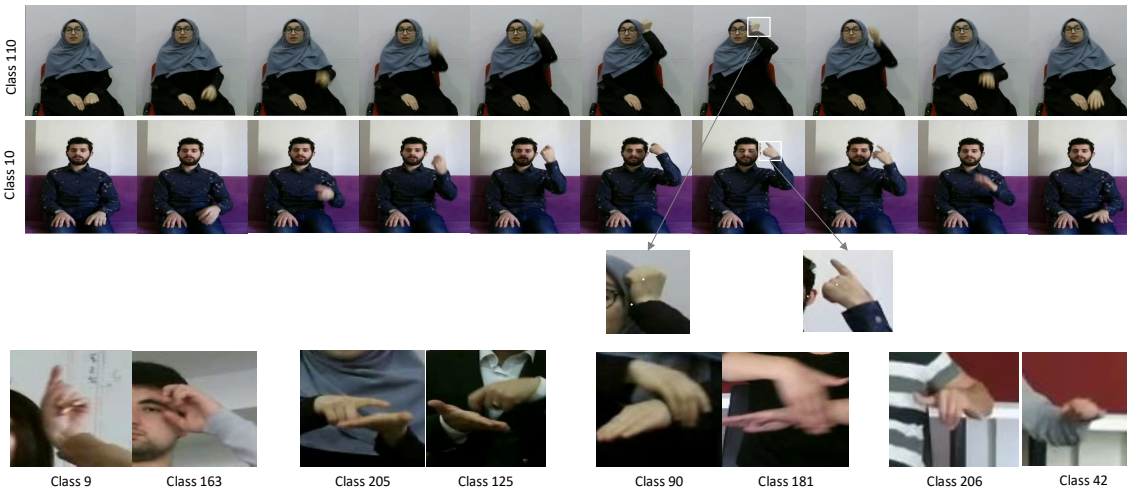


Figure 7.5: Similar looking sign video examples without sparse modeling. The top two rows shows examples from two most confused classes without sparse modeling: $class_{10}$ and $class_{110}$. The bottom rows shows some similarly confused class pairs and for each case, the distinguishing hand crops between the members of each pair.

proposed ensemble method outperforms this work by 3%. The authors in [107], as shown in the 6^{th} row, used multiple I3D networks on RGB and mask data, and transformer [42] based approach on joint keypoints data. Input representation wise, this is the most similar ensemble to our proposed approach. They used an ensemble score of 17 networks: 13 I3D networks on RGB and 4 transformers on key-points. In comparison, our ensemble method is composed of 3 I3D networks, one 2 layer LSTM and one 2 layer graph neural network. This shows that even with less modeling resources, our method still outperforms this by a margin of 0.5%. The authors in [108] used four input representation: joint keypoints, bone, joint-motion and bone-motion input on MS-G3D network [112]. On raw RGB, due to reduced parameter count, they preferred the S3D network [113] over the I3D. However, our PL-I3D version outperforms this S3D by 1.75%, i.e our only PL-I3D achieves 92.03% while only S3D, as reported in [108], achieves 90.27%. Overall ensemble in [108] outperforms our method by 0.28%. However, in motion capturing LSTM we used only joint key-points stream while this comparing method used joint key-points, bone, bone motion, and joint motion.

7.2.7 Effect of Sparse Modeling

Our initial motivation for adding sparse modeling, i.e model a sign video using 5 representative frames, was to disambiguate classes where signs have similar hand motion with minute hand-shape differences. In the result section, we observed that 102 more test videos were correctly classified in the presence of sparse modeling. A few example cases like this are shown in Figure 7.5. The top two rows show sign samples from two classes: $class_{10}$ and $class_{110}$. Without sparse modeling, 6 of the samples of $class_{10}$ were confused with $class_{110}$ during inference. However, when sparse modeling is added, 5 of the samples from $class_{10}$ were correctly predicted. This disambiguation can be explained by looking at the two classes more closely. We can see from the cropped hand patches in the middle of Figure 7.5 that, the two sign classes only differs by a single frame. In addition to that, the difference in hand shape is very subtle. Similar hand motion can be seen when looking at both of the signs from left to right at the same time. Even though PL-I3D is based on a powerful video recognition CNN architecture, it faces problem to model such tiny static variation in a frame. This is due to the fact that, I3D is designed to model spatial-temporal dynamics simultaneously. However, when a separate sparse modeling is added, the final ensemble finds it easier to distinguish such cases. Some similar cases are also shown in the bottom of Figure 7.5, where only the distinguishable hand crops are shown. For each of these class pairs, adding sparse modeling removed misclassification to some extent.

