

**Department of Computing
Project (Technical Computing)
[55-604708]
2020/21**

| | |
|--------------------------|--|
| Author: | Jack Thickett |
| Student ID: | B8028388 |
| Year Submitted: | 2021 |
| Supervisor: | Sergio Davies |
| Second Marker: | Bob Steele |
| Degree Course: | BSc Computer Science |
| Title of Project: | Classification of hand gestures using a neural network and computer vision |

Confidentiality Required?

YES / **NO**

| |
|--|
| |
| |

I give permission to make my project report, video and deliverable accessible to staff and students on the Project (Technical Computing) module at Sheffield Hallam University.

YES / NO

| |
|--|
| |
| |

Acknowledgments

I would like to take a moment to express my gratitude to those who supported me during the undertaking of this project.

Firstly, to my supervisor, Sergio Davies who provided invaluable insight, encouraged critical thinking and pointed me in the right direction when required.

To my mentor, Ashley Collins-Richardson for sparking my interest in computer science, seeing my potential and offering every possible opportunity to further my development as a person and as a professional.

Finally, To my family for their unconditional support emotionally and financially.

Abstract

Throughout history hand gestures have been essential for human interaction. It is common practice to use sign language in situations where audibility is limited or unavailable. Some examples are communicating to the hearing impaired (NIDCD, 2019), executing covert military operations silently (Wilderness Arena, 2015) and interacting underwater with other divers (AquaViews, 2018).

As technology advances human communication is becoming more dependent on computing systems and more forms of Human Computer Interaction (HCI) are becoming increasingly available. With the use of machine learning and computer vision, it is possible to make HCI more accessible for the most vulnerable in our society.

With the exponential growth of data capture (Bulao, 2021) and the annually decreasing cost of computing (Hamilton Project, 2015), the major constraints on machine learning systems are being lifted. Machine Learning frameworks such as Sci-Kit Learn, TensorFlow and PyTorch have allowed more people to experiment with these techniques and drive innovation.

The aim of this project is to develop a viable solution for hand gesture recognition using these techniques.

Contents

| | |
|---|-----|
| Acknowledgments..... | II |
| Abstract | III |
| 1 Introduction | 1 |
| 1.1 Project Aims | 1 |
| 1.2 Objectives..... | 1 |
| 2 Research | 2 |
| 2.1 Existing Work..... | 2 |
| 2.2 Gesture Detection Approaches | 2 |
| 2.2.1 Image Classification..... | 2 |
| 2.2.2 Object Detection | 3 |
| 2.2.3 Image Segmentation | 3 |
| 2.2.4 Keypoint Estimation | 4 |
| 2.3 Feature Extraction | 5 |
| 2.3.1 Histogram of Oriented Gradients..... | 5 |
| 2.3.2 Principal Component Analysis | 6 |
| 2.3.3 Kernels..... | 6 |
| 2.4 Fundamentals of Machine Learning | 7 |
| 2.4.1 Loss Functions..... | 7 |
| 2.4.2 Optimisation Algorithms | 7 |
| 2.4.3 Learning Algorithms | 8 |
| 2.5 Training Methods..... | 12 |
| 2.5.1 From Scratch | 12 |
| 2.5.2 Transfer Learning | 13 |
| 2.6 Object Detection Architectures | 13 |
| 2.6.1 Region Based Convolutional Neural Networks..... | 13 |
| 2.6.2 You Only Look Once | 13 |
| 2.6.3 Single Shot Multi-Box Detector..... | 14 |
| 2.7 Data Augmentation..... | 14 |
| 2.8 Summary | 15 |
| 3 Design | 15 |
| 3.1 Functional Design | 15 |
| 3.2 Technologies and Frameworks..... | 16 |
| 3.2.1 Frameworks | 16 |

| | | |
|-------|--|----|
| 3.2.2 | Languages | 16 |
| 3.2.3 | Package Manager | 16 |
| 3.2.4 | Development Environment | 17 |
| 3.2.5 | Development Methodology..... | 17 |
| 3.3 | Hardware and Software Requirements | 17 |
| 3.4 | Dataset | 17 |
| 4 | Development and Testing..... | 18 |
| 4.1 | Feasibility | 18 |
| 4.2 | Development..... | 18 |
| 4.2.1 | Image Classifier | 18 |
| 4.2.2 | Object Detector..... | 18 |
| 4.2.3 | Object Detector with Cloud Computing | 19 |
| 4.3 | Training | 20 |
| 4.4 | User Interface (UI) | 20 |
| 4.5 | Evaluation Metrics | 21 |
| 4.6 | Optimisation..... | 22 |
| 4.7 | Real World Testing..... | 23 |
| 4.7.1 | Personal Testing..... | 23 |
| 4.7.2 | Volunteer testing | 25 |
| 4.7.3 | Additional Testing | 27 |
| 4.7.4 | Summary..... | 28 |
| 5 | Conclusion..... | 28 |
| 6 | Critical Reflection | 29 |
| 6.1 | Relating to application | 29 |
| 6.2 | Relating to the specification..... | 29 |
| 6.3 | Future Improvements | 30 |
| 6.4 | Ethical Concerns | 31 |
| 6.5 | Personal and Professional development | 31 |
| 7 | Bibliography | 33 |
| 8 | Appendix..... | 37 |
| 8.1 | Prototype 1 – Image Classification (Gesture 1) | 37 |
| 8.2 | Prototype 1 – Image Classification (Gesture 2) | 37 |
| 8.3 | Prototype 1 – Image Classification (Gesture 3) | 37 |
| 8.4 | Prototype 1 – Image Classification (Gesture 4) | 38 |

| | | |
|------|---|----|
| 8.5 | Prototype 1 – Image Classification (Gesture 5) | 38 |
| 8.6 | Prototype 2 – Object Detection from Scratch (Gesture 1) | 39 |
| 8.7 | Prototype 2 – Object Detection from Scratch (Gesture 2) | 39 |
| 8.8 | Prototype 2 – Object Detection from Scratch (Gesture 3) | 39 |
| 8.9 | Prototype 2 – Object Detection from Scratch (Gesture 4) | 40 |
| 8.10 | Prototype 2 – Object Detection from Scratch (Gesture 5) | 40 |
| 8.11 | Prototype 3 - BGR Input (Gesture 1) | 41 |
| 8.12 | Prototype 3 - BGR Input (Gesture 2) | 41 |
| 8.13 | Prototype 3 - BGR Input (Gesture 3) | 42 |
| 8.14 | Prototype 3 - BGR Input (Gesture 4) | 42 |
| 8.15 | Prototype 3 - BGR Input (Gesture 5) | 43 |
| 8.16 | Final application - RGB - Faster RCNN (Gesture 1) | 43 |
| 8.17 | Final application – RGB - Faster RCNN (Gesture 2) | 44 |
| 8.18 | Final application - RGB - Faster RCNN (Gesture 3) | 44 |
| 8.19 | Final application - RGB - Faster RCNN (Gesture 4) | 45 |
| 8.20 | Final application - RGB - Faster RCNN (Gesture 5) | 45 |
| 8.21 | Final application - RGB – SSD Brightness (Gesture 1) | 46 |
| 8.22 | Final application - RGB – SSD Brightness (Gesture 2) | 46 |
| 8.23 | Final application - RGB – SSD Brightness (Gesture 3) | 47 |
| 8.24 | Final application - RGB – SSD Brightness (Gesture 4) | 47 |
| 8.25 | Final application - RGB – SSD Brightness (Gesture 5) | 48 |
| 8.26 | SSD TensorBoard | 48 |
| 8.27 | Faster RCNN TensorBoard | 49 |
| 8.28 | SSD Brightness TensorBoard | 49 |
| 8.29 | Participant 1 - Round 1 | 50 |
| 8.30 | Participant 2 – Round 1 | 52 |
| 8.31 | Participant 1 –Round 2 | 54 |
| 8.32 | Participant 2 – Round 2 | 56 |
| 8.33 | Project Specification | 57 |
| 8.34 | Ethics Form | 60 |

1 Introduction

Artificial Intelligence (AI) is “the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings”. (Copeland, 2020) Computer vision is a subset of artificial intelligence that centres around a computer’s ability to comprehend the content of an image or video.

By training a machine learning algorithm using a collection of photographed hand gestures this project aims to deliver an application that can detect and classify hand gestures through a video input device. The solution should provide accurate feedback on the current hand gesture being displayed.

This report will investigate some of the approaches that can be applied to gesture recognition and their feasibility. The design and development chapters will document the steps taken to deliver the application and highlight the key decisions in its development. Finally, the critical reflection will evaluate the project’s success and discuss any further lines of investigation.

1.1 Project Aims

This project aims to deliver a desktop application that integrates a machine learning system capable of recognising fingers on a hand (1, 2, 3, 4 and 5) through a video input device. The system should give continuous feedback from a video input, facilitate reliable human computer interaction and in response to a gesture being shown, accurately detect and classify it.

