YOU ONLY GESTURE ONCE (YOUGO): AMERICAN SIGN LANGUAGE TRANSLATION USING YOLOV3

by

Mehul Nanda

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science

Department of Computer and Information Technology

West Lafayette, Indiana

May 2020

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Dmitri Gusev, Chair

Department of Computer and Information Technology

Dr. John Springer

Department of Computer and Information Technology

Dr. Chad Laux

Department of Computer and Information Technology

Approved by:

Dr. Eric Mattson

Dedicated to my mom, family and friends.

Special thank you to my friend Karan Saxena.

ACKNOWLEDGMENTS

I gratefully acknowledge my graduate advisor, Dr. Dmitri Gusev for his continuous support and guidance, my thesis committee consisting of Dr. John Springer and Dr. Chad Laux for their insightful comments and reviews.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ASL: American Sign Language

CNN: Convolutional Neural Network

HMM: Hidden Markov Models

IOU: Intersection Over Union

mAP: Mean Average Precision

RCNN: Region Convolutional Neural Network

YOLO: You Only Look Once

ABSTRACT

Sign Language is a medium for communication used primarily by people who are either deaf or mute. People use it to communicate their thoughts, ideas, etc. to the world. Sign language has a defined vocabulary, grammar and associated lexicons. There are different types for sign language based on geography and context of spoken language such as American Sign Language, British Sign Language, Japanese Sign Language, etc. The emphasis of this research is on American Sign Language (ASL).

Communication through sign language can be orchestrated in a variety of ways. There are certain words of the spoken language that can be directly represented and interpreted through simple gestures. Words like Hello, Mom, Dad, etc. have designated signs or gestures. However, there are certain words that don't have pre-defined signs. In this case, a technique called "Fingerspelling" is used to spell the word out using sings for individual alphabets. Typically, fingerspelling is used when someone is trying to convey their name.

Research in the field of sign language interpretation and translation had been sparse prior to the introduction of deep learning methods and algorithms. The most common technique for interpretation is using Image Processing Algorithms to extract features from orchestrated gestures and then using Convolutional Neural Networks to learn these features and increase utility. Advances in deep learning have led to the creation of Object Detection Algorithms that, when used in conjunction with neural networks, can identify all types of objects. Currently, there is research being conducted to identify words from the sign language vocabulary by classifying them as objects and making use of such object detection plus neural network combinations. You Only Look

Once (YOLO) is one such algorithm that excels in identifying custom objects. It is used in conjunction with a neural network architecture known as Darknet.

People that make use of sign language often need to rely on a translator to convey what they are trying to say to a person that does not understand sign language. Dependency on a translator can create issues and potentially render the person incapable of acting independently. The creation of a system that can help people use sign language without depending on another person can really help them be independent and ignite the confidence to present themselves to the world without any fear.

The focus of this research is to propose a system that can accurately identify the orchestrated gesture and map it to the desired word or alphabet in the sign language vocabulary using object detection algorithms in conjunction with neural networks, typically the YOLOv3.

CHAPTER: INTRODUCTION

Sign Language is a non-verbal form of communication used by people with impaired hearing and speech across the world. Sign language has quickly established itself as a primary language among many users in different parts of the world. Sign Language is not an arbitrary language. Rather, it has been observed that most sign languages have their own vocabulary and grammatical structure. This leads to the belief that sign language possess a lot of similarities with spoken language. People that employ sign language have found it to be a gradual learning process. The end of this learning process begets an easy, simplified and effective method of communication, specially between those that belong to this community of sign language users.

While sign language users are comfortable with using it as medium for communication between each other, there arises an issue when they are to communicate with people that don't use sign language. The conversation between a user of spoken language and sign language is a situation that needs to be observed carefully. In most cases, the spoken language user has no idea regarding the dynamics of sign language. They lack knowledge about the meanings of signs, related context, grammar, etc. Hence, the sign language user cannot effectively express themselves as the person they are talking to has, essentially, no idea about what they are doing. This begets the dependency to have a translator act as a moderator in the conversation. The translator is a person that has knowledge and high-level understanding of both the spoken language and sign language. Thus, the translator can facilitate conversation between both individuals.

This research is focused on proposing a system that mimics the role of a translator in the situation where a user of sign language is verbally communicating with a user of spoken language. It is a quantitative assessment of translation with its foundation derived from Neural Networks and

Object Detection. The intent is to provide an application that can be used to make sense of sign language, especially for those that are unfamiliar with the concept altogether. To understand the focus of this research, we shall understand the motivations behind this thesis and then highlight the expected goals and outcomes.

1.1 Motivation

The dependency on a translator is one of the primary motivating factors for this study. Sign language users are constantly wary about the fact that they will be unable to express themselves in the absence of a translator. Any attempts to express themselves may also be misunderstood by other individuals that can lead to a plethora of unnecessary problems. For instance, public places are a prime example where translators will, generally, not be found in abundance. Assuming the availability of translators at every corner of the world is unwise (R. Anderson, Wiryana, Ariesta, & Kusuma, 2017). Hence, a working model that mimics the role of a translator can help improve interaction between individuals.

An application can also help in education (Adamo-Villani, Heisler, & Arns, 2007). A tool that can recognize and translate the fundamentals of sign language can be a great asset in teaching aspirants. People who are new or are struggling to learn the language can experiment with this tool at will and learn or practice at a pace that they are comfortable with.

Prior research aimed at object detection (C. Zhang, Platt, & Viola, 2006), gesture recognition (Waldron & Kim, 1995) and gesture classification (Kim, Ji, & Lee, 2018), has achieved different levels of accuracy when attempting such systems. However, applications were unable to achieve desired levels of accuracy, precision and speed due to technological constraints (Waldron & Kim, 1995). In the last few years, technology has advanced at a rapid rate. This has led to the discovery

of many great applications, specifically in the field of object detection and language translation (Kadous, 1996) (Chunli, Wen, & Jiyong, 2001) (Aran & Akarun, 2010). This bolsters the possibility of a similar breakthrough in the field of sign language translation.

1.2 Goals

The objective of this study is to propose a system that can recognize the orchestrated sign or hand gesture and provide the correct translation for it in the English language. The study uses the vocabulary and rules enforced by the American Sign Language (ASL) (Battison, 1978). Deep learning concepts are utilized to create such a system. The framework provided by the You Only Look Once algorithm (YOLO) (Redmon, Divvala, Girshick, & Farhadi, 2016), YOLOv3 (Redmon & Farhadi, 2018), is utilized for detecting and classifying different signs as depicted in ASL.

1.3 Research Question

The guiding research question for this study is to understand whether:

Is it possible to create an American Sign Language translator using YOLOv3?

1.4 Limitations

The limitations of this project are as follows:

1. The dataset used for this research only contains the ASL signs for the alphabets of the English Language (A-Z).
2. The trained model is susceptible to inconsistency when tested for images that are extremely diverse from the images that the models are trained on.
3. The Word Builder Script is applicable only for images.

1.5 Delimitations

The delimitations of this project are as follows:

1. The labelling, training and prediction of dynamic gestures for alphabets J and Z is based on the final position of the ASL signs for these alphabets.
2. The dataset is divided into 3 unique clusters. Each cluster is trained independently of the other. This is done to avoid issues with memory and to prevent system failure during training process.

CHAPTER 2: LITERATURE REVIEW

This section explores the research that has been conducted in the area of sign language translation. We start off by discussing the background of American Sign Language (ASL). Next, we briefly provide a description of the translation process as defined in computer vision. We then elucidate the concept of Object Detection, its several methods and multifarious applications. The emphasis is on the You Only Look Once (YOLO) algorithm (Redmon et al., 2016) and how research on sign language recognition/translation has been conducted in the past using this algorithm.

2.1 American Sign Language

American Sign Language is a derived type of sign language used primarily by deaf people in the United States as their medium for communication. ASL is believed to have originated in the early 1900's. It has been influenced by several different languages, predominantly the French Sign Language (Battison, 1978). Over the years, ASL has evolved into a coherent and consistent language defining its own unique vocabulary, grammar and structure. ASL provides a hand gesture for each alphabet of the English language(Battison, 1978). However, signs are not limited to just alphabets. Rather ASL has evolved to such an extent that many words and phrases of the English language can now be represented through hand gestures. Figure 1 (Vogler & Metaxas, 2003) shows the signs for alphabets A, S and L as per ASL. Figure 2 (Vogler & Metaxas, 2003) shows the sign for I Love You as per ASL.

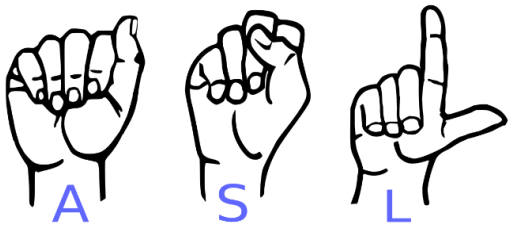


Figure 1 Representation of A, S and L in American Sign Language.
Figure adopted from (Vogler & Metaxas, 2003).

Figure 2 Representation of I Love You in American Sign Language.
Figure adopted from (Vogler & Metaxas, 2003).

At any given moment, an ASL sign consists of five salient aspects, Handshape, Location, Facial Expressions, Hand Orientation and Hand Movement (Stokoe Jr, 2005). Each of these factors have a direct impact on the meaning of a hand gesture. Based on these factors, ASL signs can be broadly classified as Static and Dynamic hand gestures. Static gestures are defined as the signs that don't involve any movement with the hand for a small fraction of time. Figure 3 (Vogler & Metaxas, 2003) shows the hand gesture for alphabet A in ASL. No movement is involved in explaining the meaning of the sign. Dynamic hand gestures involve some form of movement to help express the meaning of the sign. Figure 4 (Vogler & Metaxas, 2003) shows the hand gesture for Grandmother. The line represents the movement of the hand. This activity of representing the letter, numbers or

words of a language through hand gestures is known as Fingerspelling (Wilcox, 1992). This thesis concentrates on capturing the essence of fingerspelling through our proposed system.