7.3 Key Takeaway

We proposed an ensemble based methods for isolated sign word recognition from videos. Our ensemble consists of dense input 3D convolution models and sparse input 2D CNN and graph convolution models. Dense modeling uses 64 frame input windows while sparse modeling uses 5 representative input frames from a sign video. In both cases, models are trained on two types of input representation: the raw RGB frames and the pose data. Added sparse feature streams improves the sign classification capacity of the ensemble model. We

verified the usefulness of sparse modeling in clearing the confusions of dense modeling. Our experiments showed that adding a 5 frame sparse modeling for each video can disambiguate sign videos, which are misclassified otherwise.

Chapter 8: Conclusions and Future Directions

In this dissertation, we have tackled different challenges in video-based word level sign language recognition. More specifically, we have worked towards building an ASL dataset, modeling hand-shape features, and utilizing pose data in various ways. In Chapter 3, we described the GMU-ASL51 dataset and proposed sign classification methods. We demonstrated comparisons among these methods and showed the effectiveness of adding RGB to the modeling. Top performing method utilized spatial data augmentation and achieved better accuracy than the baselines. The useful effect of RGB modeling on sign classes motivated us to explore this direction in more detail.

In Chapter 4 and 5, we demonstrated our contributions on generating RGB based hand-shape representation for sign video recognition. We started with using the representation capacity of a CNN, pre-trained on a related dataset from a different source. Although trained on a different source, this representation yielded 8% improvement on the downstream sign recognition task. To further utilize the hand-shape representation, we proposed a hand-shape learning technique. As opposed to outside sources, this approach learns the shape representation particularly for the sign videos we want to model. Being modeled for the specific dataset, this representation obtained 11% better results than the previous best. We also presented qualitative comparisons among methods based on different hand-shape representations. We detailed this method in Chapter 5.

Chapter 6 described our contribution in localizing essential hand features in the context of sign gesture videos. We utilized the pose information to guide feature lookup in 3D feature maps. Extracted multi-scale localized features improved sign modeling task by a margin of 12%. These features were also better transferable to related data domains. In Chapter 7, we illustrated the usefulness of sparse frame modeling. We proposed the idea of using a few representative frames to learn a spatial representation of a sign video. In

addition to the RGB source, we also used a graph based modeling on the pose data, to account for the different finger orientation. When added with existing *spatial-temporal* representations, the added sparse source improved the recognition performance. We also outlined the effectiveness of the added source in modeling subtle differences among similar-looking sign classes.

8.1 Future Works

Supervised learning methods are heavily dependent on the quality of the training data. Existing state-of-the-art benchmark sign-language datasets are collected either in restricted environments or from different online sources. Online videos are often purposed for different tasks such as tutoring or describing an event. Hence, using these in a machine learning task is sometimes not optimal. There is simply not enough modeling information, either in RGB or poses, to learn from. Hence, producing good training data is always intriguing. We believe that the sign recognition problem can be solved solely using fine-grained 3D representation of finger poses. To achieve such quality pose information, high-end data collection system is required. Also the data from online sources contain temporal noise, i.e., actual sign video is a small segment of the whole input video. This puts additional difficulties on modeling techniques, because traditional methods consider the whole video as a sample of the corresponding ground truth class. If extra temporal boundary-level annotation can be obtained, the models could be trained in a more informed way.

To solve the complete sign language recognition task, the next step from world level videos is to tackle continuous sentence-level videos. These videos are longer and in addition to word level signs, they contain finger-spelling segments. The sentence-level sign problem itself should tackle several challenging sub-problems. First, detecting the boundaries of a segment, either isolated word or finger-spelling segments, is an interesting problem. A machine learning method must identify the boundary frames by looking at a context window. Hand motion data (i.e. the motion coherence of hands) can be used to detect such frames. Because, transition from a segment to another sometimes breaks the motion consistency.

Once such boundary problem is solved, one could focus on modeling the individual segments. The isolated word sign modeling, the problem we explored in this dissertation, is a well-studied area than finger-spelling transcription. Translating finger-spelling sequence itself is a sequence to sequence modeling task. In addition to the challenges described in this thesis, the finger-spelling task requires proper alignment learning between the input video frames and the output letter sequence.