The application should run on affordable hardware and recognise multiple gesture variants of a represented number.

Various neural network architectures will be evaluated to find the most suitable solution for this project. These architectures will be assessed in how they perform in real world scenarios, which will be achieved by statistical analysis and human feedback.

Techniques described in this report should apply to any set of static gestures given enough training data.

1.2 Objectives

1. Investigate how to detect a gesture through video input using computer vision and neural networks.
2. Develop a knowledge and understanding of the field of machine learning and artificial intelligence.
3. Research the most appropriate languages and frameworks for this application.
4. Collect sufficient data to train and develop a neural network powered program.
5. Develop an application that can detect gestures from a video input device and give feedback in a reasonable time.
6. Evaluate the performance of various architectures and select the most suitable approach.
7. Experiment with feature extraction algorithms to improve the accuracy of the application.

2 Research

There are many ways to approach the gesture recognition problem. This chapter will explore the current solutions and their feasibility for this project.

2.1 Existing Work

Data gloves are the classic approach to gesture recognition and are actively used in robotics and Virtual Reality (VR) (HaptX Inc., 2021). These applications require the detection of subtle movements which data gloves can provide; however, this level of detail is not required to meet the aims outlined in the project specification. Although data gloves are the most accurate form of gesture recognition, they are expensive, uncomfortable and reduce mobility (Rini Akmeiliawati, 2007). Due to the cost of these devices, it is not feasible for them to be used in this project.

Computer vision can address the problem of hardware costs because it can work with a single video input device. Webcams and personal computers are now widely available so using the hardware people already own reduces the cost of each system significantly, making this type of application more accessible to the general public.

Webcams are limited as they can only capture a 2D (2 Dimensional) perspective of our 3D reality, this means the depth of the environment is not captured. This can make it difficult to identify gestures that are not directly facing the video input device. Kinect by Microsoft attempts to address this problem using an infrared depth sensor to perform background subtraction, isolate the region of interest, and capture 3D information about the scene (Leroy D'Souza, 2011). Although this information would be valuable in identifying the gesture displayed, the extra data captured by the depth sensor would negatively impact speed at run-time. Use of a Kinect sensor would also require the purchase of additional hardware, making the application less affordable.

SignAll is a company harnessing computer vision to translate American Sign Language (ASL) into written text (SignAll, 2017). SignAll demonstrates that with the use of machine learning, communication between signers and non-signers can be achieved with a standard webcam. This technology interprets body language and facial expressions in addition to the gesture itself and deciphers the message. SignAll also use this technology to provide the means for people to learn sign language without the conventional methods of classroom teaching.

Results achieved by SignAll with just a webcam and a machine learning system show great promise for interpreting gestures at low cost. This project will attempt to achieve similar results with a single webcam and a smaller set of hand gestures due to financial and time limitations.

2.2 Gesture Detection Approaches

There are several approaches to tackling the problem of gesture recognition using computer vision, this section explores which is the most suitable for this project.

2.2.1 Image Classification

Image classification is the act of categorising an image based on its pixel values. This approach performs well in controlled environments however accuracy can suffer dramatically when there is

a change in lighting conditions or background (Maneela Jain, 2013). This is a common occurrence when using input from a webcam as the background and lighting often cannot be controlled.

Figure 2.1 shows the image classification prototype giving predictions at a high rate with moderate fluctuations in accuracy. The input of the image was reduced to the corner of the screen to capture the hand and not the user. Image classification provides correct predictions when used in reliable lighting conditions and less crowded environments. Further screenshots can be found at **appendices 8.1-8.6**.

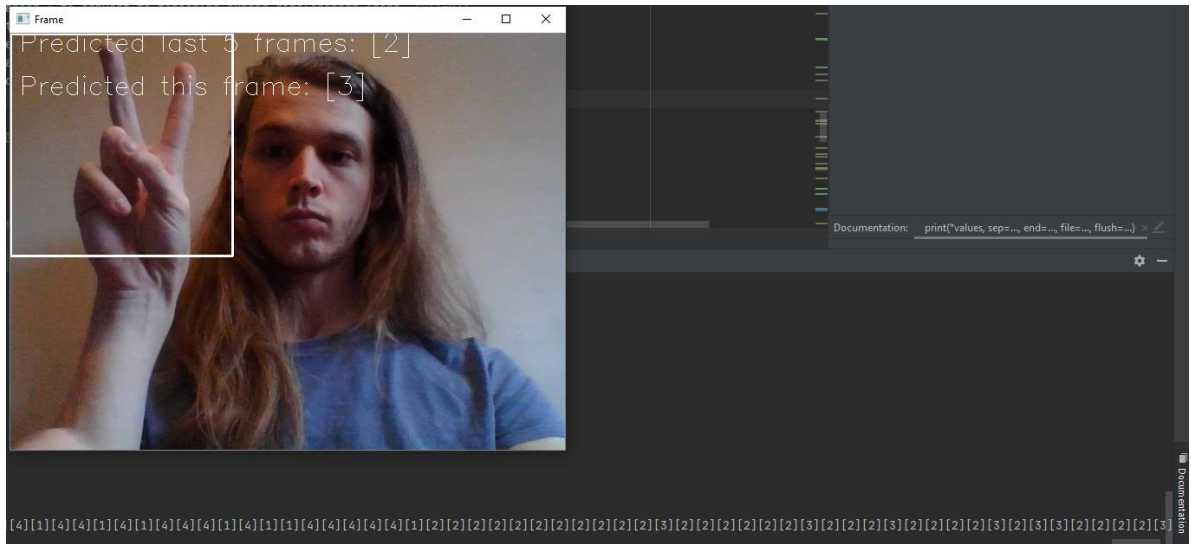


Figure 2.1 - Initial Image Classification Prototype

2.2.2 Object Detection

Object detection is an extension of image classification that localises the object within a bounding box (**figure 2.2**). This approach allows for more than one classification to be given per image as well as reducing the impact of background interference (Rodrigo Vershae, 2015).

Object detection is a more computationally demanding algorithm than image classification. For each object that is potentially detected, image classification must be performed on that subsection before a result can be calculated (Zhong-Qiu Zhao, 2018). Despite this drawback, object detection can run on affordable hardware however, this can comprise prediction speed so care must be taken when designing an appropriate implementation.

With the creation of the initial prototype, it was clear that object detection would be a more viable gesture detection method. This should provide more accurate feedback to the user. With this method, gestures could also be combined by showing two hands simultaneously facilitating additional gesture mapping for application commands.

2.2.3 Image Segmentation

Image segmentation is a further enhancement of object detection. It attempts to find the real edge of an object in an image by overlaying a mask. This results in a more granular boundary for each object in the image.

While image segmentation is more accurate than object detection, it is also more computationally demanding meaning it will take longer to train and detect objects. Because of this it is usually used where large computing power is available, one example being medical environments in Magnetic Resonance Imaging (MRI) scans (Yongxin Zhou, 2007).

Due to hardware restrictions around this project, it is not feasible to train and run a model with this approach. Model refers to a machine learning system in this context.

Figure 2.2 shows a visual representation of each category discussed.

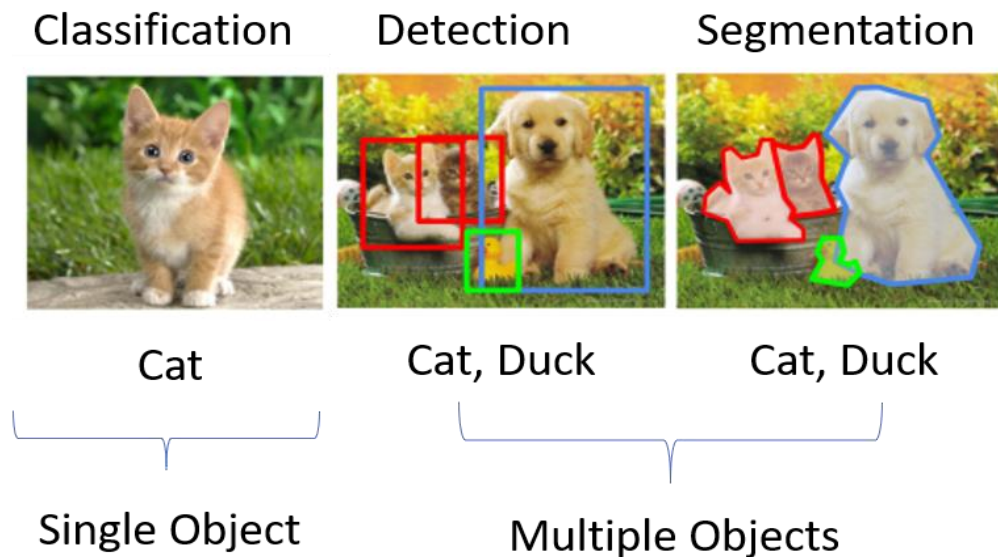


Figure 2.2 - Types of image classification (Kolungade, 2020)

2.2.4 Keypoint Estimation

Media Pipe use a keypoint approach to detect gestures (Media Pipe, 2020). A generic hand detector is used to get the coordinates of the hand and then 21 keypoints are estimated (**figure 2.3**) in a 2D space outlining the position of the hand.

