HAND DETECTION AND BODY LANGUAGE RECOGNITION USING YOLO

An Undergraduate Research Scholars Thesis

by

AASHISH ANANTHANARAYAN

Submitted to the Undergraduate Research Scholars program at
Texas A&M University
in partial fulfillment of the requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by Research Advisor: Dr. Anxiao Jiang

May 2020

Major: Computer Science

TABLE OF CONTENTS

Page

ABSTRACT 1

ACKNOWLEDGMENTS 2

IMPORTANT TERMS 3

CHAPTER

I. INTRODUCTION 4

Why Recognize Body Language? 4
About The YOLO Network 5

II. METHODS 7

YOLO for Hand Detection 7
YOLO for Body Language Recognition 9

III. RESULTS 14

Hand Detection 14
Body Language Recognition 16

IV. CONCLUSION 23

Future Work 23

REFERENCES 24

ABSTRACT

Hand Recognition and Body Language Recognition Using YOLO

Aashish Ananthanarayan
Department of Computer Science and Engineering
Texas A&M University

Research Advisor: Dr. Anxiao Jiang
Department of Computer Science and Engineering
Texas A&M University

Neural Networks play an important role in real-time object detection. Several types of networks are being developed in order to perform such detections at a faster pace. One such neural network that can prove useful is the YOLO network. Built to perform real-time detection, YOLO offers great speeds for simple detections. The goal of our research is to see how YOLO would work with body language. Would it be fast enough? And how accurate would it be?

Compared to other forms of object detection, body-language detection is more vague. There are several factors to be accounted for. This is why we first begin by talking about hand recognition and gesture recognition, and then move onto body language. This research aims at understanding how YOLO would perform when subject to several tests by using its implementations, building datasets, training and testing the models to see whether it is successful in detecting hand gestures and body language.

ACKNOWLEDGMENTS

I would like to thank my faculty advisor Dr. Anxiao Jiang for giving me the opportunity to work on this project and for his constant guidance along the way. I would also like to thank Srujana Bhaskarabhatta for mentoring me and the LAUNCH staff for helping me write this thesis.

I want to take this moment to thank my parents for all the sacrifices they have made in order to see me succeed. I would like to dedicate this paper to them. I also want to thank all my friends, especially Aakash, Param, Mitch, Abhinav, James, Cameron, Pablo, Michael, Sid, Shawn, Shravan, Sahil, Matthew, Pratik and many others for supporting me throughout this time and for always being there for me.

For Kim

IMPORTANT TERMS

Neural Networks: Systems inspired by human brains, they are interconnected webs of nodes that interact with each other and learn to perform tasks through training rather than being specifically programmed to do so.

YOLO: You Only Look Once. The neural network model used for this paper.

PyTorch: Open source Machine Learning Library

GPU: Graphics Processing Unit. Manipulates memory for image processing and computer graphics.

CHAPTER I

INTRODUCTION

Body language is an essential form of human communication. According to Carol Kinsey Goman, a contributor to Forbes Magazine, body language is 93% of human communication [1]. We thus notice the importance body language could have on the way we communicate with one another. Humans can convey a lot of different thoughts and feelings through simple bodily gestures.

Why Recognize Body Language?

Over the last few years, Deep Learning has made great advancements in the field of real-time object detection. Introduced by Redmon et al, the YOLO network is known for its superior speed and accuracy. YOLO has been very successful in detecting objects, like face detection, detecting the presence of cars on the street, detecting differences between animals and so on. Such neural networks have both speed and accuracy and show true powers of computation when it comes to such complex tasks. But what if we could go one step further and explore more intricate details? What if we could have computers detect a person's sub-conscious action in real time? Wouldn't it be awesome to know if a computer could truly be able to tell you exactly how you are feeling at a certain point? This research focuses on seeing if the YOLO network can be successful enough in detecting certain body language gestures and if it can be used even further to successfully detect how a person behaves.

About The YOLO Network

YOLO - You Only Look Once is a different approach to real-time object detection. Presented by Redmon et al, YOLO uses frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities [2]. Because of this, the single neural network can predict the bounding boxes and class probabilities directly from the full image in one evaluation. As a result, YOLO shows incredible speed in real-time object detection and is useful for simple detections like face recognition, hand recognition (as shown later in this paper), and so on. Image processing in YOLO is pretty simple, the system resizes to 448 x 448, runs a single convoluted network on the image and thresholds the resulting detections by the model's confidence [2].

YOLO is extremely useful because it also trains on full images and directly optimizes the detection performance. YOLO does not require any complex pipelining, as it is a regression problem for frame detection. Speeds are reported to run at 45 frames per second on the Titan X GPU, and the fast version runs at 150 frames per second [2]. Also, YOLO does not use any window or region based techniques, but it uses the entire image during training time and encodes contextual information about the classes as a result [2]. All of this helps in making YOLO super fast and hence, our choice of neural network for this thesis.

The version of YOLO experimented on in this thesis is YOLOv3. It is the most recent implementation of the YOLO network as it released back in 2018. It is considered to be an *incremental improvement* to the YOLO network [3]. It now performs detection of boxes at three different scales. As a result, it might not be faster than previous versions of YOLO, but it is definitely stronger and is better at detecting smaller images. The initial weights of YOLOv3 can be found in [4]. Using these weights as the starting point along with the YOLOv3 configuration files, we can begin working on object detections. It is a very easy method to follow and train, but first, you have to get your dataset ready and annotated accordingly. Over the course of this paper, in the *Dataset* sections, I have given a small description on how the dataset was annotated so that it fits the YOLO model properly.

So, without wasting any more of your time, lets dive deep and understand how YOLO can be used on the custom body language dataset in order to see if it can be successful in making quick detections.

CHAPTER II

METHODS

YOLO for Hand Detection

The first phase of this research comprises using the YOLO network for detecting hands and seeing its performance for simple hand detection. There are a few phases in this process that I am going to discuss.

Data Collection

The first step of the process is data collection. In order to get properly annotated images, I made use of the Oxford hands dataset [5]. This dataset comprises images and annotations from several sources like movies and television shows. However, these annotations have a slight difference. The YOLO network normalizes the x and y between 0 and 1 and uses x, y, w, h as its data inputs, where x and y represent the normalized center of the bounding box (the box around the area we want to identify, in this case, hands) relative to the grid cell bounds and w and h represent the width and height relative to the image. The Oxford Dataset on the other hand gives us the coordinates of the end points. Using a Python Script, these annotations were first converted into YOLO annotations and then, were placed into a text file for each image along with the class number. This means that every single image had its own annotated text file. For instance, if an image was Buffy_1.jpg, it had its annotated text file Buffy_1.txt. After this was completed, all the absolute paths of the images were stored in a text file called train.txt. This file would be used to get all the images and annotations and feed them into the YOLO network. Only certain images from this dataset were used, using different shots.