The continuous modeling in American Sign Language is in a very early stage. In my opinion, a major limitation is the lacking of a large scale continuous ASL dataset. Nowadays, it's very common to watch a television report with an ASL interpreter. These videos are boundless sources for obtaining continuous data. However, the voice or the subtitles only provide weak supervision to the data. Since there is no sequence correspondence between continuous sign video and textual English sentence, one must work towards the alignment annotation. Upon the availability of a large-scale continuous sign-language dataset, with proper ground truth annotation, deep-learning based models can be trained to achieve better performance and generalization capacity in Sign Language Recognition.

Bibliography

Bibliography

[1] D. Li, C. Rodriguez, X. Yu, and H. Li, "Word-level Deep Sign Language Recognition from Video: A New Large-scale Dataset and Methods Comparison," in *The IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 1459–1469.

[2] http://wfdeaf.org/, "World federation of deaf: Our work," 2020.

[3] https://en.wikipedia.org/wiki/List_of_sign_languages.

[4] Olson and Kemery, "2019 voice report: Consumer adoption of voice technology and digital assistants," 2019. [Online]. Available: https://about.ads.microsoft.com/en-us/insights/2019-voice-report

[5] H. Vaezi Joze and O. Koller, "MS-ASL: A Large-Scale Data Set and Benchmark for Understanding American Sign Language," in *The British Machine Vision Conference (BMVC)*, September 2019. [Online]. Available: https://www.microsoft.com/en-us/research/publication/ms-asl-a-large-scale-data-set-and-benchmark-for-understanding-american-sign-language/

[6] J. Huang, W. Zhou, H. Li, and W. Li, "Sign language recognition using 3d convolutional neural networks," in *2015 IEEE International Conference on Multimedia and Expo (ICME)*, June 2015, pp. 1–6.

[7] V. Athitsos, C. Neidle, S. Sclaroff, J. Nash, A. Stefan, , and A. Thangali, "The American sign language lexicon video dataset," in *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, June 2008, pp. 1–8.

[8] Z. Zafrulla, H. Brashear, P. Yin, P. Presti, T. Starner, and H. Hamilton, "American sign language phrase verification in an educational game for deaf children," in *2010 20th International Conference on Pattern Recognition*, Aug 2010, pp. 3846–3849.

[9] H. Cooper, E.-J. Ong, N. Pugeault, and R. Bowden, "Sign language recognition using sub-units, booktitle=gesture recognition," S. Escalera, I. Guyon, and V. Athitsos, Eds. Cham: Springer International Publishing, 2017, pp. 89–118.

[10] A. A. Hosain, P. S. Santhalingam, P. Pathak, J. Kosecka, and H. Rangwala, "Sign language recognition analysis using multimodal data," in *International Conference on Data Science and Advanced Analytics (DSAA'19)*, 2019.

[11] V. Athitsos, C. Neidle, S. Sclaroff, J. Nash, A. Stefan, , and A. Thangali, "The American Sign Language Lexicon Video Dataset," in *2008 IEEE Computer Society*

Conference on Computer Vision and Pattern Recognition Workshops, June 2008, pp. 1–8.

[12] O. Koller, H. Ney, and R. Bowden, "Deep hand: How to train a cnn on 1 million hand images when your data is continuous and weakly labelled," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[13] J. Carreira and A. Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset," 2017. [Online]. Available: http://arxiv.org/abs/1705.07750

[14] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 221–231, Jan. 2013. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2012.59

[15] A. A. Hosain, P. S. Santhalingam, P. Pathak, H. Rangwala, and J. Kosecka, "Hand pose guided 3d pooling for word-level sign language recognition," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2021, pp. 3429–3439.

[16] M. M. Zaki and S. I. Shaheen, "Sign language recognition using a combination of new vision based features, journal = pattern recognition letters." [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016786551000379X

[17] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning Spatiotemporal Features With 3D Convolutional Networks," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[18] A. Shahroudy, J. Liu, T. T. Ng, and G. Wang, "Ntu rgb+d: A large scale dataset for 3d human activity analysis," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 1010–1019.

[19] A. Boukhayma, R. d. Bem, and P. H. Torr, "3d hand shape and pose from images in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 843–10 852.