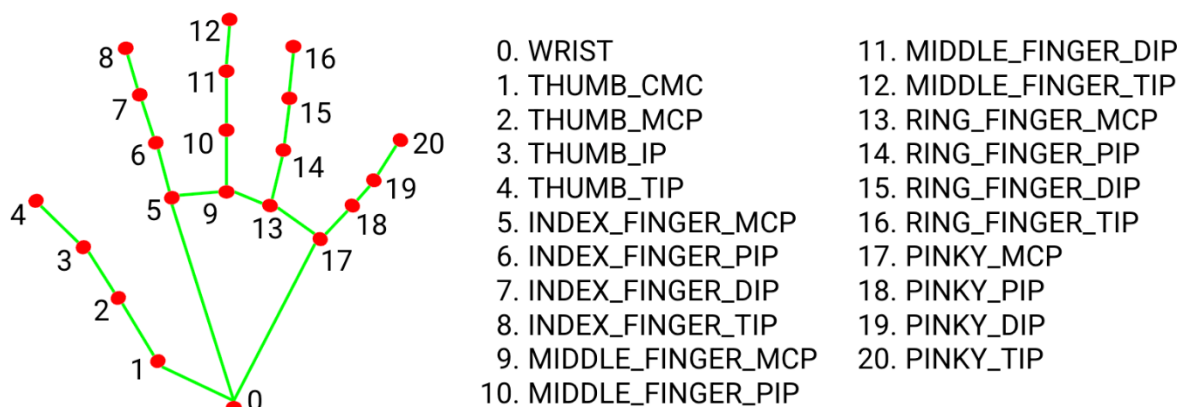


Figure 2.3 – Hand Keypoints (Media Pipe, 2020)

Because hands have a consistent shape, a neural network can be trained to learn the relationship between 2D keypoints and 3D keypoints (Christian Zimmermann, 2017). This network can then be

used to map the 2D keypoints into 3 dimensions (**figure 2.4**). This gives additional information about the configuration of the hand increasing accuracy when parts of the hand are obscured.

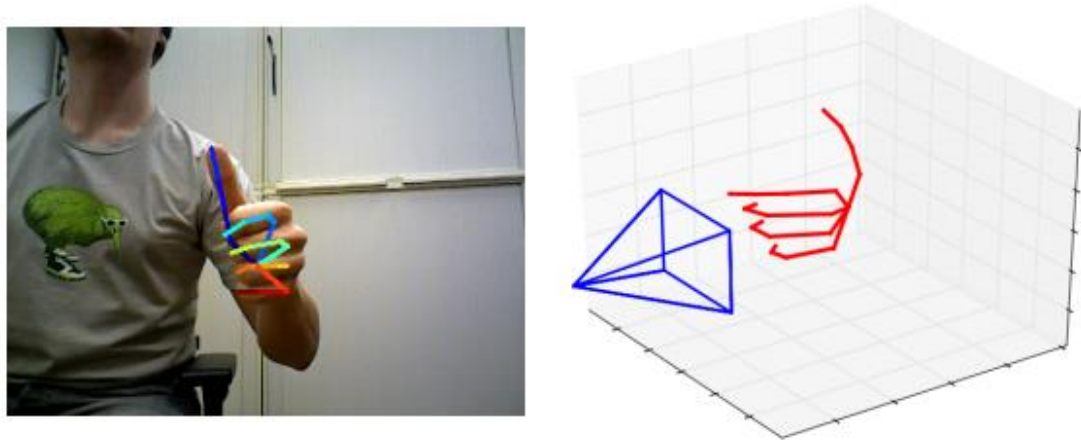


Figure 2.4 - Hand3D keypoint estimation (Christian Zimmermann, 2017)

This approach requires training 3 machine learning models each with their own dataset. It was not possible to implement this method due to the project's time constraints.

2.3 Feature Extraction

Feature extraction is the process of extracting certain features from an image. The output known as a feature map, emphasises relevant information and disregards redundant information (Triggs, 2005). Feature extraction algorithms are often used in computer vision to pre-process images so that they can be more easily identified by learning algorithms due to reduced noise and complexity.

2.3.1 Histogram of Oriented Gradients

Histogram of Oriented Gradients (HOG) works by looking at two adjacent pixels and assigning a value based on the difference between them (Ahmed, 2019). This reduces the potential for background colours and skin colours to interfere with detection as it is looking at the difference between adjacent pixels instead of their individual colour intensity. The feature maps generated can then be used by learning algorithms to perform tasks like image classification or object detection more easily. **Figure 2.5** shows HOG applied to an image from the dataset.

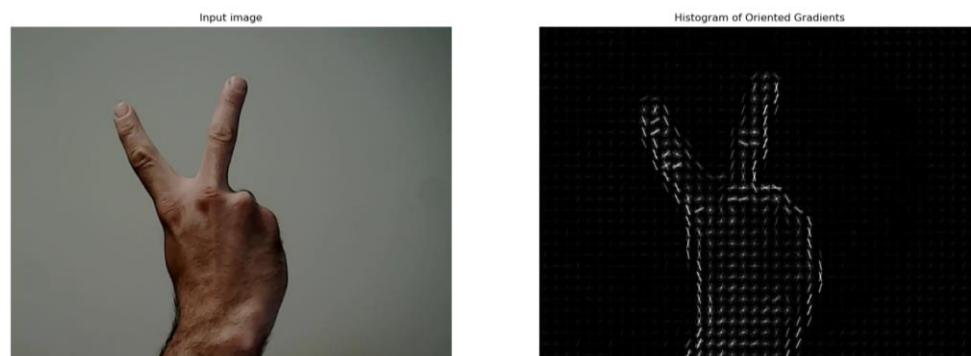


Figure 2.5 - HOG applied to an image from the dataset using OpenCV

2.3.2 Principal Component Analysis

Principal Component Analysis (PCA) is a feature extraction algorithm used to reduce dimensionality of a dataset while preserving as much of the relevant information as possible (Jolliffe, 1986). PCA determines the principal components of a dataset so that it is known which features to keep and which ones could potentially be dropped. PCA can be applied to image datasets as shown in **figure 2.6**. HOG and PCA have proven useful for eye detection with results of 98% accuracy with a support vector machine (Andreas Savakis, 2014).

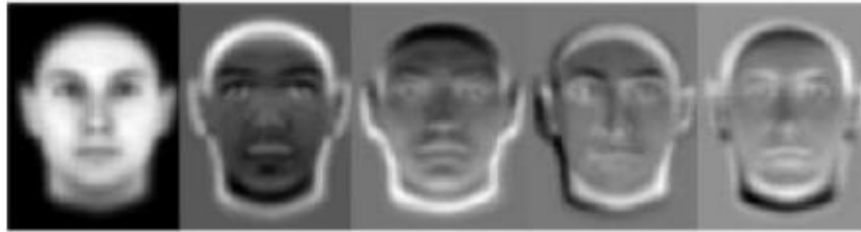


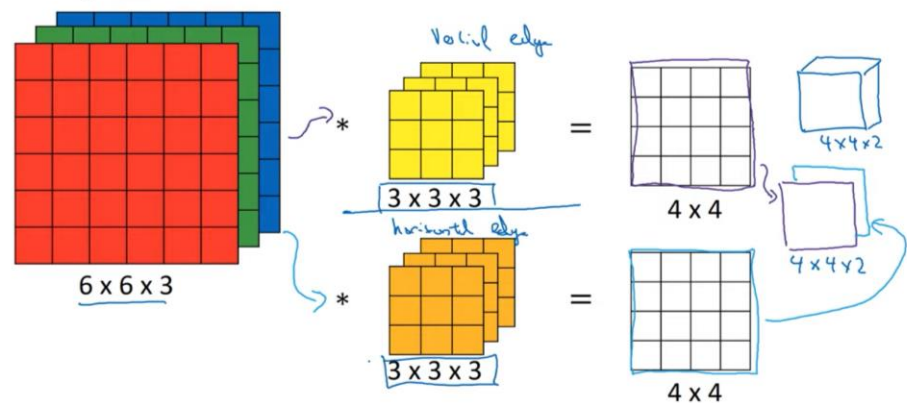
Figure 2.6 - Principal Component Analysis Image dataset (Fernando De la Torre, 2001)

2.3.3 Kernels

Kernels are widely used in image processing for example in Photoshop to apply gaussian blur or edge detection (Powell, 2015). The Kernel approach to feature extraction uses square matrices, most often odd ($n \times n$), with predefined values used to extract specific features. These kernels are convolved across the image multiplying the pixels by the corresponding kernel values, resulting in a feature map with features like edges or vertices highlighted. **Figure 2.7** shows how kernels convolved over a matrix can affect the dimensions of the output.