Figure 1. A) An example of an image from the Oxford Dataset [5]. B) An example of an image from the Oxford Dataset [5].

Training

Once data collection is complete, we can begin training the YOLO network. In order to do this, a PyTorch model of the YOLO network from [6] and [7] was used and readied for hand detection. Pre-trained YOLO weights from [3] were utilized in order to begin training the YOLO model. Once this was completed, the data was transferred to respective folders and the annotations were kept ready. In the configuration settings, the class name was set to Hands, the number of batches was set to 16 and the number of subdivisions was set to 1. This means that at a time, the GPU can look through 16 images instantly, which speeds up the training process. Also, filters = (classes + 5) x 3. As we have only one class, the filters variable was set to 18. Once this was completed, the code was run through the in-lab GPU. Initially, the number of epochs was set to 500 and fewer images were used in order to observe how training worked. The training displayed the losses of each x, y, w and h and the precision values. Every 50 epochs, weights were stored in a checkpoints folder, in order to keep a tab of the training. After that, the network was trained for 2000 epochs to observe the performance and loss values.

Testing

Once training was completed, it was time to test the network on unseen images and video in order to see if it works or not. In order to visualize the results, I worked on some code and took inspiration from [8]. The test images and video frames were unseen from the Oxford dataset and were placed in a separate folder. After this was completed, the weights from the training set were transferred to the configuration folder of the testing set. Then, a Jupyter notebook was used to view the final results. The script was run using the GPU and was opened in a Jupyter notebook using Localhost. After this was completed, the path to the weights was set to the weights we obtained from training the data and the test images were used in order to see if predictions were made. The results for these are displayed in the results section.

YOLO for Body Language Recognition

Now that we have completed hand recognition using YOLO, we should shift our attention to the second phase of this thesis, which is body language recognition. For this, we need to first understand what some body language signs mean and how YOLO could be used to recognize them. For testing YOLO on body language, I have read through a book by Joe Navarro, called *The Dictionary of Body Language - A Field Guide to Human Behavior.* A former FBI agent, Mr. Navarro has worked extensively on recognizing how humans interact with different body languages and how body language plays an important part in human non verbal communication. As a result, this is a very reliable source that I will be counting on for this research. In order to work with real time object detection, a few distinct body-language signs were used in order to figure out whether YOLO can be capable of recognizing body-language gestures. These gestures were run on the same implementation of the YOLO network, using the same methods that we used initially for hand detection to see if YOLO is successful enough in recognizing those body-language gestures in a speedy manner.

Running Fingers Through Hair (men)

When stressed, men will run their fingers through their hair both to ventilate their heads and to stimulate the nerves of skin as they press down [9]. As a result, this could be either if they are concerned, stressed or have any doubt. Another reason men run their fingers through hair is attraction. A lot of men do this subconsciously, maybe to look "cool" at times. Personally, I have done this during times of stress, for example when I was going through final revisions for this paper! This could be a sign for the neural network to pick up if a human male is stressed or not. In order to work on this, a slightly different approach was used to collect data, but the training and testing will be similar to that for hand detection.

Data Collection

For collecting the data, *web-scraping* was tried in order to find images of men running their fingers through their hair. Previous datasets were searched through in order to find images with this gesture. However, as I was unable find them through the datasets and was unsure about web-scraping laws, so I decided to just take pictures of myself running fingers through my hair. Once this was completed, bounding boxes were drawn round the required region. After this was done, the python script was executed to annotate the images according to YOLO specifications as discussed earlier, along with the class number. Then, the annotated text files and images were be placed into the same data folder, and once this was completed, the class name was changed to running_hair.

Training

Then the same training model, the PyTorch YOLO representation was used to train the network. The number of filters was again be set to 18, the number of classes to 1, number of batches at 16 and subdivisions at 1. The training script was run for 500 epochs and losses were recorded. Weights were saved every 50 epochs.

Testing

Using the object detector described in the previous Testing section for Hand Detection, unseen images from this dataset/frames from video were used in order to see if proper predictions are made. The weights from the training were be used in order to evaluate the final predictions. All results are displayed in the result section.

Finger Pointing and Gestures

There is a universal dislike to people having a finger pointed at them [9]. When someone points a finger at you, you do tend to feel a little disrespected and awkward. This form of hand gesture is is generally replaced by pointing all five fingers in a professional or a romantic setting[9]. Also, when we direct someone to a particular location, we tend to use all five fingers to usher a person to that location, this tends to be more respectful. Notice ushers in a theater, they'll always use five fingers to point to a location. Again, we would like to see if YOLO can recognize between respectful signs of finger pointing and disrespectful signs of pointing a single finger. While we are at this, let us also consider the *Gig'Em* Aggie sign, which is used by Aggies universally (except Germany, where it is considered obscene unfortunately).

Data Collection

The first step again was to collect data to feed into the YOLO network. In order to do this, the Pointing and command gestures under mixed illumination conditions dataset was selected [10]. First, the dataset was searched thoroughly to find images that contained a single finger pointing sign, a five finger pointing sign and the Aggie *Gig'Em* sign, which is called the sign of affirmation in this dataset. Once this was completed, every single image had to be labelled individually according to the sign it displayed and bounding boxes were drawn around them as the images here are not annotated. Once this was completed, the coordinates of the bounding boxes were taken and converted into the normalized YOLO format and was placed into a text file, one text annotation file for each image. The text file consisted of the class number and the x, y, w and h annotations and the class number as well.

Training

The training for this was similar to the training for hand detection. Training took place for 2000 epochs. Using the model, weights will be saved every 50 epochs and will be used for testing. The number of filters will be set to 18, number of batches to 16 and the number of subdivisions to 1.

Testing

After training was completed, the weights were be saved and transferred to the object detector as done previously. Using a Jupyter notebook, the weights from training were used and a few test images and video frames in order to test the final results and see if the system worked or not.

CHAPTER III

RESULTS

Hand Detection

Training for Initial Dataset

The initial dataset was trained for 2000 epochs and the losses for x, y, w and h and the total losses for each batch in each epoch were observed. The total losses were recorded for each batch for epochs for which the weights were stored and then averaged the batch loss within each epoch to get the epoch loss. Then, the loss curve was plotted.