Figure 3 ASL Gesture for Alphabet A.
Figure adopted by (Vogler & Metaxas, 2003).

Figure 4 ASL Gesture for word Grandmother.
Figure adopted from (Vogler & Metaxas, 2003).

2.2 Sign Language Translation

Gesture recognition is a small part of a high-level system (Vogler & Metaxas, 2003). This larger system is composed of several sequential steps, each having its own associated fields of research. Each individual step lays out the foundation for its subsequent steps (Vogler & Metaxas, 2003). Therefore, while each step can be explored independently, any advancements in one field, such as Object Detection (C. Zhang et al., 2006) (Song, Chen, Huang, Hua, & Yan, 2011), Feature Extraction (Ren, He, Girshick, & Sun, 2015), etc. can have a severe impact in all other remaining fields of research. The analogous field of research for this larger system is known as Sign Language Translation (Vogler & Metaxas, 2003). In the simplest of terms, translation can be defined as the process of inferring meaning from a foreign language to a native language. It generally involves translating each word of a different language into a more familiar language. Sign language translation follows a similar protocol. As is the case with the spoken language, the idea to is to interpret the meaning of an orchestrated gesture (Vogler & Metaxas, 2003). This can be understood if the people have knowledge about the vocabulary, grammar or contextual rules of sign language. Since sign languages can have convoluted rules regarding their use and it is difficult to learn an entirely new language, we rely on computer vision to create applications that can help ease this process.

According to Vogler et al. (Vogler & Metaxas, 2003), this process can be understood by the following steps. The first step involves capturing the relevant information. This is generally achieved by taking images or videos of orchestrated gestures. To add diversity, these are taken in batches with each batch exhibiting changes in settings such as location, lighting, orientation, etc. The second stage involves feeding these images or videos to a computer. The computer acts as a

system that is responsible for providing answers to a given question. In this case, the question asked pertains to the detection and classification of gestures.

Once all information has been inputted in the system, the focus shifts on the task of feature detection. Normally, one or more algorithms are used to extract static or dynamic features from the given input. These features correspond to the salient information of the gesture within the image such as Position of Hands, Facial Expressions, Finger Movement, etc. The final output is a feature vector that represents characteristics of a gesture in an input image.

Once feature vectors have been created, the system has enough information to begin training using these vectors. Once training is successful, the system can predict the gestures or signs orchestrated in a given image. The information given by the system regarding the predicted sign is dependent on the researcher as well as the algorithm or network used for detection and classification purposes. Some systems will only output the predicted class of the sign while others, in addition, will highlight location of the sign, confidence for prediction, etc. in the given image.

The final step involves coherent and cohesive translation. Remember, the process is not complete until we can make sense of the output. The end goal is to always have the user of such a system understand the meaning of a given sign. Typically, this step focuses on providing a textual list of all orchestrated signs in the specific order that they were conducted. Sophisticated applications will convert this textual list into audio and allow people to hear the meaning of the sign in real-time.

The goals of this research are layered between the steps of recognition and translation. We concentrate on proposing a system that can perform both tasks instantaneously.

2.3 Object Detection

Object Detection is one of the most versatile research areas in the field of computer vision (C. Zhang et al., 2006). It has become the building block for several real-world applications such as Object Tracking, Autonomous Driving, Face Detection, Video Surveillance, etc. Object Detection is the task of detecting a custom object in some form of graphical media such as an image, video, etc (Wikimedia, 2008). These images or videos can either contain multiple objects or a few objects at multiple locations. The task is not limited to just listing the different objects that are there. It also involves providing information regarding the location of the object in the image. This is usually accomplished by providing the coordinates of the object in the image. Additional information may also include a bounding box that specifies the location as well as the probability with which the object was detected.

Detection is often clubbed with classification (Fasel, Fortenberry, & Movellan, 2005) (Song et al., 2011) (Redmon et al., 2016). Both tasks have been known to be simultaneous as opposed to sequential. Typically, most algorithms or networks will not train inputs without the information regarding the different classes of objects that can exist in the training data. While training itself may become a tedious task, it has been noticed that newer algorithms can drastically decrease their computational load and provide results faster due to this information being fed to the system. The inception of object detection started in the early 1960's, but accuracy was a major concern and results were considered to be subpar (C. Zhang et al., 2006) (Ren et al., 2015) (Redmon et al., 2016). In addition, there was not enough technological support to implement high scale projects for real-time applications.

2.3.1 Deep Learning

Deep Learning is an application of Machine Learning that attempts to replicate the functioning of a human brain (Deng & Yu, 2014). The emphasis is on duplicating the cognitive and decision-making abilities of the human mind, neurons to be specific, and create a system that can work at the same level and capacity. The human brain is capable of observing intricate details about an object and learning from these subtle nuances to create a sense of understanding regarding the differences between multiple objects. Inference gathered and conclusions made regarding these objects are then stored in the memory. By using information stored in the memory, the brain can now detect the same object even in different settings. The objective of deep learning is to mimic a similar mechanism, create a network that can learn from the information provided to it and serve a purpose using all learned details (Ngiam et al., 2011) (Deng & Yu, 2014) .

2.3.2 Neural Networks Approaches

The traditional Convolutional Neural Network (CNN) uses a sliding window to search for an object in every possible position in the image. However, it is inefficient. This is due to the simple fact that different objects can exist in different sizes in the image and thus running the network with a pre-determined sliding window size was slowing down the network exponentially.

RCNN's were one of the first attempts at improving traditional convolutional neural networks for the task of object detection. RCNN's improved the detection process by creating regions using a selective search algorithm. These regions were then passed through a CNN to classify the object in the image. Bounding boxes that create these regions are then modified using regression techniques (Girshick, Donahue, Darrell, & Malik, 2014). RCNN's were a great innovation when first introduced in the realm of object detection. However, they were unable to generate real time

results. They are also largely dependent on the selective search algorithm (Uijlings, Van De Sande, Gevers, & Smeulders, 2013). It is observed that there is no process of learning at the stage of region creation (bounding box generation), thus there is a high probability of bad regions being generated.

Faster RCNN's (Von Zitzewitz, 2017) (Ren et al., 2015) (Girshick et al., 2014) are an advancement from conventional RCNN's. While both methods follow a similar approach, Fast RCNN's feed the entire image as the input to the CNN as opposed to individual regions in conventional RCNN's. The images are used to generate convolutional feature maps. A separate network is used to locate regions of interest from the feature map. The regions of interest propose areas in the image where it is likely to detect an object. A Pooling layer serves the purpose of resizing regions, which are then fed to a hidden layer of the network that predicts the appropriate class of the detected objects. Faster RCNN's are good at detection and provide improved accuracy over RCNN's but lack real-time application. In addition, they make use of a two-step network that is complex and may require individual training for each network making it computationally expensive.

You Only Look Once (YOLO) (Redmon et al., 2016), Single Shot Detectors (Liu et al., 2016), etc. are a few examples of the approaches that outperform the networks listed above. For the purpose of this research, we will not dive into the details of any other framework other than YOLO and its different types.

2.3.3 YOLO

You Only Look Once (YOLO) is one of the most popularly used advanced object detection model. The entire object detection pipeline is combined in a single neural network (Redmon et al., 2016). The framework is based on convolutional neural networks that has been modified to simultaneously predict bounding boxes, for object detection, and class probabilities, for object

classification. The unique name is based off the fact that the network sees the entire image only once during training, yet it is able to infer contextual information about the different objects in the image as well as their associated class. YOLO boasts of high-speed detection while maintaining precision and accuracy (Redmon et al., 2016). It has also shown tremendous results for real-time applications due to its low latency. YOLO provides the following advantages over existing techniques (Redmon et al., 2016):

- Conventional CNN's use separate networks for static feature extraction, region classification, bounding box prediction, etc. These are all combined into a single network, optimizing the entire process for faster and more accurate predictions.
- RCNN's proposes about 2000 regions using its selective search algorithm that are then passed through the network. YOLO proposes approximately 98 bounding boxes per image. Thus, it performs the same task with less computation.
- Fast RCNN's are known to falsely detect background images as objects due to its reduced context. YOLO performs far lesser false background detection. In addition, it proves to a better method for real-time applications.

While YOLO provides a better methodology than previous approaches, it still struggles in certain aspects. These limitations can be understood as follows (Redmon et al., 2016):

- YOLO is strict with imposing spatial constraints. It is estimated that each grid cell predicts only two bounding boxes for one class. This limits the number of objects the model can detect.
- Issues regarding generalized objects with varied aspect ratios or configurations.
- Errors regarding bounding box predictions for large and small objects. No clear distinction for both types of errors.

2.3.4 YOLO9000

YOLO proved to be a unique, novel approach that could potentially set the tone for future object detection applications. Yet, there were some limitations that prevented it from achieving its full potential. These limitations were systematically addressed and countered. This new and improved system is called YOLO9000 or YOLOv2. The motto is to provide a better, faster and stronger architecture (Redmon & Farhadi, 2017). The improvements can be understood as follows:

- YOLO9000 uses the concepts of anchors to make predictions as opposed to the fully connected layer in YOLO. Anchors are estimates of the size of bounding boxes. Doing so decreases the mean accuracy by 0.4, but it decreases computation cost by 33%.
- Anchor boxes help in increasing recall. The procedure of creating bounding boxes on objects in the training set drastically boost the speed of the training process.
- YOLOv2 forces the network to train on images with various resolution sizes. After a fixed batch size, the network chooses a new dimension size. YOLOv2 is also capable of running at 448x448 resolution, making it a high-resolution classifier.

2.3.5 YOLOv3

YOLOv3 is an incremental change to the existing YOLOv2 architecture. The changes are listed as follows (Redmon & Farhadi, 2018):

- It makes use of an “objectness score” for each predicting bounding box. This is achieved using Logistic Regression. It is done to pick the closest possible option to the ground truth.
- Prediction of classes is also done using logistic classifiers.
- The overall network is much larger than YOLOv2. Residual connections make it a 53-layer network. This helps in increasing accuracy.