[20] P. Panteleris, I. Oikonomidis, and A. A. Argyros, "Using a single RGB frame for real time 3d hand pose estimation in the wild," *CoRR*, vol. abs/1712.03866, 2017. [Online]. Available: http://arxiv.org/abs/1712.03866

[21] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. V. Gehler, and B. Schiele, "Deepcut: Joint subset partition and labeling for multi person pose estimation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[22] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele, "Deepercut: A deeper, stronger, and faster multi-person pose estimation model," 2016.

[23] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields," in *arXiv preprint arXiv:1812.08008*, 2018.

[24] Y. Xiu, J. Li, H. Wang, Y. Fang, and C. Lu, "Pose Flow: Efficient online pose tracking," in *BMVC*, 2018.

[25] D. Xing, X. Wang, and H. Lu, "Action recognition using hybrid feature descriptor and VLAD video encoding," in *Computer Vision - ACCV 2014 Workshops - Singapore, Singapore, November 1-2, 2014, Revised Selected Papers, Part I*, ser. Lecture Notes in Computer Science, C. V. Jawahar and S. Shan, Eds., vol. 9008. Springer, 2014, pp. 99–112. [Online]. Available: https://doi.org/10.1007/978-3-319-16628-5_8

[26] P. Jangyodsuk, C. Conly, and V. Athitsos, "Sign language recognition using dynamic time warping and hand shape distance based on histogram of oriented gradient features," in *Proceedings of the 7th International Conference on PErvasive Technologies Related to Assistive Environments*, ser. PETRA '14. New York, NY, USA: Association for Computing Machinery, 2014. [Online]. Available: https://doi.org/10.1145/2674396.2674421

[27] Z. Zafrulla, H. Brashear, T. Starner, H. Hamilton, and P. Presti, "American sign language recognition with the kinect," in *Proceedings of the 13th International Conference on Multimodal Interfaces*, ser. ICMI '11. New York, NY, USA: ACM, 2011, pp. 279–286. [Online]. Available: http://doi.acm.org/10.1145/2070481.2070532

[28] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning Representations by Back-Propagating Errors*. Cambridge, MA, USA: MIT Press, 1988, p. 696–699.

[29] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *CoRR*, vol. abs/1609.02907, 2016. [Online]. Available: http://arxiv.org/abs/1609.02907

[30] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *CoRR*, vol. abs/1606.09375, 2016. [Online]. Available: http://arxiv.org/abs/1606.09375

[31] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "Cayleynets: Graph convolutional neural networks with complex rational spectral filters," *CoRR*, vol. abs/1705.07664, 2017. [Online]. Available: http://arxiv.org/abs/1705.07664

[32] Z. C. Lipton, "A critical review of recurrent neural networks for sequence learning," *CoRR*, vol. abs/1506.00019, 2015. [Online]. Available: http://arxiv.org/abs/1506.00019

[33] Y. Bengio, P. Frasconi, and P. Simard, "The problem of learning long-term dependencies in recurrent networks," in *IEEE International Conference on Neural Networks*, March 1993, pp. 1183–1188 vol.3.

[34] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *Trans. Neur. Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994. [Online]. Available: http://dx.doi.org/10.1109/72.279181

[35] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[36] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *CoRR*, vol. abs/1409.1259, 2014. [Online]. Available: http://arxiv.org/abs/1409.1259

[37] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014. [Online]. Available: http://arxiv.org/abs/1412.3555

[38] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, p. 2342–2350.

[39] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *CoRR*, vol. abs/1503.04069, 2015. [Online]. Available: http://arxiv.org/abs/1503.04069

[40] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[41] M. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *CoRR*, vol. abs/1508.04025, 2015. [Online]. Available: http://arxiv.org/abs/1508.04025

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[43] Z. Zhang, "Microsoft kinect sensor and its effect," *IEEE MultiMedia*, vol. 19, no. 2, pp. 4–10, Apr. 2012. [Online]. Available: http://dx.doi.org/10.1109/MMUL.2012.24

[44] L. Keselman, J. I. Woodfill, A. Grunnet-Jepsen, and A. Bhowmik, "Intel realsense stereoscopic depth cameras," *CoRR*, vol. abs/1705.05548, 2017. [Online]. Available: http://arxiv.org/abs/1705.05548

[45] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2d human pose estimation: New benchmark and state of the art analysis," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3686–3693.