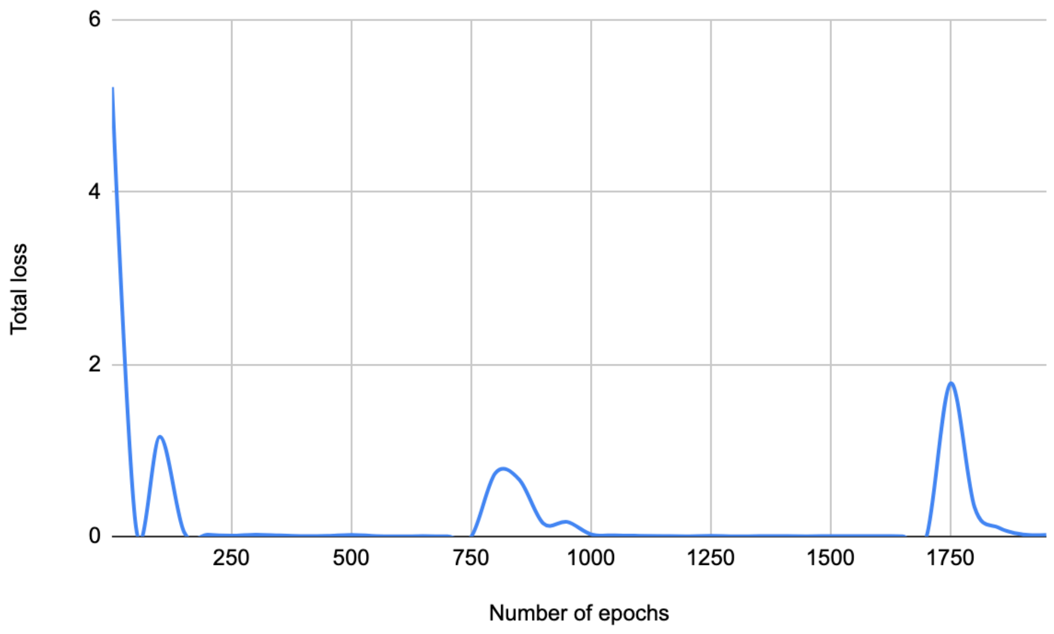


Figure 2. Loss curve for the initial hand dataset over 2000 epochs.

From figure 2, we can see that the loss decreases from extremely high during the first few epochs to very low during the later epochs. However, some spikes in the loss function were observed, which gradually reduced over time.

Testing for Initial Dataset

In order to test the results, weights from different epochs were used to see which ones worked best. Weights from multiple epochs were utilized to see which weights worked best with testing the image. Certain misclassifications were also noted for hand detection.

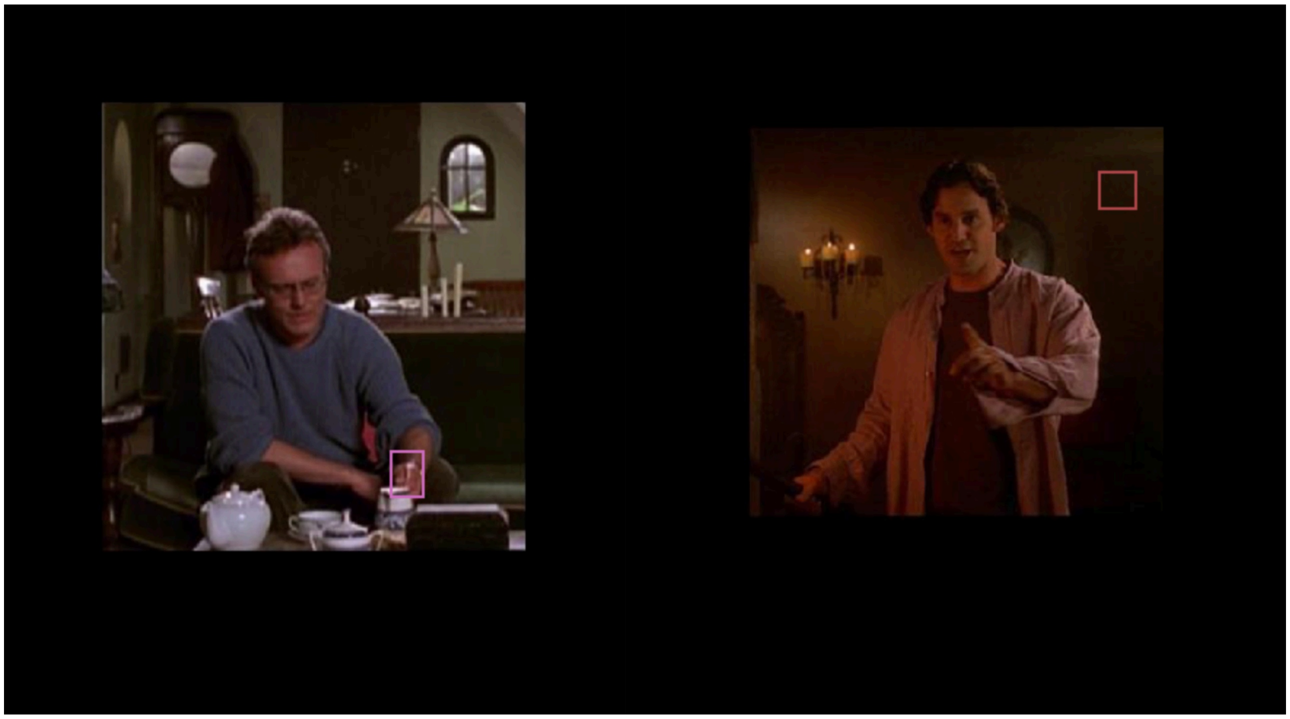


Figure 3. A) Correctly classified sample. B) Misclassified sample.

Body Language Recognition

Running Fingers Through Hair (men)

Training

The first thing observed was that due to large image sizes (as I used the .jpg images straight from my iPhone), training took a longer time. Also, the initial loss was extremely high, training took place for 500 epochs and the loss was calculated.