2.4 Related Work on Sign Language

There has been significant research done on sign language recognition and translation in the pre-deep learning era. Majority of these were based on techniques such as Hidden Markov Models (Lee, Gauvain, Pieraccini, & Rabiner, 1993), Maximum-Likelihood approaches (Siskind & Morris, 1996), Hierarchical approaches (Cui, Swets, & Weng, 1995), etc. Research conducted by Nam et al. used HMM's for continuous gesture recognition (Nam & Wohn, 1996). Gao et al. (Gao, Fang, Zhao, & Chen, 2004) propose methods to recognize German sign language using Dynamic Time Wrapping (Mustafa, 2014). Braffort (Braffort, 1997) discovered ARGO, a dynamic architecture for recognizing French Sign Language. Their research proposed an integration of recognition and understanding as opposed to having them as separate steps. Kadous et al. (Kadous, 1996) worked on recognition of Auslan signs using Power Gloves. Their focus was on using computationally inexpensive methods for recognition of 95 isolated signs in the Auslan sign language. Chunli W. et al. (Chunli et al., 2001)proposed an isolated recognition system for the Chinese Sign Language with more than 5000 unique signs using HMM's. The breakthrough provided by Deep Learning in the field of computer vision stimulated research using neural networks. Erenshteyn (Erenshteyn & Laskov, 1996) recognized fingerspelling using neural networks. Pugeault et al. (Pugeault & Bowden, 2011) use Microsoft Kinect to capture and translate fingerspelling in real time. Waldron et al. (Waldron & Kim, 1995) performed sign recognition for a small set of isolated signs using neural networks. Rao et al. (Rao, Syamala, Kishore, & Sastry, 2018) use a mobile application to provide real-time gesture recognition. They use multiple CNN architectures and compare results of metrics such as accuracy, precision, etc. Kim et al. (Kim et al., 2018) propose a sign language learning method using Region of Interest segmentation. This is implemented using the YOLO detection network. Zhang et al. (Q. Zhang, Zhang, & Liu, 2019) propose a continuous gesture recognition algorithm using Channel State Information and

YOLOv3. Data acquisition is conducted using CSI-based radio frequencies to generate grey-scale images, which are provided as input to the YOLOv3 network.

Hence, we can notice that there have been significant attempts made at Sign Language Recognition in the past. While classical methods such as Hidden Markov Models (C-H Lee, 1993), dominate most of these attempts (Vogler & Metaxas, 2003), we can still see a burgeoning interest in utilizing neural networks and its associated algorithms to achieve the same task. This thesis is dedicated to further that interest using YOLOv3 as its foundation.

CHAPTER 3: DESIGN METHODOLOGY

The methodology employed in this research is adopted from Sajanraj (2018). According to Sajanraj (2018), sign language detection using YOLOv3 can be accomplished by training the network on pre-labelled images that accurately label the gesture in the image. Their research is conducted on Indian Sign Language (Tomkins, 1969). Gestures used to indicate alphabets, number or words, in the American Sign Language, are treated as objects. These objects are then passed through the YOLOv3 network also knows as Darknet. The network is responsible for detecting the location of the object as well as predicting the appropriate class that the object belongs to. The class refers to the different alphabets, number or words, in the English Language, that the network has been trained on. For words in the English language that don't have a static gesture associated with them in ASL, we assume that Fingerspelling will be employed to enact the respective word. Figure 5 shows the methodology. Each individual component is described in this section.

Figure 5 Methodology Flowchart

3.1 Dataset

The dataset used for this research consists of images representing each alphabet of the English language as depicted in ASL. Figure 6 shows the different images that were used for the purpose of training and testing the neural network. The images contain upper/full body of an individual making the appropriate ASL sign. The primary reason for using full body images is to help the

network learn to isolate the ASL gesture from the body of an individual. This helps reduce and eliminate false positives that the network is prone to identifying on the face of an individual. This also boosts the applicability of the trained network in a real-world scenario that may consist of multiple individuals making multiple ASL gestures. Each image has a resolution of 3024X4032. The high resolution is to ensure that the network can clearly learn the differences between each individual ASL gesture and the likelihood of finding an ASL gesture in an image, thus helping accuracy and precision. There are 26 classes, each representing an alphabet of the English language.

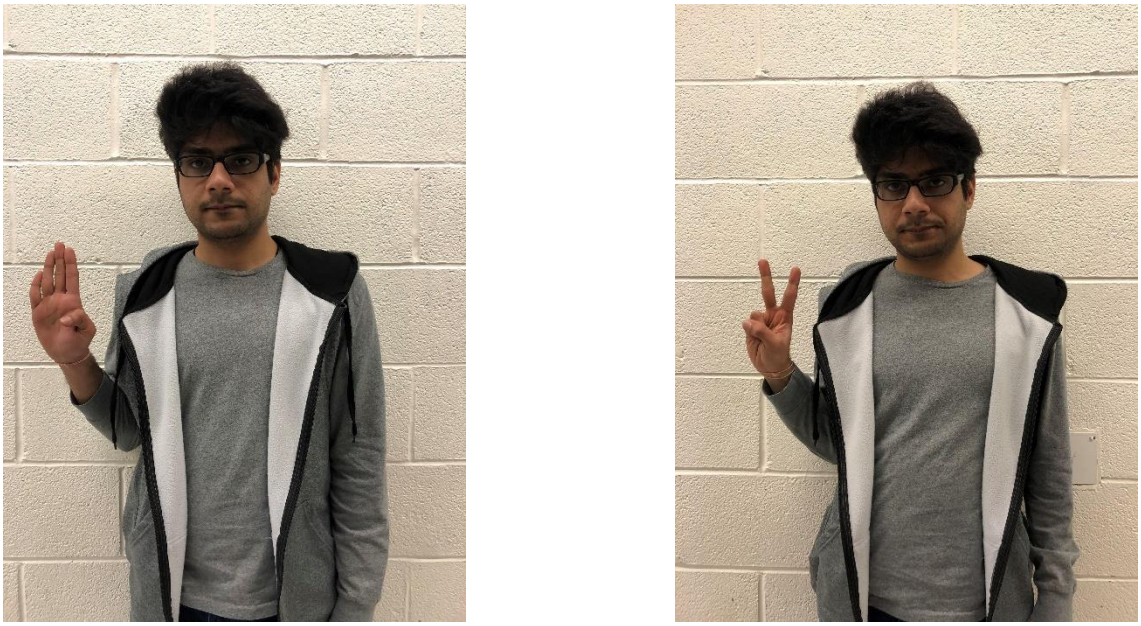


Figure 6 Sample images from the dataset. The images represent Alphabet B (in ASL) and Alphabet V (in ASL).

3.1.1 Image Labelling

A fundamental requirement of the YOLOv3 network is the use of pre-labelled images for training. The labeller used for this purpose is the Yolo_Label (Kwon, 2018). This labeller allows a user to

label multiple objects in the same image. The user-interface is very user friendly. It permits the user to select the appropriate object class and create corresponding bounding boxes anywhere in the image. Each bounding box has 5 parameters. The first parameter is the class of the object that the box is said to represent. The next two parameters are for the centre of the box with respect to the dimensions of the image. The last two parameters are the width and height of the box relative to the image. This file contains information regarding all objects that have been labelled in the image. Once a box is created, the information regarding its position is stored in .txt file (Kwon, 2018). This file is supplied to the network along with the image when the network is being trained.

3.2 YOLO Network

Before we dive into the details of the changes made to the existing network to suit our needs, it is important to understand the background of the existing network and the different concepts associated with it. These are described as follows.

3.2.1 YOLO Architecture

The YOLO algorithm is focused on examining a given input image only once. When it is being examined, the network is trying to extract features from the image to create a corresponding feature map that can be used to identify objects in a given image. As the feature map is populated with more information regarding the subtle nuances of each object, the network gains more confidence in predicting objects and their classes in an image. Feature extraction is accomplished using a conventional Convolutional Neural Network. The network consists of several convolution layers that help in transforming the input image. The network pipeline is shown in Figure 7 (Redmon et al., 2016). At the end of the pipeline are two fully connected network layers that are responsible for predicting bounding box co-ordinates and object class probabilities. The architecture can be

modified as per user requirement. This is generally accomplished by changing the number of identifiable objects, the number of batches for training, the anchors for bounding box predictions, etc.

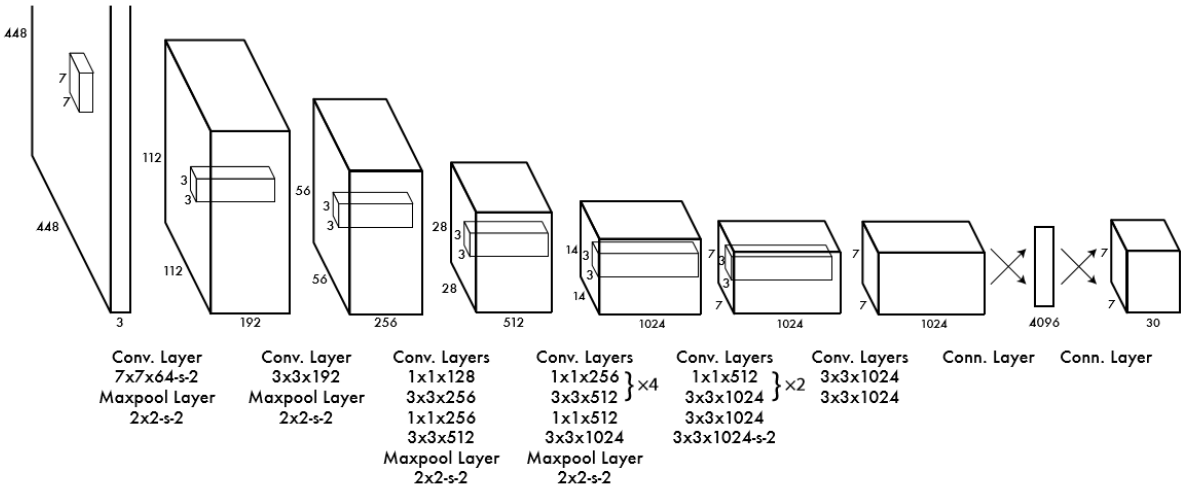


Figure 7 YOLO Architecture. The network has 24 convolutional layers followed by 2 fully connected layers.
Figure adopted from (Redmon et al., 2016)

The YOLO system divides an image into a grid. Each cell contains a portion of the image that may or may not contain an object. Typically, an object will occupy more than one cell in the grid. In such a scenario, only the grid cell that hosts the centre of an object is said to contain that object. This grid cell is thus responsible for predicting the object. This concept is realized by assigning bounding boxes to objects that are present in the image. Each bounding box has a confidence score associated with it. These scores are a reflection of how confident and accurate the network is while predicting that object. Ideally, if there is no object the score should be 0 and if the predicted position of the object is at the exact same position as in the original image then the score should

be 1. The confidence score is calculated using a concept known as Intersection Over Union (IOU) (Redmon et al., 2016).