When working with 3D images (Red, Green, Blue (RGB) for example), kernels are applied through the entire depth of the image giving one output. Any number of kernels can be used on the same input, each producing one feature map which can be joined with the others to create a 3D feature map. This allows control over the dimensions of a feature map without losing significant information.

Multiple filters



Andrew Ng

Figure 2.7 - Kernel Application (Ng, 2019)

2.4 Fundamentals of Machine Learning

This section aims to provide an understanding of how machine learning systems work at a fundamental level. Knowing how machine learning systems operate at a fundamental level can provide valuable insight as to why a particular model (machine learning system) is not performing as expected.

2.4.1 Loss Functions

Loss functions are a fundamental part of machine learning. It is a method of calculating how well a model is performing on a dataset by comparing the predicted value against the actual value (Algorithma, 2018).

There are many different loss functions that fundamentally work the same way. **Figure 2.8** shows the formula for mean squared error. It calculates the sum of squared differences for all data points (y = actual value, \hat{y} = predicted value) then divides it by the number of data points to get the result.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

Figure 2.8 – Mean Squared Error

2.4.2 Optimisation Algorithms

The aim of a machine learning system is to minimize the loss function so that the predictions are as accurate as possible. Gradient descent, shown in **figure 2.9** is an optimisation algorithm that can be used to achieve this. By calculating the derivative of the loss function in relation to each parameter we can determine the best way to adjust the weights towards convergence (When loss is the minimum for the dataset) (Brownlee, 2016).

In **figure 2.9** α (alpha) is the learning rate hyperparameter that determines the size of each step taken by gradient descent and $J(\theta_0, \theta_1)$ represents the loss function.

$$\begin{aligned} &\text{Repeat until convergence} \{ \\ &\quad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \\ &\quad \text{for } (j = 1 \text{ and } j = 0) \\ &\} \end{aligned}$$

Figure 2.9 - Gradient Descent

For object detection, two loss functions are required. One (classification) for the confidence of the prediction and one (regression) for the accuracy of the bounding box (Zhong-Qiu Zhao, 2018). The formulae presented here can be taken care of by machine learning frameworks.

Optimisation algorithms can be applied after each data point (Stochastic), after each epoch (Batch) or after a manually assigned value (Mini-batch). Each has their own advantages and disadvantages. Since updating the weights can be a large operation, doing this after every example might be inefficient. Alternatively, updating the weights after each epoch can result in large fluctuations in loss (Patrikar, 2019).

Various optimisers are used today each with their own benefits and drawbacks. **Figure 2.10** shows some of them and their performance on the Modified National Institute of Standards and Technology (MNIST) digits dataset.

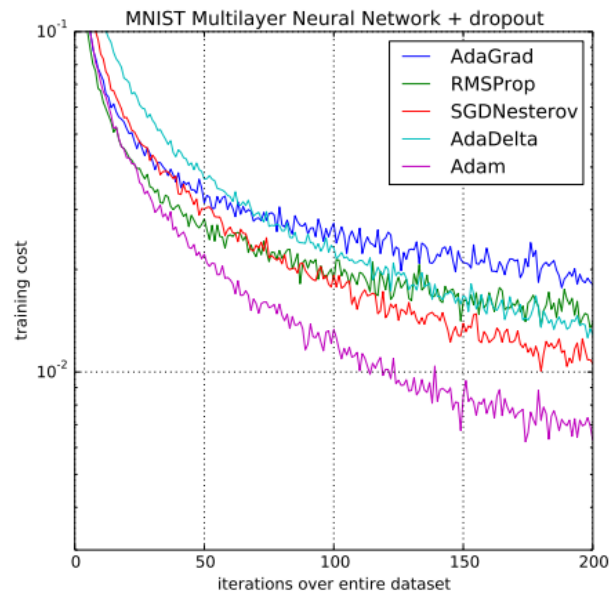


Figure 2.10 – Optimisation Algorithms on MNIST digits (Diederik P. Kingma, 2015)

2.4.3 Learning Algorithms

Many learning algorithms could be used for object detection. This chapter will focus on neural networks and some variations of them due to their suitability to computer vision problems.

2.4.3.1 Artificial Neural Networks

In 1958 Frank Rosenblatt proposed the single layer perceptron (Rosenblatt, 1958). The perceptron was a physical machine able to perform linearly separable binary classification but due to restrictions around computing power it was not until 1985 that the multilayer perceptron (now known as an Artificial Neural Network) was innovated and its potential realised. In the past few decades, we have seen its actualisation in self-driving cars, personalised sales recommendations and facial recognition.

Figure 2.11 shows the architecture of a single neuron which resembles the neurons found in our brain. This architecture remains much the same in modern neural networks but with the additional layers and neurons enabling the learning of more complex patterns.

Every neuron has a weight per incoming connection as well as a single bias value. The weighted sum of a neuron's inputs added to the bias is passed through an activation function (LIU, 2018). An activation function is a gate that determines if the signal gets passed to the next layer. Activation functions are assigned per layer of the network.

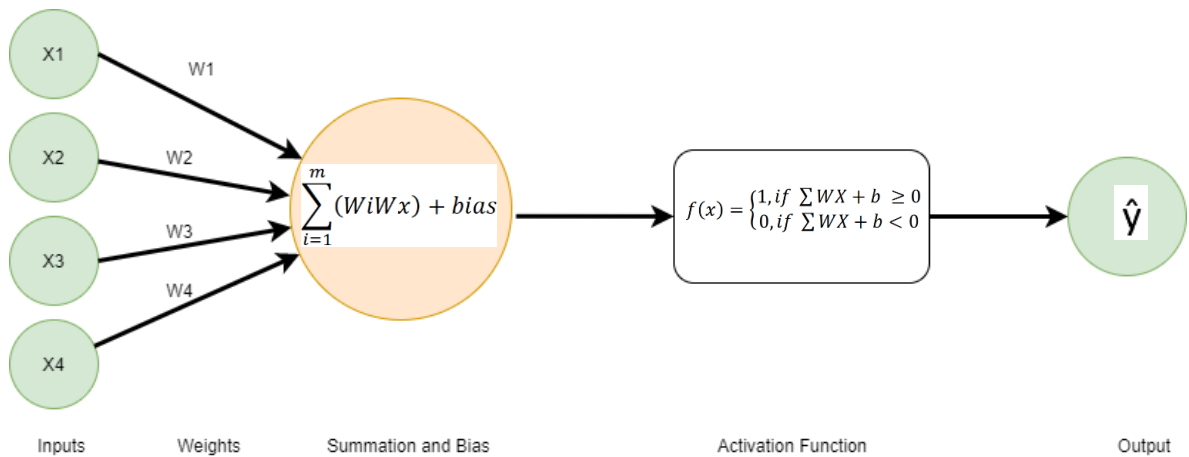


Figure 2.11 – Single Artificial Neuron

Figure 2.12 shows neurons organised into layers where data travels from left to right between them and outputs a result. Weights are adjusted during training using the loss function and an optimisation algorithm to encapsulate the patterns in the training data. After the result is produced the weights are updated in relation to the loss function during a backward pass. This is known as backwards propagation (Buscema, 1998).

Once trained, this type of network can be used to recognise previously learned patterns.

ANNs can be used for image classification tasks however they can become too large as each pixel maps directly to the input layer of the network. This exponentially increases the number of weights that must be updated (PAI, 2020). ANNs are suitable for small images but with larger images this approach becomes inefficient. ANNs are not suitable for object detection due to the higher computational demand over other methods of image classification.

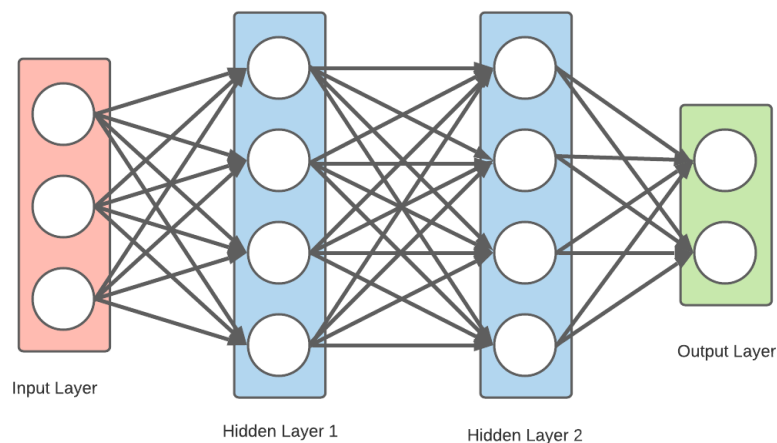


Figure 2.12 - Artificial Neural Network

The majority ANN calculations involve matrices and therefore the architecture is suitable for massive parallelisation. Each node in a layer can be processed in parallel allowing for more complex models to be deployed without compromising speed if a suitable Graphics Processing Unit (GPU) is used (Zhihao Jia, 2018).