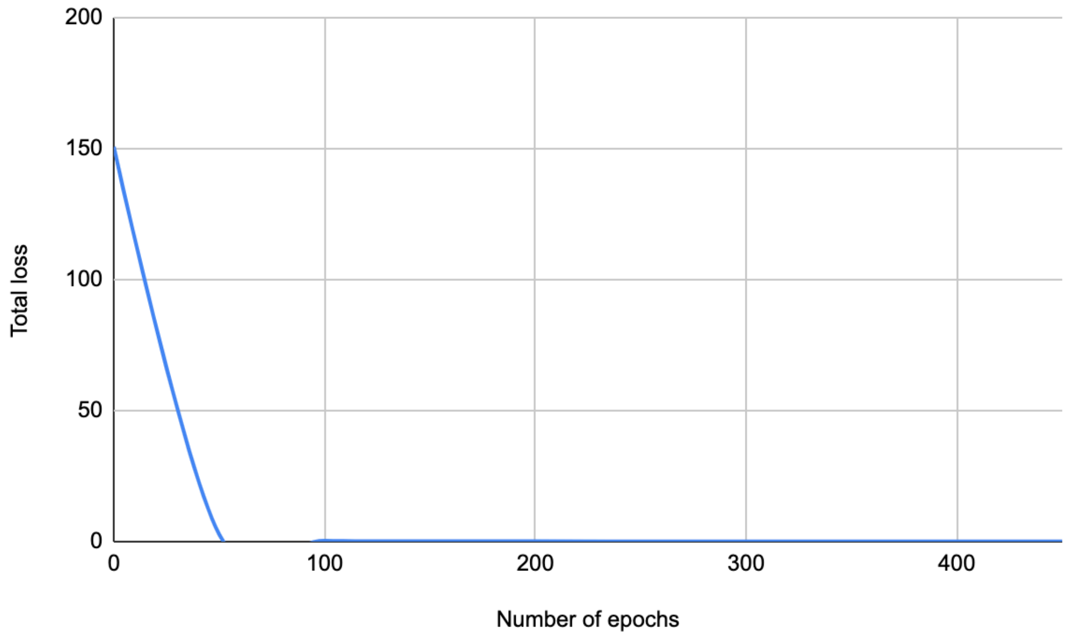


Figure 4. Loss curve for running fingers through hair training.

From this loss curve, we can see that the initial loss was indeed extremely high but reduced drastically as the model trained on. It took about 2.5 hours to train for 500 epochs which was a lot more training time than those for other features. This could be attributed to the image sizes and pixelation.

Testing

For the first 100 epochs, the loss reduced drastically, yet the image was not properly detected. When I ran a testing image using those weights, the bounding box did not accurately fit around the gesture. But as the epochs progressed, the results got better.

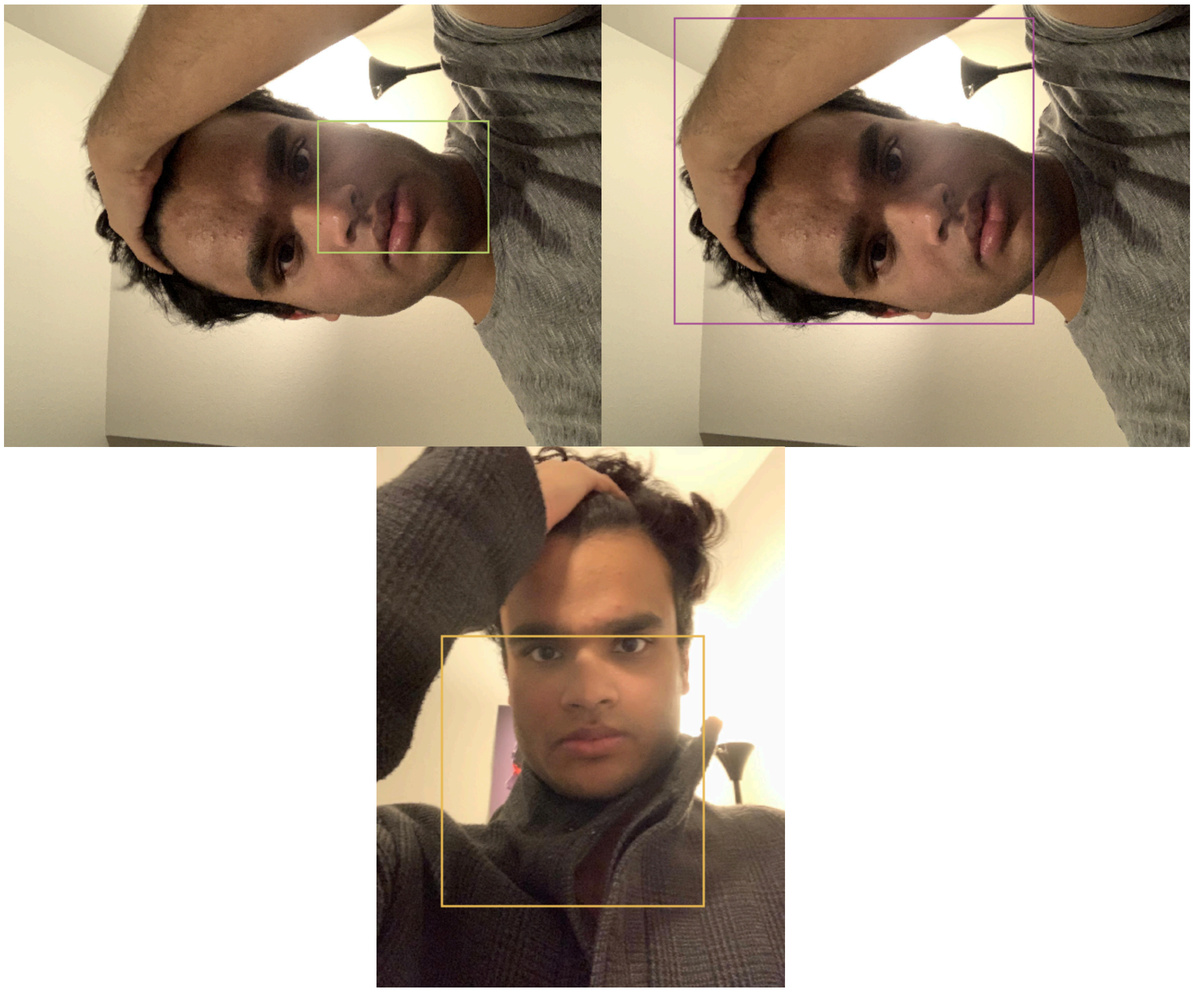


Figure 5. A) Testing after using weights from the first 100 epochs. B) Testing the same image after using weights from the 250th epoch. C) Testing a different image with different conditions (video frame) on the 250th epoch.

Finger Pointing and Gestures

Single Finger Pointing Training

First, data for the single point gesture was trained in the same way other data was trained. The number of epochs was set to 2000 and the x, y, w and h losses, along with the total loss was observed. Weights for every 50 epochs were saved as a checkpoint. In order to find the total loss for an epoch, the total losses for each batch in the epoch were averaged.