The bounding box predictions contain 5 parameters. The first two parameters are for the centre of the box with respect to the dimensions of the grid cell. The next two parameters are the predicted width and height of the box relative to the image. The last parameter is the confidence score. Images that are used to train the architecture are labelled in the exact same way as described. Each image is labelled with bounding boxes that encapsulate each instance of an object in the image. Once a bounding box is drawn, the data regarding the 5 parameters is stored in a corresponding .txt file. This file contains information regarding all objects that have been labelled in the image. This file is supplied to the network along with the image when the network is being trained.

Smaller objects might be contained in one cell of the defined gird. Larger objects are more likely to be spread across multiple cells of the grid. Generally, the network will allocate an object to a cell if it hosts the centre of that object. However, this may be difficult to identify in case of large objects. The network may predict the same object multiple times, relative to each grid cell that it thinks this object belongs to. This will result in multiple predicted box locations for the same object that differ only slightly from each other. This can have a severe impact on accuracy. Figure 8 (Vogler & Metaxas, 2003) (Redmon et al., 2016) highlights a scenario where multiple boxes are served as results for the same object.

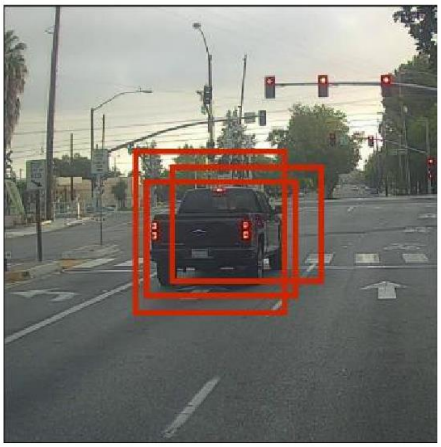


Figure 8 The same object with multiple predicted bounding boxes.
Figure adopted from (Redmon et al., 2016)

To avoid this issue, the network is equipped with Non-Max Suppression. Non-max suppression is used to filter through the predictions made by the network. The objective is to suppress duplicates that have low confidence scores. The box with the highest confidence is given priority, assuming that higher confidence translates to a stronger prediction. The final output from the YOLO network is shown in Figure 9 shows the final output obtained from a YOLO network.

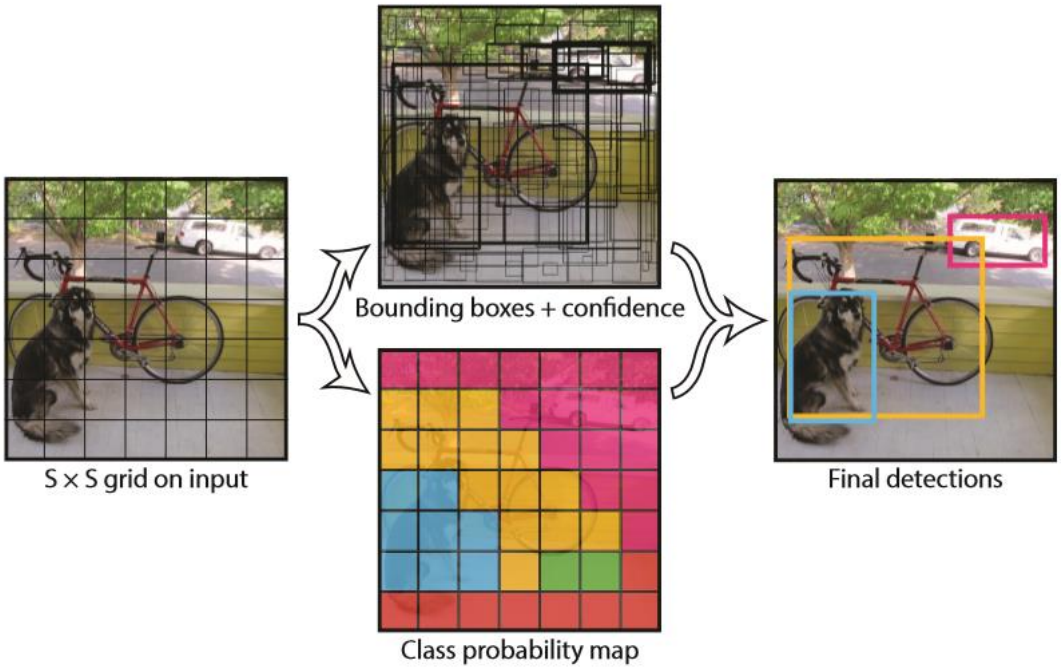


Figure 9 Final YOLO Output.
Figure adopted from (Redmon et al., 2016).

3.2.2 YOLO9000 Architecture

YOLO9000 or YOLOv2 was introduced to strengthen the existing YOLO network architecture. There were significant changes made to the network with the aim of making it better, faster and stronger. One of its most noted accomplishments is the ability to detect over 9000 distinct objects while providing similar real-time detection capabilities as its predecessor (Redmon & Farhadi, 2017).

The most notable change is the alteration of the convolutional network structure. The new network known as Darknet-19 consists of 19 convolutional layers and 5 max-pooling layers. The network doubles the feature maps after each pooling step. Figure 10 (Redmon & Farhadi, 2017) shows the composition of Darknet-19. Another notable change is the inclusion of Batch Normalization. This helps regularize the model and increases overall accuracy.

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Figure 10 Composition of YOLO9000 or Darknet-19.
Figure adopted from (Redmon & Farhadi, 2017).

Anchor boxes are the most significant addition to the network. YOLO uses its fully connected and convolutional layers together to predict co-ordinates of bounding boxes. This is very tedious. Predicting offsets instead of co-ordinates simplifies prediction and helps the network learn. By using anchor boxes, each grid cell now predicts the change in the box dimensions rather than predicting absolute height and width. Further improvements can be understood through Redmon (Redmon & Farhadi, 2017). Figure 11 (Redmon & Farhadi, 2017) shows the improvement in accuracy from YOLO.

	YOLO							YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓
anchor boxes?				✓	✓			
new network?					✓	✓	✓	✓
dimension priors?						✓	✓	✓
location prediction?						✓	✓	✓
passthrough?							✓	✓
multi-scale?								✓
hi-res detector?								✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

Figure 11 Transition from YOLO to YOLO9000.
Figure adopted from (Redmon & Farhadi, 2017).

3.2.3 YOLOv3 Architecture

The YOLOv3 architecture follows the same principles as its predecessors. The difference arises in its ability to provide a faster and more reliable prediction. This is achieved by making subtle changes to the existing YOLO9000 architecture. The most notable change is the addition of more layers is the convolutional neural network. The new feature extractor is created to act as hybrid that encapsulates residual network architectures onto the YOLO framework. The network is larger than Darknet-19 but has several significant shortcut connections. The new network is termed as Darknet-53. Figure 12 (Redmon & Farhadi, 2018) shows the structure of Darknet-53. The system is said to perform as efficiently as other approaches and has the added advantage of being faster.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 12 Composition of Darknet-53.
Figure adopted from (Redmon & Farhadi, 2018).

Across all YOLO architectures, classification was implemented using the standard softmax classifier. It was noted that many objects tend to have overlapping labels. In such a scenario, standard softmax does not provide adequate accuracy, at par with expectations. Hence, standard softmax is replaced with independent logistic classifiers (Redmon & Farhadi, 2018). This suppresses the assumption that each box, mapping a given object, has exactly one class. Logistic classifiers enforce multi-label classification. This is more suited to real-time processing and application. A more comprehensive guide to all changes can be found in Redmon (Redmon & Farhadi, 2018).

3.3 Training

3.3.1 Gilbreth Architecture

Gilbreth is Purdue University's community cluster for running sophisticated applications. It is optimized for communities running GPU intensive applications such as machine learning, deep learning, artificial intelligence, etc (Purdue). It consists of Dell compute nodes and Nvidia Tesla GPU's with Intel Xeon processors. Each sub-cluster consists of different specifics. Primarily, there are 20 cores per node with 256GB of storage available and P100 or V100 GPU's available for use (Purdue).

3.3.2 Cluster Creation

A fair number of ASL signs have somewhat overlapping or somewhat similar gestures. The similarity can be attributed to factors such as: Number of fingers used to make the sign, Orientation of fingers used to make the sign, Orientation of palm, etc.(Stokoe Jr, 2005). This can be understood as follows:

- G and Q have the exact same handshape, the difference arising from the orientation of the palm. Q has the palm facing downwards whereas G has the palm facing sideways. Figure 13 shows the difference between the gestures.
- H and U use the same handshape, with two fingers pointing in a specific direction. For H the fingers point horizontal whereas for U the fingers are vertical. Figure 14 shows the difference between the gestures.
- A and S have the same closed fist orientation with the difference arising from the position of the thumb relative to the closed fist. Figure 15 shows the difference between the gestures. The same can be noted for alphabets M, N and T.

Figure 13 Difference between ASL signs from Alphabets G and Q. G has the fingers pointing sideways whereas Q has the fingers pointing downwards.