ANNs have no feature extraction mechanism so pre-processing is often required to reduce the size of the input and highlight key features to produce an accurate result.

In this case the output classes are known meaning it is a supervised learning problem. It is often easier for a learning algorithm to spot patterns when the training data is labelled because it has guidance surrounding what patterns it is trying to identify. Neural Networks can be used for supervised or unsupervised classification and regression problems (Happiness Ugochi Dike, 2018).

ANNs are the basis for other deep learning algorithms that are more suitable to gesture recognition using computer vision.

2.4.3.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a variation of ANNs specifically designed for image processing. CNNs adopt a kernel approach to extract features from images. In addition to learning neuron weights, they adjust values in the kernels to improve feature extraction over time (S. Albawi, 2017). With this approach the number of parameters does not increase with the input size like with ANNs. This makes CNNs especially suitable to computer vision problems.

The main body of the CNN outputs a map of extracted features which is passed into several fully connected layers (**Figure 2.13**). These layers perform the classification and produce the output.

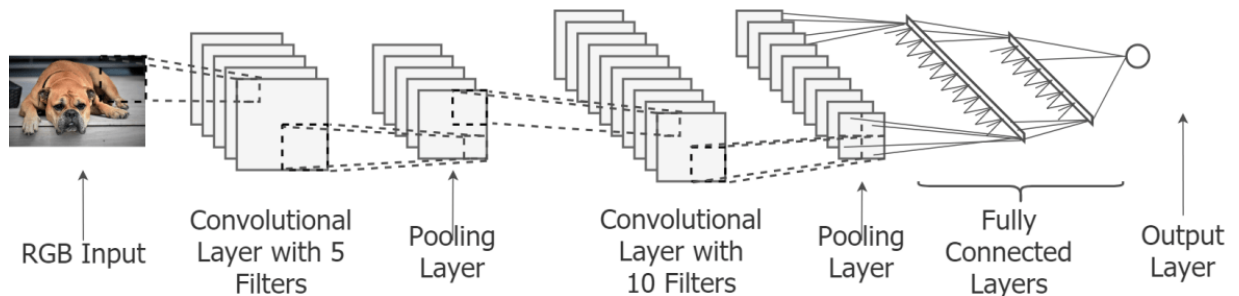


Figure 2.13 - CNN Architecture (Ganegedara, 2018)

Each convolutional layer consists of a set of trainable kernels that are convolved over the input. The number of filters directly correspond to the number of channels in the output of that layer. Chaining together layers this way this means that towards the end of the network, layers can identify high level features like objects, facial expressions, or animals. **Figure 2.14** gives a visual representation of earlier layers identifying edges and vertices with features becoming more complex as the network gets deeper.

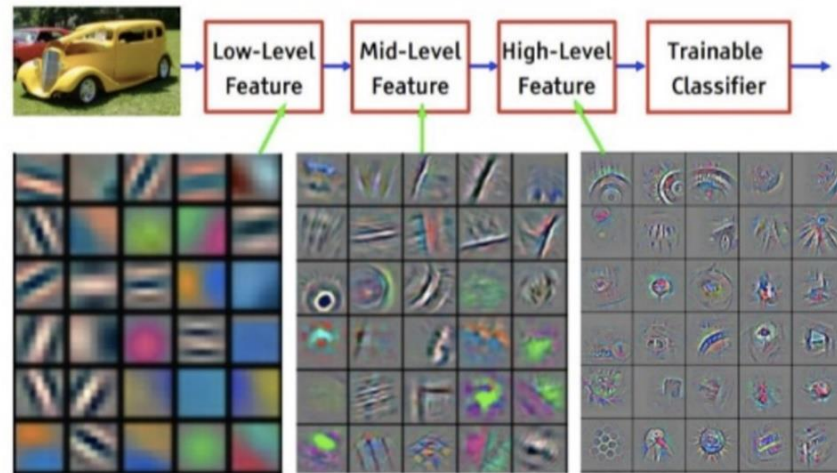


Figure 2.14 – Features identified at each layer of the CNN (Mahapattanakul, 2019)

Pooling is a technique used to decrease the size of a feature map. Pooling is often applied after a convolutional layer (**figure 2.13**). The most common pooling methods are max pooling and average pooling. Max pooling takes the highest pixel value in an (n x n) area to represent a reduced version (**figure 2.15**). This maintains the most significant features of an image and discards the rest resulting in a smaller feature map while minimising the loss of any important information. Average pooling is similar but instead of the maximum value it takes an average of all the pixels in the area.

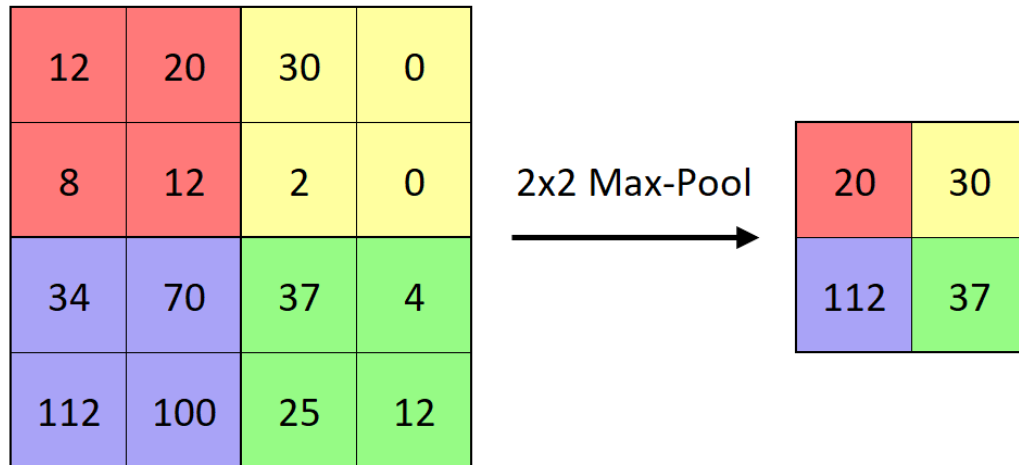


Figure 2.15 - Max Pooling

As well as reducing the dimensionality of a feature map, pooling can help reduce overfitting in learning algorithms due to its ability to remove noise. Max pooling often works better than average pooling in the context of convolutional neural networks (Chollet, 2017) (Page 129).

Pooling layers do not require any parameters as they perform a compression operation on the feature maps. Without pooling layers, the size of the feature maps would increase rapidly together with the amount of information the network needs to process. Keeping feature maps small with regular pooling layers allows the model to train and give predictions faster.

2.4.3.3 Residual Neural Networks

As deeper models are employed to capture increasingly complex patterns in data, the impact of the vanishing and exploding gradient problem becomes more prevalent. In 2015 Kaiming presented the residual neural network (RNN) which uses identity mappings to skip layers so that it does not degrade performance (He, 2015). Residual blocks ensure that at the very least a superfluous layer is ignored and does not have a negative impact on learning progress as well as minimising the impact on the performance of the network.

Figure 2.16 shows a residual block that can be applied over a layer of a network. Residual connections have enabled very deep neural networks with over 1000 layers to be trained with reasonable success (Ng, 2019).

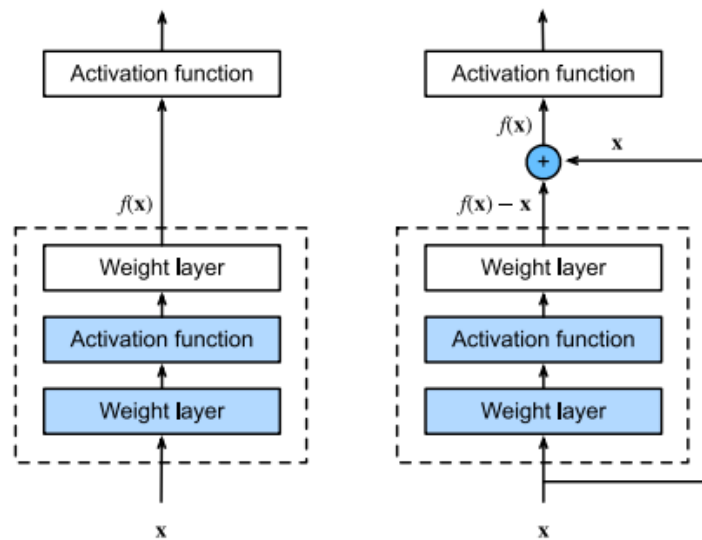


Figure 2.16 – Regular (left) and Residual (right) blocks (Aston Zhang, 2020)

ResNets have become very popular for object detection problems with complex features as they allow the use of much deeper neural networks while preventing overfitting (Mohammad Sadegh Ebrahimi, 2018). Almost all TensorFlow object detection models make use of residual connections due to their large size.

2.5 Training Methods

2.5.1 From Scratch

After developing the first prototype it became clear that image classification would not be sufficient for this project. This led to the decision that an object detection algorithm would be required.