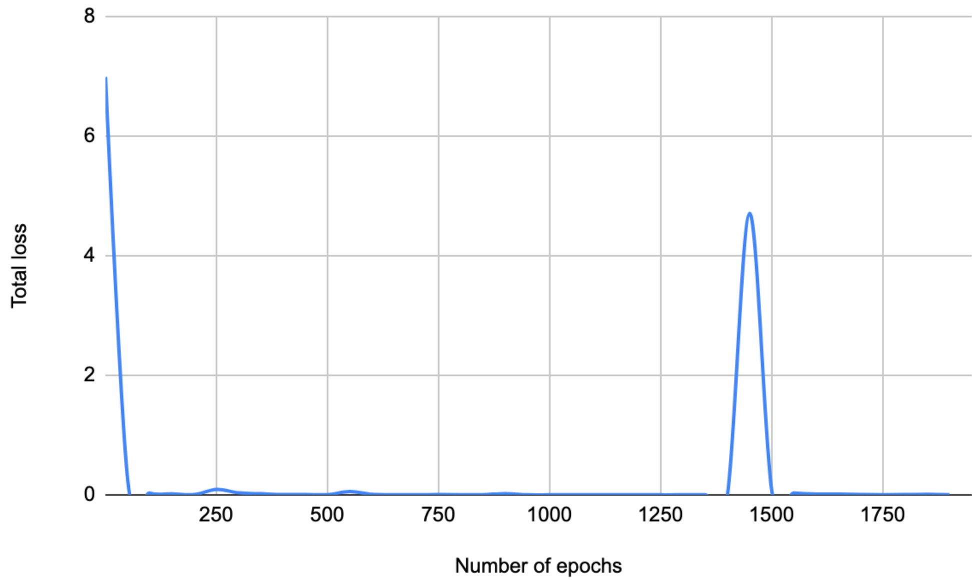
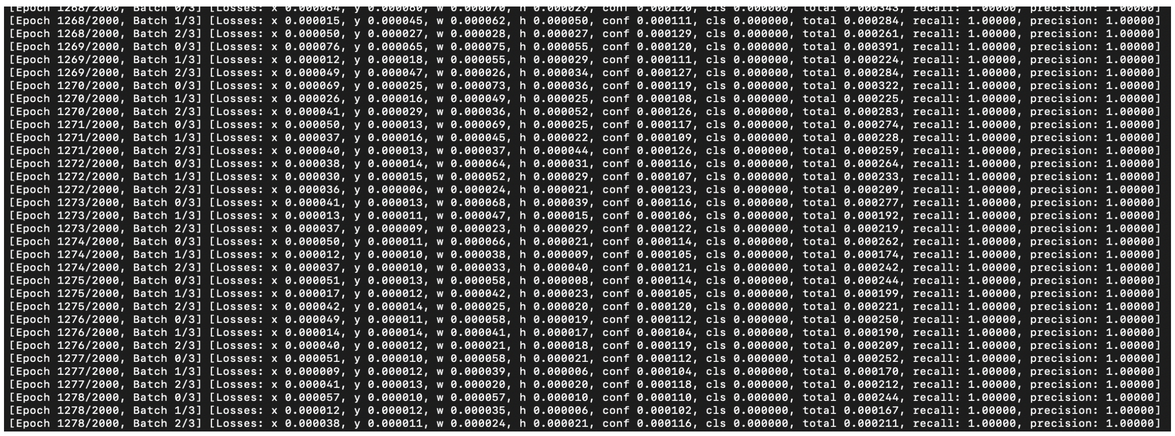


Figure 6. Loss curve for single finger pointing training.