[46] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[47] G. Papandreou, T. Zhu, N. Kanazawa, A. Toshev, J. Tompson, C. Bregler, and K. Murphy, "Towards accurate multi-person pose estimation in the wild," 2017.

[48] G. Gkioxari, B. Hariharan, R. Girshick, and J. Malik, "Using k-poselets for detecting people and localizing their keypoints," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, ser. CVPR '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 3582–3589. [Online]. Available: https://doi.org/10.1109/CVPR.2014.458

[49] U. Iqbal and J. Gall, "Multi-person pose estimation with local joint-to-person associations," *CoRR*, vol. abs/1608.08526, 2016. [Online]. Available: http://arxiv.org/abs/1608.08526

[50] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *CVPR*, 2016.

[51] K. Simonyan and A. Zisserman, "Two-Stream Convolutional Networks for Action Recognition in Videos," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 568–576. [Online]. Available: http://papers.nips.cc/paper/5353-two-stream-convolutional-networks-for-action-recognition-in-videos.pdf

[52] P. Narayana, J. R. Beveridge, and B. A. Draper, "Gesture recognition: Focus on the hands," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2018, pp. 5235–5244.

[53] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Val Gool, "Temporal Segment Networks: Towards Good Practices for Deep Action Recognition," in *ECCV*, 2016.

[54] K. Liu, W. Liu, C. Gan, M. Tan, and H. Ma, "T-C3D: Temporal Convolutional 3D Network for Real-Time Action Recognition," 2018. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17205

[55] H. Xu, A. Das, and K. Saenko, "R-C3D: Region Convolutional 3D Network for Temporal Activity Detection," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[56] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, and A. Zisserman, "Temporal Cycle-Consistency Learning," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[57] I. Misra, C. L. Zitnick, and M. Hebert, "Shuffle and Learn: Unsupervised Learning Using Temporal Order Verification," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 527–544.

[58] J. Liu, A. Shahroudy, D. Xu, A. K. Chichung, and G. Wang, "Skeleton-based action recognition using spatio-temporal lstm network with trust gates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2017.

[59] Y. Du, W. Wang, and L. Wang, "Hierarchical recurrent neural network for skeleton based action recognition," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 1110–1118.

[60] V. Veeriah, N. Z, and G. J. Qi, "Differential recurrent neural networks for action recognition," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 4041–4049.

[61] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu, "An end-to-end spatio-temporal attention model for human action recognition from skeleton data," in *AAAI Conference on Artificial Intelligence*, 2017, pp. 4263–4270.

[62] J. Liu, G. Wang, L. Y. Duan, K. Abdiyeva, and A. C. Kot, "Skeleton-based human action recognition with global context-aware attention lstm networks," *IEEE Transactions on Image Processing*, vol. 27, no. 4, pp. 1586–1599, April 2018.

[63] Q. Ke, M. Bennamoun, S. An, F. A. Sohel, and F. Boussaïd, "A new representation of skeleton sequences for 3d action recognition," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017, pp. 4570–4579. [Online]. Available: https://doi.org/10.1109/CVPR.2017.486

[64] J. Liu, N. Akhtar, and A. Mian, "Skepxels: Spatio-temporal image representation of human skeleton joints for action recognition," *CoRR*, vol. abs/1711.05941, 2017. [Online]. Available: http://arxiv.org/abs/1711.05941

[65] L. Wang, Y. Qiao, and X. Tang, "Action Recognition with Trajectory-Pooled Deep-Convolutional Descriptors," *CoRR*, vol. abs/1505.04868, 2015. [Online]. Available: http://arxiv.org/abs/1505.04868

[66] C. Cao, Y. Zhang, C. Zhang, and H. Lu, "Action recognition with joints-pooled 3d deep convolutional descriptors," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, ser. IJCAI'16. AAAI Press, 2016, p. 3324–3330.

[67] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *CoRR*, vol. abs/1506.05163, 2015. [Online]. Available: http://arxiv.org/abs/1506.05163

[68] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and leman go neural: Higher-order graph neural networks," *CoRR*, vol. abs/1810.02244, 2018. [Online]. Available: http://arxiv.org/abs/1810.02244

[69] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, "Structural-rnn: Deep learning on spatio-temporal graphs," *CoRR*, vol. abs/1511.05298, 2015. [Online]. Available: http://arxiv.org/abs/1511.05298

[70] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," *CoRR*, vol. abs/1801.07455, 2018. [Online]. Available: http://arxiv.org/abs/1801.07455

[71] L. Pigou, S. Dieleman, P.-J. Kindermans, and B. Schrauwen, "Sign language recognition using convolutional neural networks," in *Computer Vision - ECCV 2014 Workshops*, L. Agapito, M. M. Bronstein, and C. Rother, Eds. Cham: Springer International Publishing, 2015, pp. 572–578.