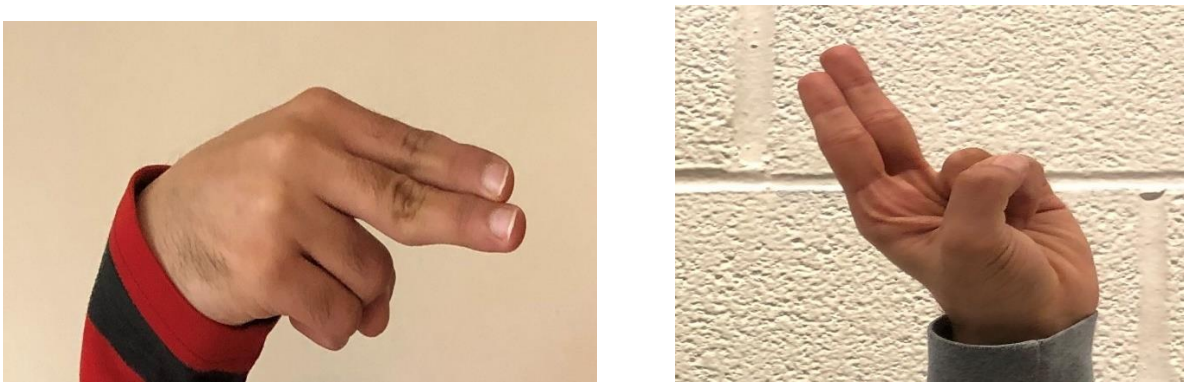


Figure 14 Differences between ASL signs for Alphabets H and U. H has fingers facing sideways whereas U has fingers facing upwards.

Figure 15 Differences between ASL signs for Alphabets A and S. A has the thumb next to the index finger whereas S has the thumb over the index and middle fingers of the hand.

Several tests were conducted using pre-training models to confirm if the model would indeed find it confusing to differentiate between these signs. Each pre-training model was trained for 5 alphabets, selected at random. The only condition used was to keep the alphabets with somewhat similar signs, such as A and S or M and N, in different sets. Tests were conducted with the images of all alphabets the model was trained on and the images of alphabets that had similar signs with the current alphabet set the model was trained on. The model showed great results for test images of alphabets that it was trained on. However, it would also show false positives for test images of alphabets that had very similar signs to the alphabets that the model was trained on. For example: A small network trained for alphabets A, B, C, D and F, when tested with images of the alphabet S, incorrectly marked all images of S with the alphabet A. This bolstered the notion that there was an inherent similarity between the ASL signs of alphabets A and S. Therefore, these alphabets needed to be trained together so that the model could learn the differences between the ASL signs of both alphabets. Similar results were observed for alphabets M and N, M and T, G and Q, etc. Therefore, keeping in mind the repetitive nature of results, appropriate clusters of alphabets were created for optimal training. Each cluster was trained separately with the images of all the alphabets that were deemed a part of that cluster. The specifications of these clusters are as follows:

- **Cluster 1:** It contains alphabets B, D, E, F, I, J, L, R, V, W, X, Y and Z. The ASL signs for these gestures are unique and independent of any other alphabet. All pre-training models had no difficulty in correctly identifying these alphabets for a majority of the tests conducted.
- **Cluster 2:** It contains alphabets A, M, N, S and T. These alphabets have very similar ASL signs, primarily attributed to the closed fist and spatial positioning of the thumb.

- **Cluster 3:** It contains alphabets C, G, H, K, O, P, Q and U. These alphabets have similar signs, primarily attributed to the position of fingers, the direction fingers are pointing to and the direction the palm is facing.

3.3.3 Source Fork and Modifications

The original source code for Darknet is provided by Redmon (Redmon et al., 2016) (Redmon & Farhadi, 2018) (Joseph, 2013--2016). This repository primarily defines the structure of darknet and its various specifics that are essential for training, testing and implementing pre-trained models. A more refined repository is provided by AlexeyAB (AlexeyAB, 2018). This fork adds additional features to the existing darknet repository making it easier to use, optimal and efficient. A thorough explanation of all changes that need to be made to the configuration of darknet are also provided by AlexeyAB (AlexeyAB, 2018). Standard changes are mentioned as follows:

- Changes to the MakeFile to indicate use of GPU's and OPEN CV.
- Changes to the training file to specify location of custom dataset.
- Changes to the obj.data and obj.names files to incorporate custom dataset.
- Including the modified .cfg file and pre-trained weights file to be used for training.

3.4 Testing

Testing is conducted in two phases, first with images and then with videos. In both phases, each cluster is tested independently of all other clusters. This is done to ensure that each cluster is efficient, accurate and proficient enough to correctly predict and recognize the alphabets that it has been trained on.

3.4.1 Google Colaboratory

Testing is conducted on Google Colaboratory (Colab) for ease and convenience. Google Colab is a large scale project that has transformed the landscape for machine learning and deep learning by providing computational hardware that can be used to train small and large neural networks (Carneiro et al., 2018). GPU's provided by Google Colab are extensively being used by researchers for various applications (J. Anderson, 2019) (Çavdar & Faryad, 2019). Processing and efficiency are not compromised by shifting to Google Colab from the Gilbreth cluster.

3.5 Word Builder Script

The world builder is a novel attempt at making sense from the information obtained from the network. In the above section, it has been established that the trained models are able to identify a given alphabet within an image and video. However, application should not be limited to just identifying an alphabet in an image. It is important to gather the results obtained from many such images and make sense of the results provided by the trained models. The word builder script accomplishes this task by combining the results, identified alphabet, of multiple images when supplied as a continuous chain of input to the trained models. This script is only applicable for images that require testing and has not been created or tested for videos. Testing is conducted with the set of test images that were used to train each cluster.

The script is created in Python. It primarily parses through the output generated by darknet when an image is tested with a trained model. While it is parsing the output, it extracts all information related to the alphabet that has been identified by the trained model for that respective image. This information contains the identified alphabet and the confidence with which the model has predicted the existence of this alphabet in the test image. For example: A: 90%. B: 100%, etc. The extracted

information is stored in a dictionary that is then sorted based on the confidence level next to each alphabet. The alphabet with the highest confidence score in the dictionary is then concatenated in the string, named as Word, that contains the results of all previously tested images. This procedure is repeated for each image. Since each cluster has its own unique trained model, each image is tested with the weights file (model) of each cluster and results are stored and parsed accordingly. This script was tested with unique words such as Drama, Car, Driving, Man, etc. and was successful in identifying each word correctly.

CHAPTER 4: RESULTS

4.1 Images

Testing is conducted in two phases, first with images and then with videos. In both phases, each cluster is tested independently of all other clusters. This is done to ensure that each cluster is efficient, accurate and proficient enough to correctly predict and recognize the alphabets that it has been trained on. The first phase of testing focuses on testing with images derived from the training set of images. The images used to train each cluster contain identical images of each alphabet trained in that cluster. For example, the alphabet B is trained repeatedly using the same image. Before the models were trained, data was split into a 90:10 ratio with 90% of images used for training and 10% preserved for testing. This was repeated for each cluster. Each image in these test sets share characteristics such as white background, moderate lighting, ASL sign in fixed position and no blurred elements in the image. Additional images that had similar characteristics but were not directly taken from the training set of images were also included. These images have the ASL sign performed at different locations relative to the body of the individual performing the ASL sign. This adds diversity to the set of test images. Four standard areas have been highlighted:

1. ASL sign performed by the right hand with right hand close to the body
2. ASL sign performed by the right hand with right hand away from the body
3. ASL sign performed by the left hand with left hand close to the body
4. ASL sign performed by the left hand with left hand away from the body

Average Precision for each alphabet in a cluster is noted and these are used to calculate the Mean Average Precision (mAP) for the entire cluster. Figure 16 and Figure 17 show the Average Precision per alphabet, Mean Average Precision and predictions made by the model for Cluster 1.

Figure 18 and Figure 19 show the Average Precision per alphabet, Mean Average Precision and predictions made by the model for Cluster 2. Figure 20 and Figure 21 show the Average Precision per alphabet, Mean Average Precision and predictions made by the model for Cluster 3.

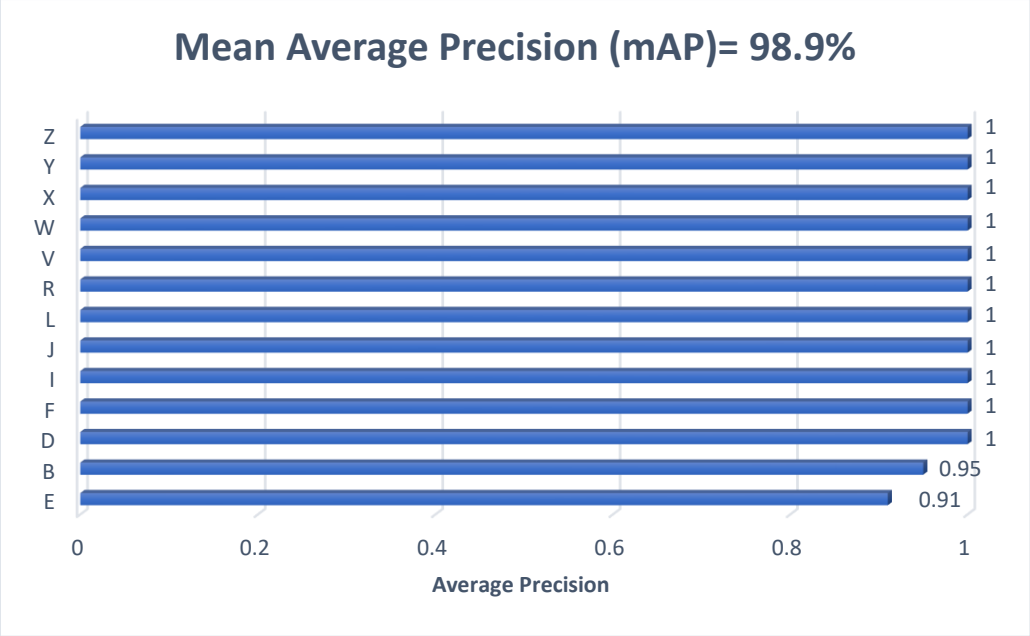
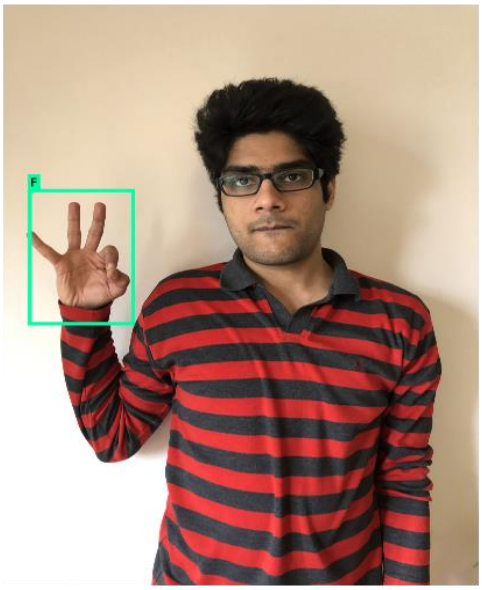
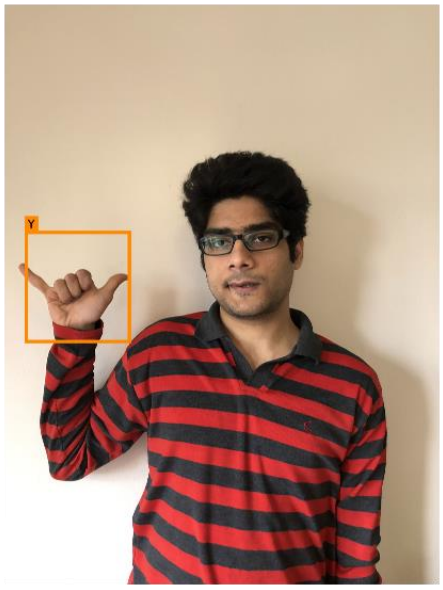