As we can see through the loss curve in figure 6, the total loss decreases. At this point, the precision (when the loss was near 1000 - 1300 epochs) was roughly at 1.00000 as seen in figure 7, which means that the loss was minimal. Around the 1500th epoch or so, the loss drastically increased, before gradually decreasing again.



```
[Epoch 1268/2000, Batch 0/3] [Losses: x 0.000084, y 0.000060, w 0.000070, h 0.000029, conf 0.000120, cls 0.000000, total 0.000343, recall: 1.00000, precision: 1.00000]
[Epoch 1268/2000, Batch 1/3] [Losses: x 0.000015, y 0.000045, w 0.000062, h 0.000050, conf 0.000111, cls 0.000000, total 0.000284, recall: 1.00000, precision: 1.00000]
[Epoch 1268/2000, Batch 2/3] [Losses: x 0.000050, y 0.000027, w 0.000028, h 0.000027, conf 0.000129, cls 0.000000, total 0.000261, recall: 1.00000, precision: 1.00000]
[Epoch 1269/2000, Batch 0/3] [Losses: x 0.000076, y 0.000065, w 0.000075, h 0.000055, conf 0.000120, cls 0.000000, total 0.000391, recall: 1.00000, precision: 1.00000]
[Epoch 1269/2000, Batch 1/3] [Losses: x 0.000012, y 0.000018, w 0.000055, h 0.000029, conf 0.000111, cls 0.000000, total 0.000224, recall: 1.00000, precision: 1.00000]
[Epoch 1269/2000, Batch 2/3] [Losses: x 0.000049, y 0.000047, w 0.000026, h 0.000034, conf 0.000127, cls 0.000000, total 0.000284, recall: 1.00000, precision: 1.00000]
[Epoch 1270/2000, Batch 0/3] [Losses: x 0.000069, y 0.000025, w 0.000073, h 0.000036, conf 0.000119, cls 0.000000, total 0.000322, recall: 1.00000, precision: 1.00000]
[Epoch 1270/2000, Batch 1/3] [Losses: x 0.000026, y 0.000016, w 0.000049, h 0.000025, conf 0.000108, cls 0.000000, total 0.000225, recall: 1.00000, precision: 1.00000]
[Epoch 1270/2000, Batch 2/3] [Losses: x 0.000041, y 0.000029, w 0.000036, h 0.000052, conf 0.000126, cls 0.000000, total 0.000283, recall: 1.00000, precision: 1.00000]
[Epoch 1271/2000, Batch 0/3] [Losses: x 0.000050, y 0.000013, w 0.000069, h 0.000025, conf 0.000117, cls 0.000000, total 0.000274, recall: 1.00000, precision: 1.00000]
[Epoch 1271/2000, Batch 1/3] [Losses: x 0.000037, y 0.000016, w 0.000045, h 0.000022, conf 0.000109, cls 0.000000, total 0.000228, recall: 1.00000, precision: 1.00000]
[Epoch 1271/2000, Batch 2/3] [Losses: x 0.000040, y 0.000013, w 0.000037, h 0.000044, conf 0.000126, cls 0.000000, total 0.000259, recall: 1.00000, precision: 1.00000]
[Epoch 1272/2000, Batch 0/3] [Losses: x 0.000038, y 0.000014, w 0.000064, h 0.000031, conf 0.000116, cls 0.000000, total 0.000264, recall: 1.00000, precision: 1.00000]
[Epoch 1272/2000, Batch 1/3] [Losses: x 0.000030, y 0.000015, w 0.000052, h 0.000029, conf 0.000107, cls 0.000000, total 0.000233, recall: 1.00000, precision: 1.00000]
[Epoch 1272/2000, Batch 2/3] [Losses: x 0.000036, y 0.000006, w 0.000024, h 0.000021, conf 0.000123, cls 0.000000, total 0.000209, recall: 1.00000, precision: 1.00000]
[Epoch 1273/2000, Batch 0/3] [Losses: x 0.000041, y 0.000013, w 0.000068, h 0.000039, conf 0.000116, cls 0.000000, total 0.000277, recall: 1.00000, precision: 1.00000]
[Epoch 1273/2000, Batch 1/3] [Losses: x 0.000013, y 0.000011, w 0.000047, h 0.000015, conf 0.000106, cls 0.000000, total 0.000192, recall: 1.00000, precision: 1.00000]
[Epoch 1273/2000, Batch 2/3] [Losses: x 0.000037, y 0.000009, w 0.000023, h 0.000029, conf 0.000122, cls 0.000000, total 0.000219, recall: 1.00000, precision: 1.00000]
[Epoch 1274/2000, Batch 0/3] [Losses: x 0.000050, y 0.000011, w 0.000066, h 0.000021, conf 0.000114, cls 0.000000, total 0.000262, recall: 1.00000, precision: 1.00000]
[Epoch 1274/2000, Batch 1/3] [Losses: x 0.000012, y 0.000010, w 0.000038, h 0.000009, conf 0.000105, cls 0.000000, total 0.000174, recall: 1.00000, precision: 1.00000]
[Epoch 1274/2000, Batch 2/3] [Losses: x 0.000037, y 0.000010, w 0.000033, h 0.000040, conf 0.000121, cls 0.000000, total 0.000242, recall: 1.00000, precision: 1.00000]
[Epoch 1275/2000, Batch 0/3] [Losses: x 0.000051, y 0.000013, w 0.000058, h 0.000008, conf 0.000114, cls 0.000000, total 0.000244, recall: 1.00000, precision: 1.00000]
[Epoch 1275/2000, Batch 1/3] [Losses: x 0.000017, y 0.000012, w 0.000042, h 0.000023, conf 0.000105, cls 0.000000, total 0.000199, recall: 1.00000, precision: 1.00000]
[Epoch 1275/2000, Batch 2/3] [Losses: x 0.000042, y 0.000014, w 0.000025, h 0.000020, conf 0.000120, cls 0.000000, total 0.000221, recall: 1.00000, precision: 1.00000]
[Epoch 1276/2000, Batch 0/3] [Losses: x 0.000049, y 0.000011, w 0.000058, h 0.000019, conf 0.000112, cls 0.000000, total 0.000250, recall: 1.00000, precision: 1.00000]
[Epoch 1276/2000, Batch 1/3] [Losses: x 0.000014, y 0.000014, w 0.000041, h 0.000017, conf 0.000104, cls 0.000000, total 0.000190, recall: 1.00000, precision: 1.00000]
[Epoch 1276/2000, Batch 2/3] [Losses: x 0.000040, y 0.000012, w 0.000021, h 0.000018, conf 0.000119, cls 0.000000, total 0.000209, recall: 1.00000, precision: 1.00000]
[Epoch 1277/2000, Batch 0/3] [Losses: x 0.000051, y 0.000010, w 0.000058, h 0.000021, conf 0.000112, cls 0.000000, total 0.000252, recall: 1.00000, precision: 1.00000]
[Epoch 1277/2000, Batch 1/3] [Losses: x 0.000009, y 0.000012, w 0.000039, h 0.000006, conf 0.000104, cls 0.000000, total 0.000170, recall: 1.00000, precision: 1.00000]
[Epoch 1277/2000, Batch 2/3] [Losses: x 0.000041, y 0.000013, w 0.000020, h 0.000020, conf 0.000118, cls 0.000000, total 0.000212, recall: 1.00000, precision: 1.00000]
[Epoch 1278/2000, Batch 0/3] [Losses: x 0.000057, y 0.000010, w 0.000057, h 0.000010, conf 0.000110, cls 0.000000, total 0.000244, recall: 1.00000, precision: 1.00000]
[Epoch 1278/2000, Batch 1/3] [Losses: x 0.000012, y 0.000012, w 0.000035, h 0.000006, conf 0.000102, cls 0.000000, total 0.000167, recall: 1.00000, precision: 1.00000]
[Epoch 1278/2000, Batch 2/3] [Losses: x 0.000038, y 0.000011, w 0.000024, h 0.000021, conf 0.000116, cls 0.000000, total 0.000211, recall: 1.00000, precision: 1.00000]
```

Figure 7. Hitting maximum precision

Single Finger Pointing Testing

The testing set of images was run through the detector with the weights obtained from training to see if the network could immediately recognize the single finger point gesture. For most part, the network seemed successful to recognize images from the testing set under same conditions. Here are some of the correct predictions

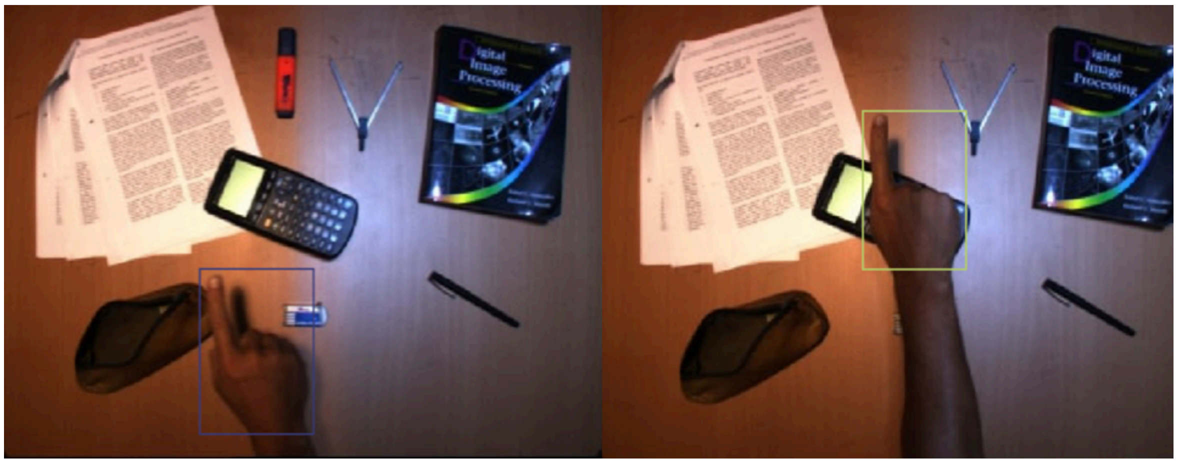


Figure 8. Correct predictions for Single Finger Pointing

Five Finger Pointing Training

The annotated data for this gesture was sent through the YOLO network for training. Training took place for roughly 2000 epochs. The losses for x, y, w and h were observed and total loss for the epochs was calculated using the average of losses for batches for that epoch. The weights from every 50 epochs were saved in checkpoints.