[72] C. Sun, T. Zhang, and C. Xu, "Latent support vector machine modeling for sign language recognition with kinect," *ACM Trans. Intell. Syst. Technol.*, vol. 6, no. 2, pp. 20:1–20:20, Mar. 2015. [Online]. Available: http://doi.acm.org/10.1145/2629481

[73] S. Bambach, S. Lee, D. Crandall, and C. Yu, "Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions," in *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[74] A. Mittal, A. Zisserman, and P. H. S. Torr, "Hand detection using multiple proposals," in *British Machine Vision Conference*, 2011.

[75] J. Forster, C. Schmidt, O. Koller, M. Bellgardt, and H. Ney, "Extensions of the sign language recognition and translation corpus RWTH-PHOENIX-weather," in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, 2014, pp. 1911–1916.

[76] R. Cui, H. Liu, and C. Zhang, "Recurrent Convolutional Neural Networks for Continuous Sign Language Recognition by Staged Optimization," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[77] J. Pu, W. Zhou, and H. Li, "Dilated Convolutional Network with Iterative Optimization for Continuous Sign Language Recognition," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 7 2018, pp. 885–891. [Online]. Available: https://doi.org/10.24963/ijcai.2018/123

[78] D. Guo, S. Wang, Q. Tian, and M. Wang, "Dense Temporal Convolution Network for Sign Language Translation," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 7 2019, pp. 744–750. [Online]. Available: https://doi.org/10.24963/ijcai.2019/105

[79] D. Guo, S. Tang, and M. Wang, "Connectionist Temporal Modeling of Video and Language: A Joint Model for Translation and Sign Labeling," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 7 2019, pp. 751–757. [Online]. Available: https://doi.org/10.24963/ijcai.2019/106

[80] J. Pu, W. Zhou, and H. Li, "Iterative Alignment Network for Continuous Sign Language Recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[81] D. Metaxas, M. Dilsizian, and C. Neidle, "Scalable ASL sign recognition using model-based machine learning and linguistically annotated corpora," 2018. [Online]. Available: https://open.bu.edu/handle/2144/30049

[82] D. Metaxas, M. Dilsizian, and M. Carol, "Linguistically-driven Framework for Computationally Efficient and Scalable Sign Recognition," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA), May 2018. [Online]. Available: https://www.aclweb.org/anthology/L18-1271

[83] A. Thangali, J. P. Nash, S. Sclaroff, and C. Neidle, "Exploiting phonological constraints for hand shape inference in ASL video," in *CVPR 2011*, June 2011, pp. 521–528.

[84] S. Kulkarni, "Appearance based recognition of american sign language using gesture segmentation."

[85] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[86] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[87] D. Bragg, O. Koller, M. Bellard, L. Berke, P. Boudreault, A. Braffort, N. Caselli, M. Huenerfauth, H. Kacorri, T. Verhoef, C. Vogler, and M. Ringel Morris, "Sign language recognition, generation, and translation: An interdisciplinary perspective," in *The 21st International ACM SIGACCESS Conference on Computers and Accessibility*, ser. ASSETS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 16–31. [Online]. Available: https://doi.org/10.1145/3308561.3353774

[88] T. Starner and A. Pentland, "Real-time american sign language recognition from video using hidden markov models," in *Proceedings of International Symposium on Computer Vision - ISCV*, Nov 1995, pp. 265–270.

[89] M. Tapia and E. , "Using machine learning for real-time activity recognition and estimation of energy expenditure," 01 2008.

[90] Q. Miao, Y. Li, W. Ouyang, Z. Ma, X. Xu, W. Shi, and X. Cao, "Multimodal gesture recognition based on the resc3d network," in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, Oct 2017, pp. 3047–3055.

[91] H. Wang, P. Wang, Z. Song, and W. Li, "Large-scale multimodal gesture recognition using heterogeneous networks," in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, Oct 2017, pp. 3129–3137.