Figure 16 Mean Average Precision (mAP) of Cluster 1.





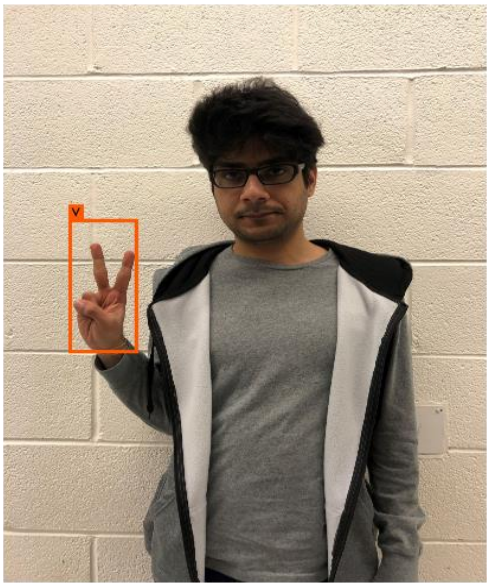


Figure 17 Predictions for Cluster 1. Alphabets B, F, Y and V respectively.

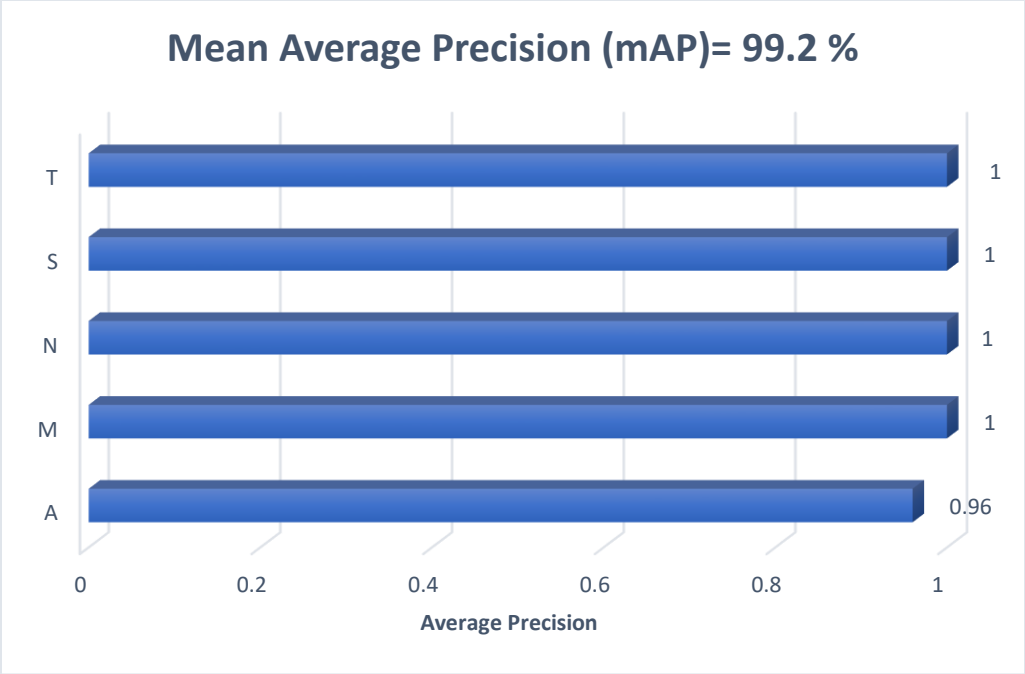


Figure 18 Mean Average Precision (mAP) of Cluster 2.

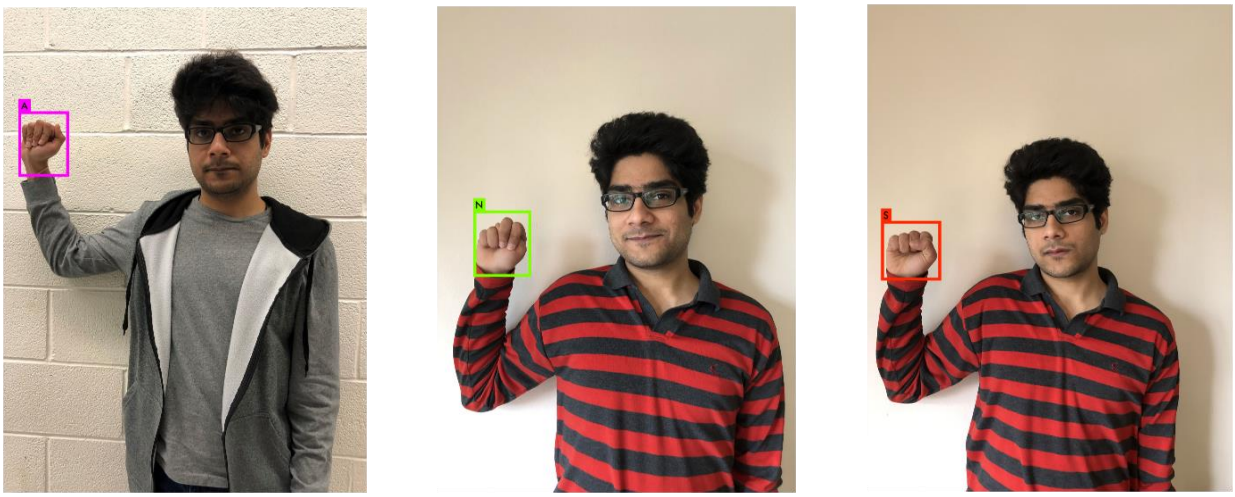


Figure 19 Predictions for Cluster 2. Alphabets A, N and S respectively.

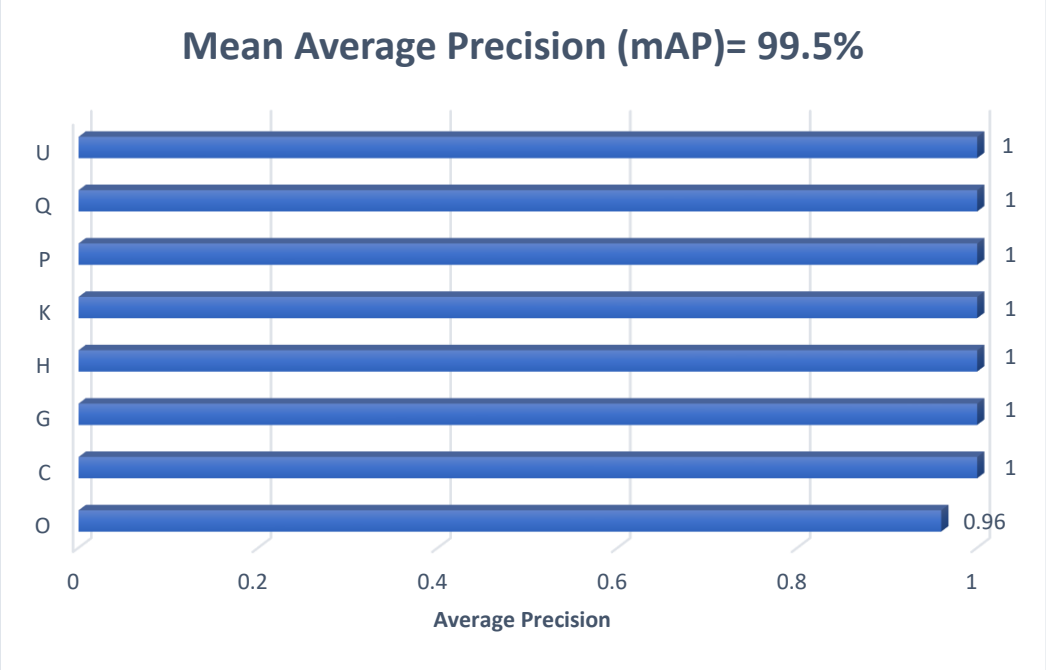


Figure 20 Mean Average Precision (mAP) of Cluster 3.

Figure 21 Predictions for Cluster 3. Alphabets K, C and U respectively.