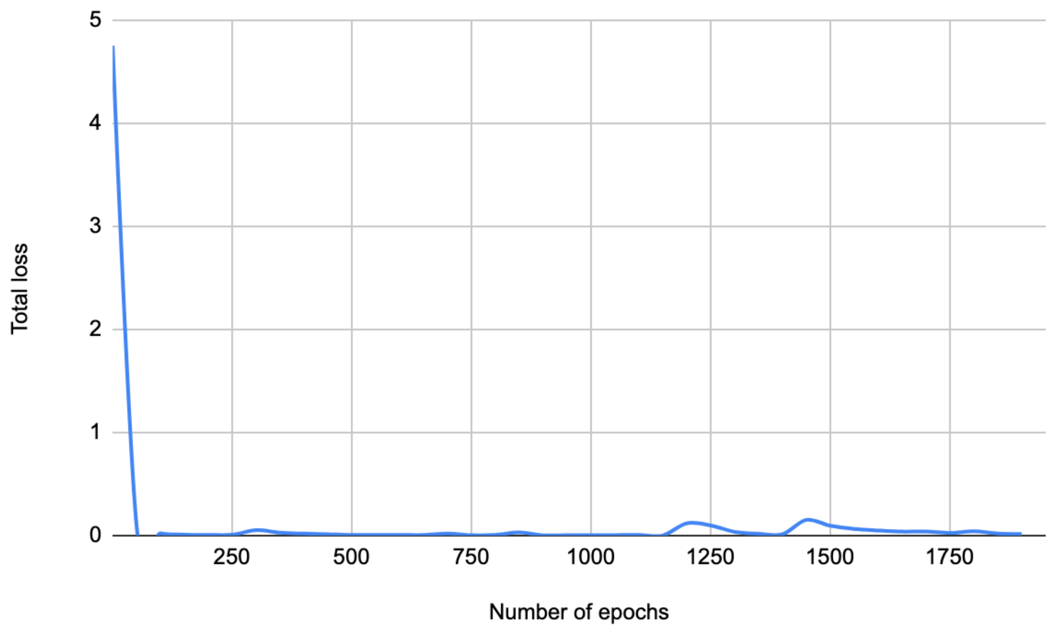


Figure 9. Loss curve for Five Finger Pointing Training

As we can see from figure 9, the loss is initially high but then gets much lower as the number of epochs increase. Hence, there are no massive loss increases in the training, giving us a smoother result. Still, between epochs 1200 and 1600, we do observe some increases in the total loss but compared to other training before, this is less frequent and smaller.

Five Finger Point Testing

The testing images were run through the detector and using the weights obtained through training to check if YOLO could immediately recognize the hand gesture. For most part, YOLO was successful in recognizing the gestures from the testing set.

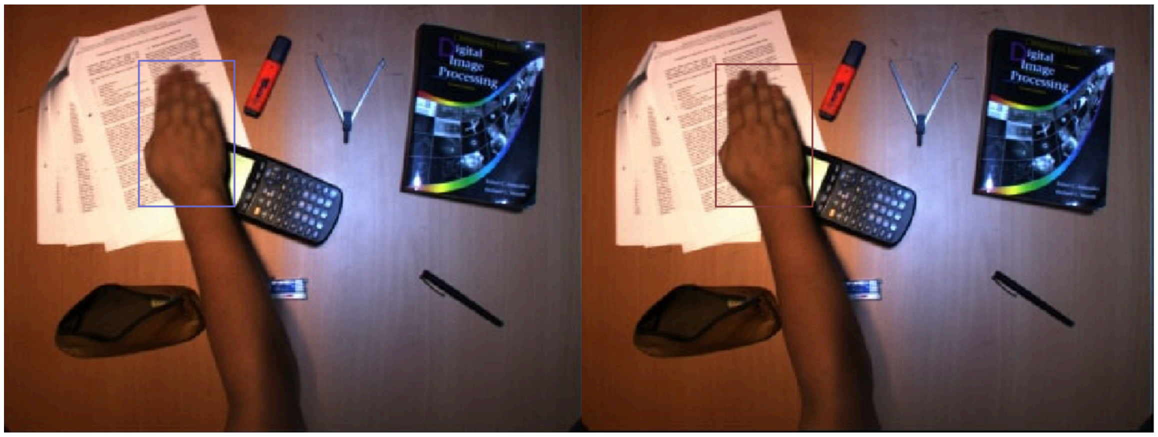


Figure 10. Five finger pointing predictions

As seen in figure 10, for YOLO correctly predicted the bounding boxes around the pointing signs.

Gig'Em Training

The annotated dataset for this gesture was sent through the YOLO network just like the previous trainings and the x, y, w, h losses were observed, along with the total loss and precision. The total loss for the epochs was calculated using the average of the total loss for every batch in that epoch. Weights were saved every 50 epochs.

The loss again starts extremely high for the initial epochs before getting lower and lower with progressing epochs. However, this time that the average total loss wasn't as low as compared to the other trainings, but there wasn't much of fluctuation, just one around the 750th epoch and a small one near the 1300th epoch.