Object detection models require significantly more data and computational power therefore training an object detection model from scratch was not feasible. It made more sense to use a pre-trained model and fine-tune it to the dataset.

2.5.2 Transfer Learning

Transfer learning allows a model trained on a large general dataset to be repurposed for a specific task by fine-tuning it to a custom dataset. The first few layers of a CNN often detect the same features in an image as seen earlier in **figure 2.14**. An example of transfer learning is how a musician can learn a second instrument faster than learning their first (Karl Weiss, 2016). This means that a pre-trained model can be trained to converge on a new dataset, with a small to medium amount of data, in fewer steps compared to training from scratch (Fuzhen Zhuang, 2019).

2.6 Object Detection Architectures

Most machine learning frameworks offer transfer learning with a variety of network architectures. This chapter will look at the benefits and drawbacks of several popular object detection architectures and their suitability for this project.

2.6.1 Region Based Convolutional Neural Networks

Region Based Convolutional Neural Networks (RCNN) work by using a Region Proposal Network (RPN) which takes in a feature map from a CNN and outputs regions of interest and the probability that there is an object present (Ross B. Girshick, 2013). This information is then passed on to a classifier that assigns the box a class.

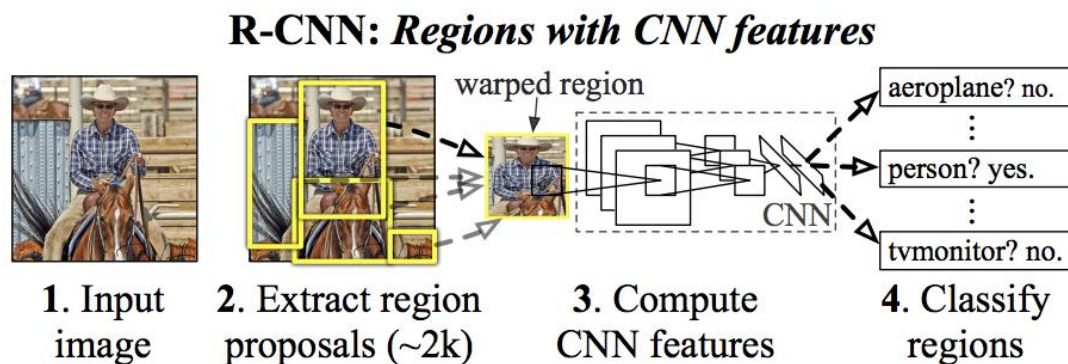


Figure 2.17 RCNN Architecture (Ross B. Girshick, 2013).

RCNN has proven to be accurate but slow in recognising objects in images. This is mainly due to each proposed region being processed sequentially (Gandhi, 2018). Newer iterations of RCNN including Fast RCNN and Faster RCNN address some of the issues in the original algorithm to improve training and inference speed.

Speed is essential in delivering useful feedback to the user in an appropriate amount of time. Faster RCNN may be suitable if the accuracy of faster models does not meet the required accuracy during testing.

2.6.2 You Only Look Once

You Only Look Once (YOLO) is an end-to-end object detection algorithm that computes detections in a single pass. YOLO works by dividing the image into a $(n \times n)$ grid and each cell tries to calculate

the bounding box and class probabilities of objects in that cell. While YOLO is fast, it is less accurate than Faster RCNN and often fails to detect small objects in the image due to spatial constraints enforced by the grid approach (Joseph Redmon, 2016). This should not be a major issue for this project as the gesture will often be the largest object in the image.

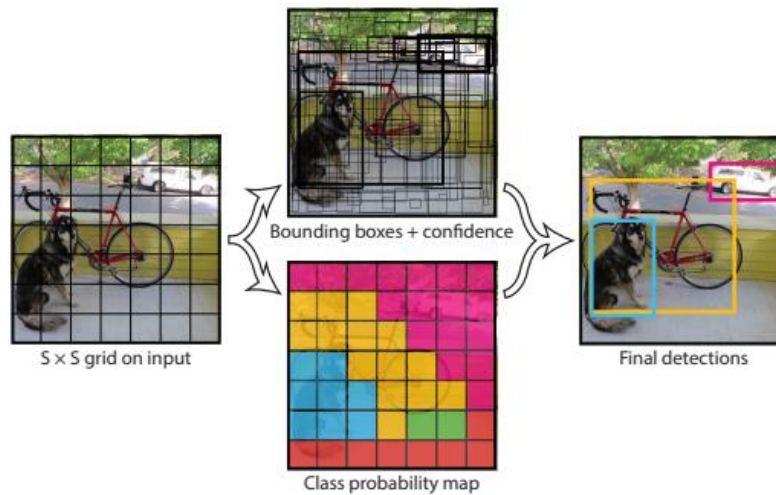


Figure 2.18 – YOLO (Joseph Redmon, 2015)

2.6.3 Single Shot Multi-Box Detector

Single Shot Multi-Box Detector (SSD) is an object detection architecture proposed by Wei Liu. SSD achieves object detection by using the single pass approach like YOLO while competing with the accuracy of algorithms like Faster RCNN (Wei Liu, 2015).

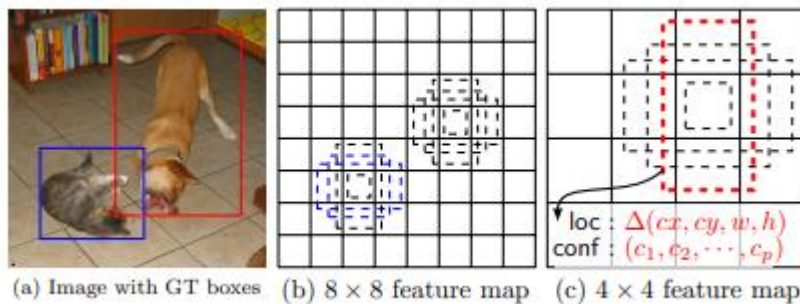


Figure 2.19 – SSD (Wei Liu, 2015)

SSD utilises a multi-reference approach with a set number of various sized bounding boxes that are proposed and evaluated using a confidence score on a potential object in the image. It then can adjust the best proposals for increased accuracy. A VGG16 (Visual Geometry Group) CNN is used to extract a feature map, then a convolutional layer detects objects in the image.

2.7 Data Augmentation

Data Augmentation is very useful when you have a limited amount of data or want to try and reduce overfitting. Data augmentation creates additional new examples from your dataset increasing size and variety (Gondhalekar, 2020). Augmentation steps could include random crop,

horizontal flip, introducing noise and adjusting brightness. Applying these transformations randomly to a dataset could give the neural network the edge it needs to perform better in difficult scenarios. These options must be applied to the dataset prior to training.

2.8 Summary

While an image classification approach has low computational requirements during training and inference, the accuracy of the initial prototype was not up to a suitable standard. Object detection should provide a better user experience by restricting the classification to a local area using bounding boxes.

CNNs will be the primary approach taken for the deliverable because they are designed with computer vision tasks in mind. CNNs reduce computational complexity and memory requirements in comparison to other learning algorithms applied to computer vision tasks.

Transfer learning is the preferred approach for this project as it requires less computational power and data to train a model compared to training from scratch. It also enables the use of tried and tested model architectures reducing development time and the chance of developer errors.

The SSD algorithm seems like the most appropriate solution to the gesture recognition problem due to its high accuracy and low inference time. YOLO also looks like it would perform well for this task due to its fast inference speed, but accuracy may not be high enough.

Data Augmentation options could be useful for adjusting the data set to help the network generalise to real world scenarios. Data augmentation can be easily implemented with most machine learning frameworks, but the training process must be repeated.

3 Design

This chapter will cover the various design decisions made when planning the deliverable.

3.1 Functional Design

The application should allow users to hold up a gesture to the webcam while inference is activated and give an output with a bounding box and a classification label. The application should work in mid to high light environments. Ideally results should be provided as close to real-time as possible.

This application should be able to compare various models and pre-processing steps and evaluate their performance in real world scenarios.

User Interface (UI) elements will be minimal as this application is only intended to test model's suitability for gesture detection.

Although a clear idea of what the software should do is essential, a typical software design specification was not suitable for this project. The focus is the performance of the trained model rather than the functionality the application needs to fulfil.

*Full software specification can be found at **appendix 8.33**.*

3.2 Technologies and Frameworks

3.2.1 Frameworks

TensorFlow was the initial framework investigated for this project. TensorFlow was created by Google and provides long-term support and good documentation. TensorFlow also provides the use of TensorFlow Object Detection API which can be used to perform transfer learning on a custom dataset.

PyTorch was also considered as a machine learning framework however after trying to build a couple of models, it seemed less intuitive than TensorFlow for beginners and did not have as many resources on transfer learning. This framework gives the programmer a lot of flexibility but would require more development time to complete an application.