[92] L. Zhang, G. Zhu, P. Shen, and J. Song, "Learning spatiotemporal features using 3dcnn and convolutional lstm for gesture recognition," in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, Oct 2017, pp. 3120–3128.

[93] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[94] J. Wan, S. Z. Li, Y. Zhao, S. Zhou, I. Guyon, and S. Escalera, "Chalearn looking at people rgb-d isolated and continuous datasets for gesture recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, June 2016, pp. 761–769.

[95] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta, "Hollywood in homes: Crowdsourcing data collection for activity understanding," *CoRR*, vol. abs/1604.01753, 2016. [Online]. Available: http://arxiv.org/abs/1604.01753

[96] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-Term Recurrent Convolutional Networks for Visual Recognition and Description," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[97] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond Short Snippets: Deep Networks for Video Classification," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[98] G. Varol, I. Laptev, and C. Schmid, "Long-Term Temporal Convolutions for Action Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1510–1517, 2018.

[99] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler, "Convolutional Learning of Spatio-Temporal Features," in *Proceedings of the 11th European Conference on Computer Vision: Part VI*, ser. ECCV'10. Berlin, Heidelberg: Springer-Verlag, 2010, p. 140–153.

[100] G. Chéron, I. Laptev, and C. Schmid, "P-cnn: Pose-based cnn features for action recognition," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 3218–3226.

[101] A. Hosain, P. Santhalingam, P. Pathak, H. Rangwala, and J. Kosecka, "Finehand: Learning hand shapes for american sign language recognition," in *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020) (FG)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2020, pp. 397–404. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/FG47880.2020.00062

[102] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[103] S. Arora, A. Bhaskara, R. Ge, and T. Ma, "Provable bounds for learning some deep representations," in *Proceedings of the 31st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, E. P. Xing and T. Jebara, Eds., vol. 32, no. 1. Bejing, China: PMLR, 22–24 Jun 2014, pp. 584–592. [Online]. Available: http://proceedings.mlr.press/v32/arora14.html

[104] O. M. Sincan and H. Y. Keles, "AUTSL: A large scale multi-modal turkish sign language dataset and baseline methods," *CoRR*, vol. abs/2008.00932, 2020. [Online]. Available: https://arxiv.org/abs/2008.00932

[105] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[106] M. De Coster, M. Van Herreweghe, and J. Dambre, "Isolated sign recognition from rgb video using pose flow and self-attention," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2021, pp. 3441–3450.

[107] I. Gruber, Z. Krnoul, M. Hruz, J. Kanis, and M. Bohacek, "Mutual support of data modalities in the task of sign language recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2021, pp. 3424–3433.

[108] M. Vazquez-Enriquez, J. L. Alba-Castro, L. Docio-Fernandez, and E. Rodriguez-Banga, "Isolated sign language recognition with multi-scale spatial-temporal graph convolutional networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2021, pp. 3462–3471.

[109] O. M. Sincan, J. C. S. J. Junior, S. Escalera, and H. Y. Keles, "Chalearn lap large scale signer independent isolated sign language recognition challenge: Design, results and future research," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2021, pp. 3472–3481.

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

[111] D. Neimark, O. Bar, M. Zohar, and D. Asselmann, "Video transformer network," *CoRR*, vol. abs/2102.00719, 2021. [Online]. Available: https://arxiv.org/abs/2102.00719

[112] Z. Liu, H. Zhang, Z. Chen, Z. Wang, and W. Ouyang, "Disentangling and unifying graph convolutions for skeleton-based action recognition," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[113] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy, "Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

Curriculum Vitae

Al Amin Hosain is a Ph.D. candidate in the Computer Science Department at George Mason University (GMU). In December, 2012, he obtained his Bachelor of Science (Bs.C.) from Chittagong University of Engineering and Technology (CUET), Bangladesh. Until the Fall, 2015, he worked as a Software Engineer at Samsung Research Institute, Bangladesh. He started working as a Graduate Teaching Assistant at GMU CS department in Fall, 2015. He also worked as a Research Assistant under the supervision of Dr. Huzefa Rangwala and Dr. Jana Kosecka. His primary research area is the application of Deep Learning in the context of Sign Language Understanding from video. He has general expertise in Machine Learning, Data Mining, Deep Learning and Video Understanding.