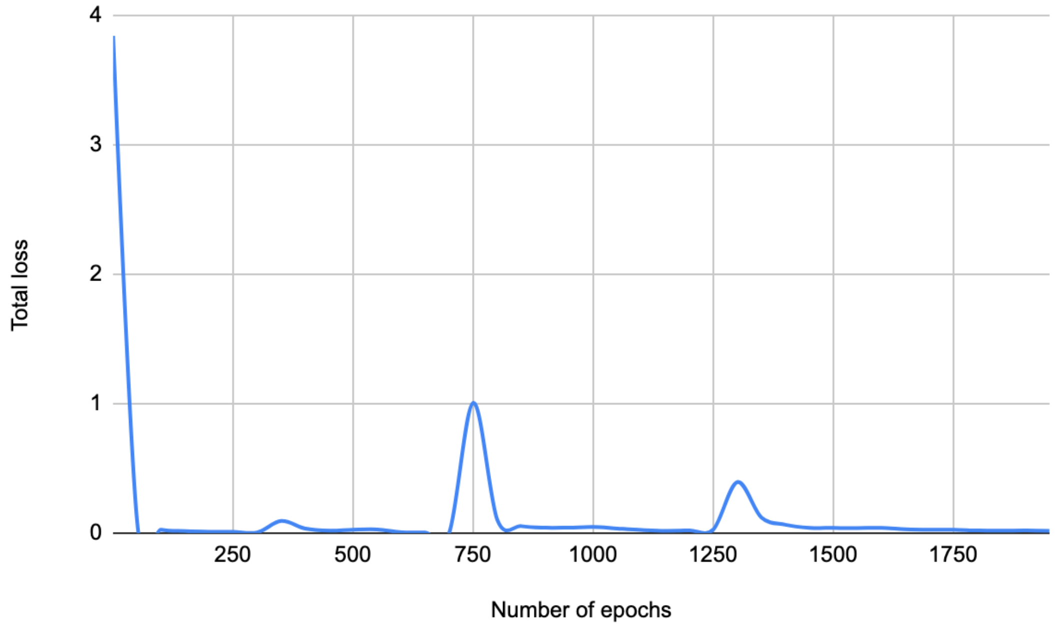


Figure 11. Loss curve for the Gig'Em training.

Gig'Em Testing

The weights from the training were then transferred to visualize the testing. Testing images were run through YOLO using the weights that were trained previously to see its performance. Figure 12 shows us that YOLO could successfully detect the Aggie Gig'Em sign well.

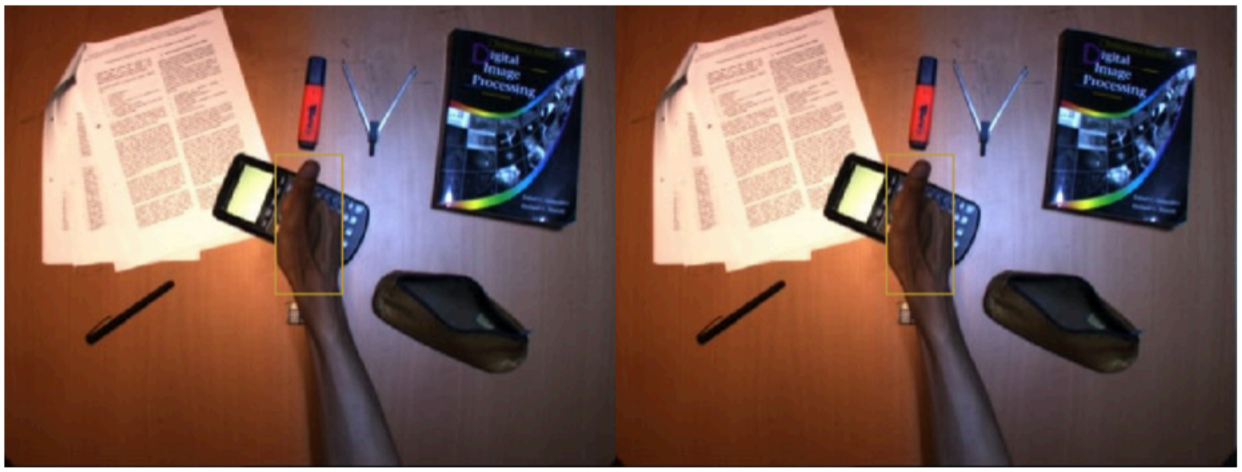


Figure 12. Predictions for the Gig'Em Sign

CHAPTER IV

CONCLUSION

This thesis was aimed at displaying the uses of YOLO in recognizing certain body language features and viewing its potential for speedy detections. YOLOv3 was an extremely fun neural network to work with and it was fun to see how it can be effectively used in recognizing certain gestures instantly after training the network for not a very long time. Thus, it proved to be very effective. There were certain misclassifications that occurred during the process and sometimes, YOLO failed to recognize certain images placed in extreme conditions, but overall, it is an extremely useful neural network for object detection.

Future Work

As of now, YOLO's uses have only been observed through testing images/video frames and feeding this information as the testing set to see it make predictions. Sometimes, there were some misclassifications and failed detections. So, we will be working to improve the weights even more in order to get better results. We would also like to see if YOLO can perform body language recognition in real-time, that is, if someone sits in front of a webcam and makes certain gestures, can YOLO be used to successfully identify those traits? We also plan on adding more sophisticated features to the training set, like multiple body language features in order to observe a certain human emotion/feeling, and see if YOLO is capable to perform that as well.

REFERENCES

[1] C. K. Goman, “10 Body Language Myths That Limit Your Success,” Forbes, 07-Nov-2017. [Online]. Available: https://www.forbes.com/sites/carolkinseygoman/2017/11/05/10-body-language-myths-that-limit-your-success/#5ae9fd551d2a. [Accessed: 05-Mar-2020].

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[3] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. CoRR, abs/1804.02767, 2018.

[4] AlexeyAB, “AlexeyAB/darknet,” *GitHub*. [Online]. Available: https://github.com/AlexeyAB/darknet. [Accessed: 05-Nov-2019].

[5] A. Mittal, A. Zisserman, P. H. S. Torr, Hand detection using multiple proposals British Machine Vision Conference, 2011

[6] C. Fotache, “Training Yolo for Object Detection in PyTorch with Your Custom Dataset- The Simple Way,” Medium, 10-Oct-2019. [Online]. Available: https://towardsdatascience.com/training-yolo-for-object-detection-in-pytorch-with-your-custom-dataset-the-simple-way-1aa6f56cf7d9. [Accessed: 05-Feb-2020].

[7] C. Fotache, “cfotache/pytorch_custom_yolo_training,” GitHub, 09-Oct-2019. [Online]. Available: https://github.com/cfotache/pytorch_custom_yolo_training. [Accessed: 07-Feb-2020].

[8] C. Fotache, “Object detection and tracking in PyTorch,” *Medium*, 10-Oct-2019. [Online]. Available: https://towardsdatascience.com/object-detection-and-tracking-in-pytorch-b3cf1a696a98. [Accessed: 05-Mar-2020].

[9] J. Navarro, *The dictionary of body language: a field guide to human behavior*. New York, NY: William Morrow, an imprint of HarperCollinsPublishers, 2018.

[10] “Pointing and command gestures under mixed illumination conditions: video sequence dataset,” *Pointing and command gesture dataset*. [Online]. Available: http://www-prima.inrialpes.fr/FGnet/data/03-Pointing/. [Accessed: 01-Mar-2020].