OpenCV offers many tools to enable quick processing images which are useful for reducing noise from the input (Bradski, 2000). OpenCV can also be used to capture webcam input and display it to the user. Even though OpenCV has machine learning capabilities, it only offers a high-level interface whereas TensorFlow is more flexible in allowing the programmer to adjust the hyperparameters, training pipeline and model architecture.

Machine learning frameworks are very useful for developers to get projects up and running quickly. As a developer you are dependent on long term support for your implementations from the creators. When building a neural network from scratch, it is very unlikely to be affected by future updates as you are relying only on simple operations to perform the calculations. The benefits of using these frameworks usually outweighs the huge amount of time required to implement a machine learning algorithm from scratch.

3.2.2 Languages

For this project, producing multiple prototypes in a short space of time was essential for finding the best solution.

Python has fast development speed however is slower at run time because it is an interpreted language. Python is also the recommended language for TensorFlow by Google (and most other machine learning frameworks) meaning documentation in this language is more comprehensive.

C++ would be faster at run-time however development would likely take significantly longer.

For these reasons Python was chosen to be the main programming language.

3.2.3 Package Manager

Anaconda was used to manage the installation of required modules. It comes with Python installed and allows for the creation of additional environments. This made experimentation with other frameworks and libraries more convenient without interfering with previous installations. Anaconda also works with Preferred Installer Program (PIP) so any packages that are not available through anaconda could be installed using PIP.

3.2.4 Development Environment

PyCharm was used in conjunction with Anaconda to run development environments. PyCharm provides a simple interface for writing, running, and debugging python programs. It is designed specifically for Python and is free with the JetBrains education package. PyCharm provides very good code suggestions and a reliable environment to speed up application development.

Google Collaboratory (Colab) offers free cloud GPUs and Tensor Processing Units (TPUs) with a session limit of 12 hours. Training these relatively large models on a Central Processing Unit (CPU) proved to be an extremely slow process. Using Colab, multiple models can be trained and tested in a reasonable amount of time. The time limits on Colab were not a problem as the pre-trained models only required roughly an hour of training before they approached convergence.

3.2.5 Development Methodology

Due to the high level of uncertainty surrounding this project an ability to adapt is essential. Agile focuses on one small piece of functionality at a time, providing customers with updates regularly. Machine learning projects however, focus on the research and improvement of the model rather than releasing features. By combining agile with a prototyping approach, multiple prototypes could be built in a short amount of time allowing rapid experimentation and testing. This approach caters for uncertainty while maintaining focus on short terms goals to help break up development into more manageable chunks.

3.3 Hardware and Software Requirements

The application should run on any Windows 10 computer and a webcam will be required to gather video input. No additional software should be needed to run the application.

Even though the model has been trained prior to prediction, inference speed will still depend heavily on the CPU or GPU of the user's computer. To increase inference speed, a powerful GPU and additional configuration would be required. On an average power CPU, users should be able to get between 1-2 seconds inference speed resulting in a useable but not seamless experience.

3.4 Dataset

The dataset used to train the model was provided by Sergio Davies, Alexandr Lucas, Carlos Ricolfe-Viala and Alessandro Di Nuovo (Davies S., 2021). The dataset contains images of both human and robot hands making different gestures 1, 2, 3, 4 and 5. (Davies S., Human-Robot_Finger-Counting_Dataset, 2020)

The "Human_Fingers/xxxx_set/silhouettes" directories contain 1400 images (640 x 480) of human hands and accompanying annotation files which contain the coordinates of the bounding box as well as the classification. This is what will be used in to train the object detection model to recognise and localise these gestures. 200 images are kept separate for test set and 200 for the validation set leaving the training set with 1000 images.

All TensorFlow 2 object detection pre-trained models were trained on the Common Objects in Context (COCO17) dataset by Microsoft (Tsung-Yi Lin, 2014).

4 Development and Testing

4.1 Feasibility

Leveraging TensorFlow Object Detection API (Huang J, 2017) and cloud computing from Google Colab, it is realistic and achievable to develop an application powered by a neural network to recognise finger counting gestures and provide visual feedback to the user.

4.2 Development

The strategy for this project was a prototype-based approach. This allowed for fast trial and error to get a better idea of what would be required for the final deliverable. Several iterations of the project were built and tested to reach a place where the application worked as intended.

4.2.1 Image Classifier

The first prototype was a CNN image classifier trained from scratch on labelled hand images (**Figure 2.1**). The hope for this was that if trained enough, given a webcam input, the model would be able to predict what hand was being shown. This did not go to plan for several reasons. First the model was considering the whole image rather than just the hand region. This led to rapid fluctuations in the predictions to the point where you could not tell what the true prediction was. To address this only the top corner of the input was considered which helped somewhat but there was still lots of variation in the result.

To increase accuracy, the most frequent prediction of the last 5 frames was calculated. This can hinder prediction when changing between gestures but can assist when a gesture is held. Changing between gestures is not expected to be accurate as no gesture is being conveyed.

Even though this approach was very fast at giving predictions at run time, accuracy was too low. (**Figure 2.1**) (**Appendices 8.1-8.5**)

It became clear at this point that an image classification solution would not be sufficient for gesture detection. This realisation led to the investigation of object detection models and algorithms.

4.2.2 Object Detector

The second prototype was an object detection model, Faster RCNN. Although this is a much more appropriate model for the problem, it was trained on a CPU which was fine for the first prototype but would have probably taken weeks of training to reach convergence. Early stopping meant the model could be tested however results were poor, giving confidence scores of around 40% (**Figure 4.1**) (**Appendices 8.6-8.10**). This meant the model struggled to tell the difference between gestures and would fluctuate between predictions when the gesture was held.

At this point it was obvious that object detection models require much more computational power than image classifiers and inference speed is slower due to increased size of the model.

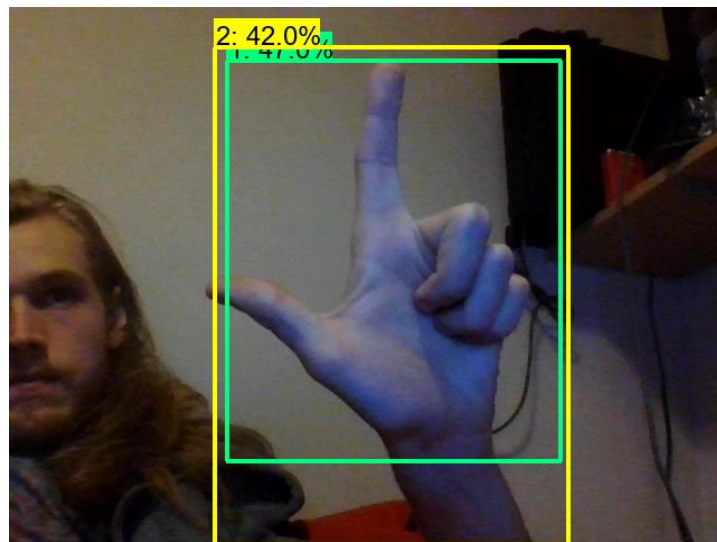


Figure 4.1 – Prototype 2 Object Detector trained on CPU showing 42% for 2 and 47% for 1.

4.2.3 Object Detector with Cloud Computing

The third and final prototype incorporated the power of cloud computing to train the model. Google Colab offers free GPU to anyone wanting to use it for research purposes. With the help of Google Colab and a Jupyter Notebook tutorial by Kapkar (Kapkar, 2015), training multiple object detection models in reasonable time was possible. Once training was complete, the model could be downloaded and run in a separate python project by reading webcam frames with OpenCV, feeding them to the model and overlaying the prediction over the frame. The extra training time meant the model was able to predict gestures at a much higher accuracy than the previous method (**Figure 4.2**) (**Appendix 8.11-8.15**).

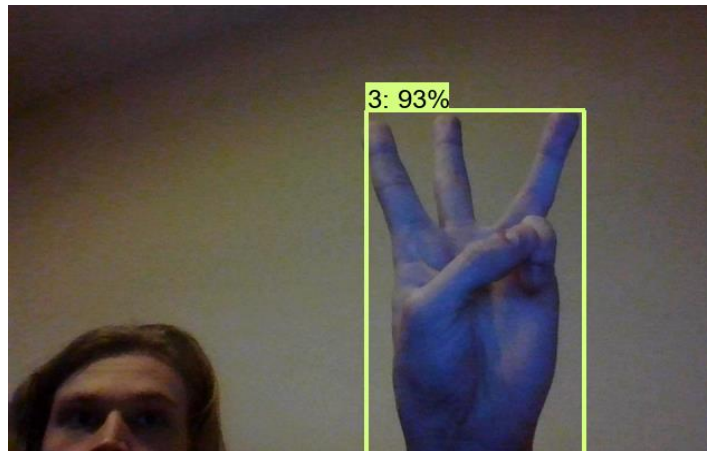


Figure 4.2 – Prototype 3 Object Detector trained with Google Collaboratory

Once the training pipeline had been established, different models were evaluated to find the most suitable one. From the selection of models available in TensorFlow's Object Detection model zoo "Faster R-CNN ResNet50 V1 640x640" (Faster RCNN) and "SSD ResNet50 V1 FPN 640x640" (SSD) were chosen as they seemed most appropriate due to their moderate size and good inference speed.