4.2 Video

The next phase of testing is conducted on videos. Each cluster was tested independently of other clusters. Videos primarily contained the alphabets the cluster was trained on. To add diversity, some videos contained alphabets the cluster was not trained on. This is done to ensure that minimal false positives or true negatives arise during detection. Predictions by darknet are obtained at an average of 23 Frames per Second (FPS) for each cluster. Figure 22, Figure 23 and Figure 24 shows screenshots of some of the predictions made for each cluster.

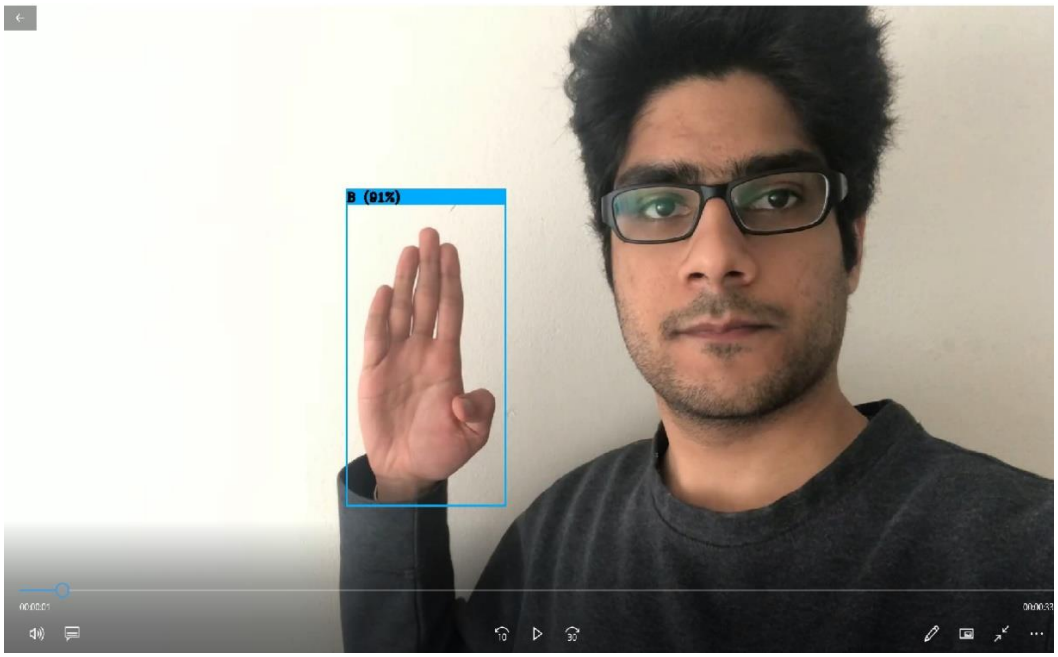


Figure 22 Prediction for Alphabet B in Cluster 1.

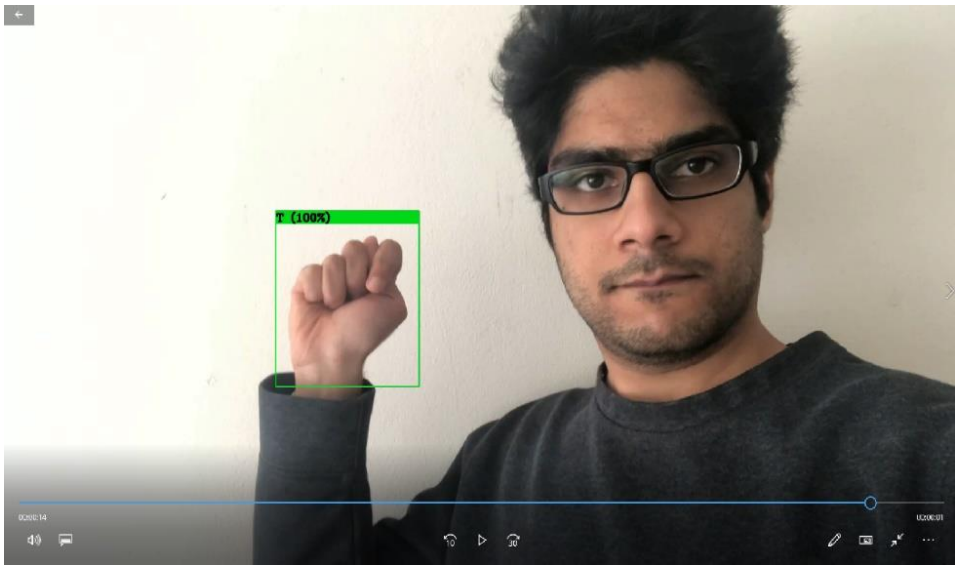


Figure 23 Prediction for Alphabet T in Cluster 2.

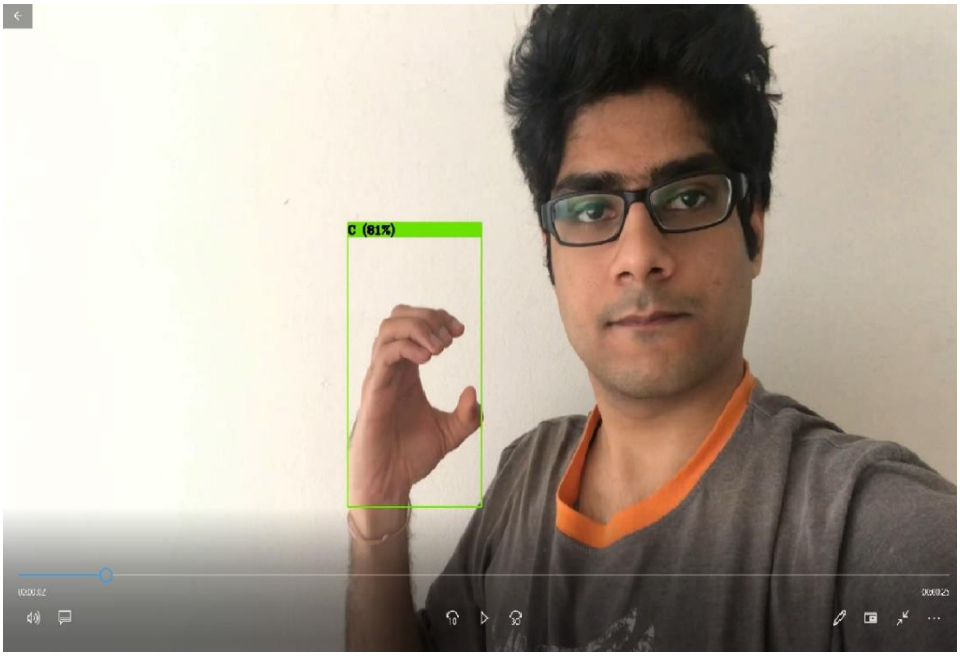


Figure 24 Prediction for Alphabet C in Cluster 3.

CHAPTER 5: DISCUSSION

5.1 Improving the effectiveness of Sign Language Translation using YOLOv3

Sign language translation using traditional methods has been explored for more than 25 years (Lee et al., 1993) (Nam & Wohn, 1996) (Gao et al., 2004). These methods have proven to be useful in their restricted domains and have provided adequate results to prove their efficiency and accuracy (Chunli et al., 2001) (Vogler & Metaxas, 2003). However, the era of deep learning has just begun. The current study aims to enhance the effectiveness of sign language translation by employing deep learning. The application of deep learning techniques in sign language translation is still unexplored and nascent. This research emphasizes on using deep learning methods, specifically YOLOv3, for achieving sign language translation. The research question particularly targets the use of YOLOv3 as the basis of achieving sign language translation. Viewing the translation process as a task of object detection and classification provides a unique way of approaching the problem. The results of this research provide a mean average precision of 99.2% across the three trained clusters, a metric comparable with the results produced by other novel deep learning approaches targeting sign language translation. Therefore, the results clearly indicate that YOLOv3 is a capable and reliable method for achieving sign language translation.

Prior research on sign language translation using deep learning reveals that Rao et al. (Rao et al., 2018) use multiple conventional CNN architectures and create unique sign language translators for each architecture. As mentioned in Section 2.3.3, Section 2.3.4 and Section 2.3.5, YOLO eliminates the problems encountered with conventional CNN architectures and provides results better suited for diverse real-time inputs. Kim et al. (Kim et al., 2018) proposed sign language translation using region of interest segmentation. The segmented region is found using YOLO.

Post the region is located, the identified region is further processed using a different network to classify identified signs. Our current research combines the task of identifying the region of interest and classification as simultaneous steps, by implementing the YOLOv3 algorithm. This helps improve the efficiency of the network which positively impacts the accuracy, precision and recall of the trained models. Zhang et al. (Q. Zhang et al., 2019) propose a gesture recognition algorithm using Channel state information to generate gray-scale images which are then provided as input to a YOLOv3 network. Our research builds on this idea and eliminates the need to provide gray-scale images as input to the YOLOv3 network. Instead, this research uses full color images as input to the network and emphasizes on training the network to detect and classify gestures in full color. Juan et al. (Figueroa, Sierra, & Arzuaga, 2019) propose a system for real-time ASL recognition using YOLOv3. The study conducted by Juan et al. (Figueroa et al., 2019) primarily targets detection and classification of ASL gestures in images and videos that contain only the hand performing the gesture. Their results prove the capability of the YOLOv3 network for performing sign language translation. Our research adds on to the existing body of knowledge with an intent to incorporate full-body images for training the network. The results provided in Chapter 4 reinforce the use of YOLOv3 as a unique, consistent and reliable medium for achieving sign language translation and provides an adequate answer to our research question.

5.2 Division of training set into clusters

Another key point of discussion is the division of the training set into clusters. This division is primarily done to avoid memory errors and system crashes during training. Prior research, using traditional methods or deep-learning techniques, focused on creating networks/systems that would train an alphabet set, list of words or other specifics altogether (Kim et al., 2018) (Figueroa et al., 2019). Training is conducted for a large number of classes without the need of dividing them into

sub-clusters. It is often speculated that the accuracy and precision of the translation process might be affected negatively due to this division (Waldron & Kim, 1995) (Pugeault & Bowden, 2011). However, the results of this study proved more effective than expected. Division into clusters significantly reduced the size of the training data by dividing into 3 parts. Each network that was trained on a specific clusters data was shielded from the heavy load of images that it would have encountered had this division not been done. In addition, it boosted the training process as each iteration/epoch of training made use of a significantly lower number of images than what would have been observed without this division. The results strengthen the notion of dividing training set into clusters as it did not impact average precision or mean average precision when trained models were tested with images. False positives of alphabets that the cluster was not trained on were minimal and had very little confidence scores associated with them. Hence, the trained models were proficient enough to recognize alphabets that it wasn't trained on and refrained from providing information regarding them. This can also be observed with the Word Builder script that specifically targets this ability of each trained model. Therefore, as future research strives towards capturing more signs in ASL that extend to words and phrases of the English language, the size of training data will increase exponentially and it can be argued that dividing the training set into clusters would prove to be as beneficial as training the entire data set together.

The results of the study highlight the efficiency and power of the trained models in achieving sign language translation. However, it is important to note that there are factors that limit this research. The study provides a basis for translation for only the alphabets of the English language, A-Z, as depicted in ASL. However, signs for words and phrases that are commonly used by individuals are excluded from this study. In addition, the trained models are susceptible to inconsistency when tested or images that are extremely diverse from the images the model was trained on. Section 3.1

and Section 4.1 provide information about the characteristics of the images used to train and test models. Testing with different backgrounds, lighting, other people, etc. can also be conducted to assess the level of variance that trained models can handle. Furthermore, a more sophisticated application could potentially provide predictions for real-time recognition using computer webcams, mobile cameras and other devices.

CHAPTER 6: CONCLUSION AND FUTURE WORK

The study focused on creating and proposing a model that could accurately and precisely predict the occurrence of an American Sign Language gesture for an alphabet in the English Language using the You Only Look Once (YOLOv3) Algorithm. The training dataset used for this study was custom created and was further divided into clusters based on the uniqueness of the ASL sign. Three diverse clusters were created. Each cluster was trained with the network known as darknet. Testing was conducted using images and videos for fully trained models of each cluster and Average Precision for each alphabet in each cluster and Mean Average Precision for each cluster was noted. An overall mean average precision of 99.2 % was recorded for all clusters. In addition, a Word Builder script was created. This script combined the trained models, of all 3 clusters, to create a comprehensive system that would create words when the trained models were supplied with images of alphabets in the English language as depicted in ASL. Therefore, the research question, “Is it possible to create an American Sign Language translator using YOLOv3”, is successfully and positively answered.

The following suggestions, thoughts and ideas can be incorporated for improving and extending the proposed model:

1. Make use of videos for training. These can add diversity to the training data set and allow the network to understand the subtle differences between gestures.
2. Increase the ASL signs that the network can learn and predict. Incorporate words in the English language that have designated ASL signs.
3. If there are no issues regarding memory and system failure, then the entire dataset can be trained together instead of being divided into clusters.

4. Incorporate different backgrounds, lighting, scenarios, people, etc. in the training dataset to increase the efficiency of the trained network and the diversity of testing that the network can handle.

REFERENCES

Adamo-Villani, N., Heisler, J., & Arns, L. (2007). *Two gesture recognition systems for immersive math education of the deaf.* Paper presented at the Proceedings of the First International Conference on Immersive Telecommunications.

AlexeyAB. (2018). Windows and Linux Version of Darknet YOLOv3 Neural Network for Object Detection. Retrieved from https://github.com/AlexeyAB/darknet

Anderson, J. (2019). Fully Convolutional Networks for Text Classification. *arXiv preprint arXiv:1902.05575*.

Anderson, R., Wiryana, F., Ariesta, M. C., & Kusuma, G. P. (2017). Sign language recognition application systems for deaf-mute people: A review based on input-process-output. *Procedia computer science, 116*, 441-448.

Aran, O., & Akarun, L. (2010). A multi-class classification strategy for Fisher scores: Application to signer independent sign language recognition. *Pattern Recognition, 43*(5), 1776-1788.

Battison, R. (1978). Lexical borrowing in American sign language.

Braffort, A. (1997). ARGo: An architecture for sign language recognition and interpretation. In *Progress in Gestural Interaction* (pp. 17-30): Springer.

Carneiro, T., Da Nóbrega, R. V. M., Nepomuceno, T., Bian, G.-B., De Albuquerque, V. H. C., & Reboucas Filho, P. P. (2018). Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications. *IEEE Access, 6*, 61677-61685.

Chunli, W., Wen, G., & Jiyong, M. (2001). *A real-time large vocabulary recognition system for Chinese Sign Language.* Paper presented at the International Gesture Workshop.

Cui, Y., Swets, D. L., & Weng, J. J. (1995). *Learning-based hand sign recognition using SHOSLIF-M.* Paper presented at the Proceedings of IEEE International Conference on Computer Vision.

Deng, L., & Yu, D. (2014). Deep learning: methods and applications. *Foundations and Trends® in Signal Processing, 7*(3–4), 197-387.

Erenshteyn, R., & Laskov, P. (1996). *A multi-stage approach to fingerspelling and gesture recognition.* Paper presented at the Proceedings of the Workshop on the Integration of Gesture in Language and Speech.

Fasel, I., Fortenberry, B., & Movellan, J. (2005). A generative framework for real time object detection and classification. *Computer Vision and Image Understanding, 98*(1), 182-210.

Figueroa, J., Sierra, H., & Arzuaga, E. (2019). Real-Time Hand Detection with the use of YOLOv3 for ASL recognition. Retrieved from https://indico.cern.ch/event/809812/contributions/3391220/attachments/1834047/3004274/Real-Time_Hand_Detection_with_the_use_of_YOLOv3_for_ASL_recognition.pdf

Gao, W., Fang, G., Zhao, D., & Chen, Y. (2004). *Transition movement models for large vocabulary continuous sign language recognition.* Paper presented at the Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). *Rich feature hierarchies for accurate object detection and semantic segmentation.* Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

Joseph, R. (2013--2016). Darknet: Open Source Neural Network in C. Retrieved from https://pjreddie.com/darknet/

Kadous, M. W. (1996). *Machine recognition of Auslan signs using PowerGloves: Towards large-lexicon recognition of sign language.* Paper presented at the Proceedings of the Workshop on the Integration of Gesture in Language and Speech.

Kim, S., Ji, Y., & Lee, K.-B. (2018). *An effective sign language learning with object detection based ROI segmentation.* Paper presented at the 2018 Second IEEE International Conference on Robotic Computing (IRC).

Kwon, Y. (2018). Yolo_Label. Retrieved from https://github.com/developer0hye/Yolo_Label

Lee, C. H., Gauvain, J. L., Pieraccini, R., & Rabiner, L. R. (1993). Subword-Based Large-Vocabulary Speech Recognition. *AT&T Technical Journal, 72*(5), 25-36.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). *Ssd: Single shot multibox detector.* Paper presented at the European conference on computer vision.

Nam, Y., & Wohn, K. (1996). *Recognition of space-time hand-gestures using hidden Markov model.* Paper presented at the Proceedings of the ACM symposium on virtual reality software and technology.

Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., & Ng, A. Y. (2011). *Multimodal deep learning.* Paper presented at the Proceedings of the 28th international conference on machine learning (ICML-11).

Pugeault, N., & Bowden, R. (2011). *Spelling it out: Real-time ASL fingerspelling recognition.* Paper presented at the 2011 IEEE International conference on computer vision workshops (ICCV workshops).

Purdue. Gilbreth User Guide. Retrieved from https://www.rcac.purdue.edu/knowledge/gilbreth/overview

Rao, G. A., Syamala, K., Kishore, P., & Sastry, A. (2018). *Deep convolutional neural networks for sign language recognition.* Paper presented at the 2018 Conference on Signal Processing And Communication Engineering Systems (SPACES).

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). *You only look once: Unified, real-time object detection.* Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

Redmon, J., & Farhadi, A. (2017). *YOLO9000: better, faster, stronger.* Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). *Faster r-cnn: Towards real-time object detection with region proposal networks.* Paper presented at the Advances in neural information processing systems.

Sajanraj, D. T. (2018). Real-Time Indian Sign Language Recognition using YOLO Object Detection Method. Retrieved from https://sajanrajtd.wordpress.com/2018/05/31/real-time-indian-sign-language-recognition-using-yolo-object-detection-method/

Siskind, J. M., & Morris, Q. (1996). *A maximum-likelihood approach to visual event classification.* Paper presented at the European Conference on Computer Vision.

Song, Z., Chen, Q., Huang, Z., Hua, Y., & Yan, S. (2011). *Contextualizing object detection and classification.* Paper presented at the CVPR 2011.

Stokoe Jr, W. C. (2005). Sign language structure: An outline of the visual communication systems of the American deaf. *Journal of deaf studies and deaf education, 10*(1), 3-37.

Tomkins, W. (1969). *Indian sign language* (Vol. 92): Courier Corporation.

Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision, 104*(2), 154-171.

Vogler, C., & Metaxas, D. N. (2003). *American sign language recognition: reducing the complexity of the task with phoneme-based modeling and parallel hidden markov models.* Citeseer,

Von Zitzewitz, G. (2017). Survey of neural networks in autonomous driving. In: Jul.

Waldron, M. B., & Kim, S. (1995). Isolated ASL sign recognition system for deaf persons. *IEEE Transactions on rehabilitation engineering, 3*(3), 261-271.

Wilcox, S. (1992). *The phonetics of fingerspelling* (Vol. 4): John Benjamins Publishing.

Zhang, C., Platt, J. C., & Viola, P. A. (2006). *Multiple instance boosting for object detection.* Paper presented at the Advances in neural information processing systems.

Zhang, Q., Zhang, Y., & Liu, Z. (2019). *A Dynamic Hand Gesture Recognition Algorithm Based on CSI and YOLOv3.* Paper presented at the Journal of Physics: Conference Series.

Çavdar, İ. H., & Faryad, V. (2019). New Design of a Supervised Energy Disaggregation Model Based on the Deep Neural Network for a Smart Grid. *Energies, 12*(7), 1217.