4.3 Training

When training a neural network there are many hyperparameters that determine how the model should be trained. These can be found in the config file that accompanies TensorFlow saved models. This section will explain some of the most impactful hyperparameter configurations.

Network weights are updated after every mini-batch. In this case the mini-batch size was set to 8 for SSD and 4 for Faster RCNN. Ideally when comparing these models, they should be the same size however an Out of Memory (OOM) error was thrown when trying to train Faster RCNN. To address this, the batch size had to be reduced so that training could resume.

The learning rate for both models was set to start at 0.013 and increase over the first 2000 steps to 0.039. This was the default setting and should prevent the model from exploding gradient during the early stages of training.

Both models were initially set to use the momentum optimiser with a gamma value of 0.9 (The amount of momentum). That was preserved to ensure the model trained as intended by the developers.

All models were trained for around 8000 steps. This seemed to be where the total loss started to level out. The requirements for stopping were that the model consistently evaluated below 0.2 total loss which all models achieved around the same time. TensorBoard screenshots of the training progress can be found at **appendices 8.21, 8.22, 8.23**.

Data augmentation was set to random horizontal flip and random crop as a baseline. This creates new images from the original dataset to introduce variation to the model without collecting more data. Horizontal flip helps the model detect mirror images and random crop dissuades the model from using the background to aid in the detection of the gesture.

4.4 User Interface (UI)

The Python Tkinter library was used to create a simple UI. The interface allows users to choose the model they want to load, enable pre-processing if needed and run inference using the chosen model. The UI is not intended to be appealing, it is only for switching between the available models during testing. **(Figure 4.3)**

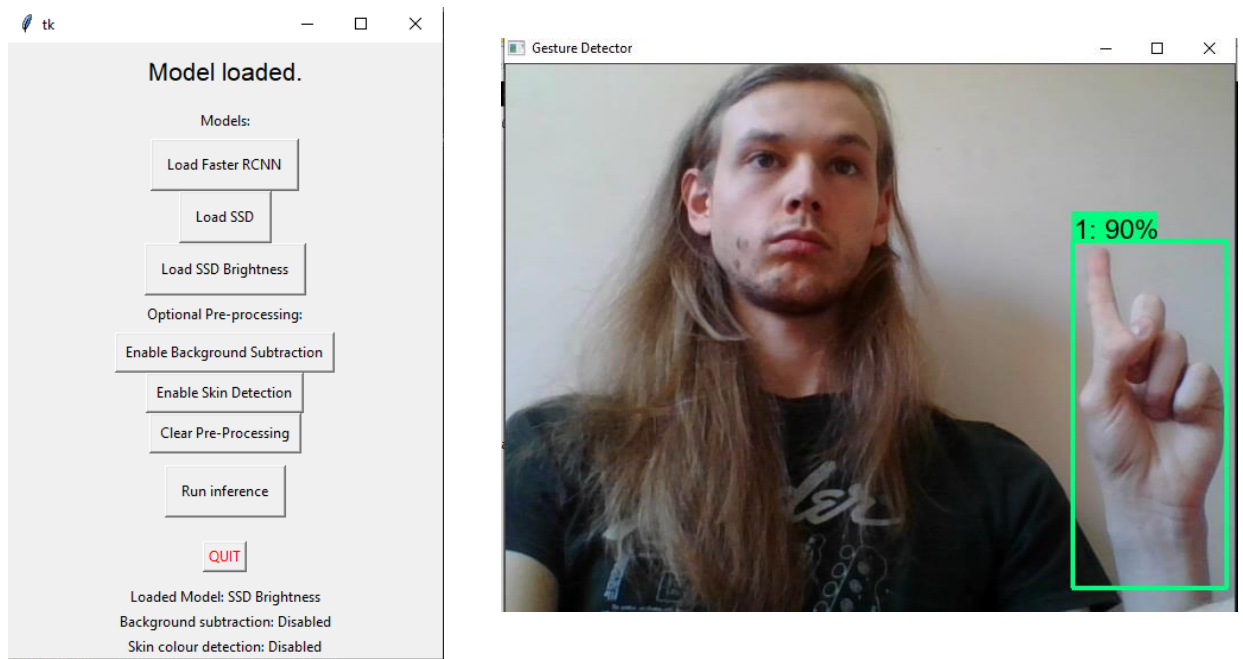


Figure 4.3 - Gesture Detector User Interface – Tkinter

4.5 Evaluation Metrics

Confusion matrices are one of the best ways to get an idea of how a model is performing on a dataset. Confusion matrices for object detection consider the closeness of the bounding box to the ground truth coordinates as well as the correctness of the prediction.

Confusion matrixes were generated by modifying “confusion_matrix_object_detection.py” from Sujith Kumar’s TensorFlow 2 object detection tutorial (Kumar, 2020).

The test set consisted of 240 images, 40 of each gesture and 40 that had no gesture present to test negative predication.

Intersection Over Union (IOU) threshold sets how close the box must be to the ground truth to be classed as correct. Confidence threshold determines how confident the prediction must be to be classed as correct. These parameters can be tweaked to highlight the differences between models. (Figures 4.4 and 4.5).

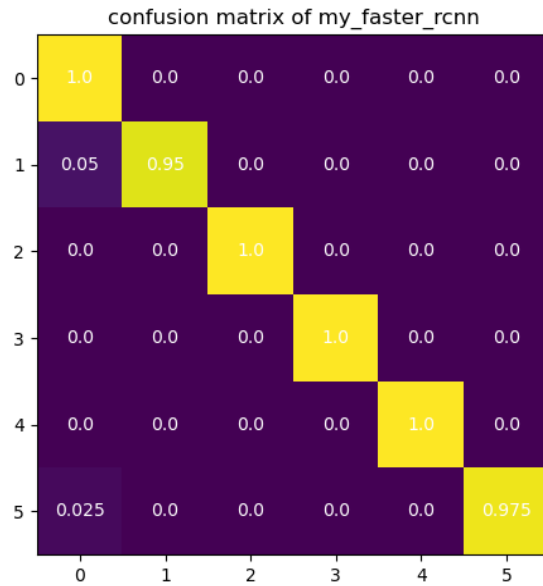


Figure 4.4 - Confusion Matrix Faster RCNN

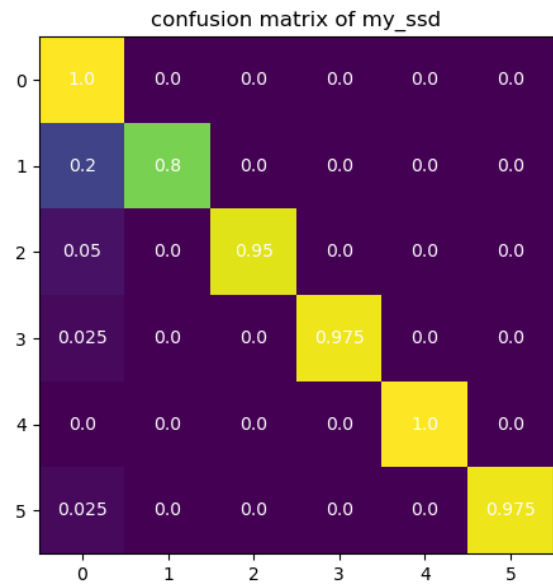


Figure 4.5 - Confusion Matrix SSD

IOU Threshold = 0.7 | Confidence Threshold = 0.8

Figures 4.4, 4.5 and 4.6 show that Faster RCNN performs better on most metrics than SSD when evaluated against the test set however the former is slower at giving predictions. For this project it is more important that the solution can give useful feedback in a reasonable amount of time rather than have a slight increase in accuracy.

| Metric Name | Formula | Faster RCNN | SSD |
|----------------------------------|-------------|-----------------|-----------------|
| Average Precision | $TP/TP+FP$ | 0.988372 | 0.961538 |
| Average Accuracy | $TP+TN/P+N$ | 0.9875 | 0.95 |
| Negative Prediction | $TN/TN+FN$ | 0.987998 | 0.956156 |
| Average Recall | TP/P | 0.9875 | 0.95 |
| Specificity | TN/N | 1 | 1 |
| Average Inference Time (seconds) | n/a | 2.5671244667422 | 1.4439247604144 |

Figure 4.6 – Metrics generated from confusion matrix.

T = True | F = False
P = Positive | N = Negative

- **Accuracy** is how many predictions were correct.
- **Precision** represents what percentage of positive predictions are correct.
- **Negative Prediction** is how reliable the model is at predicting negative examples.
- **Recall** represents what percentage of the positive examples were identified correctly.
- **Specificity** represents the correct classification of negative examples. Both models resulted in 1 for this metric but it should be noted that the negative images were just pictures of blank walls, not of objects attempting to throw off the models.

4.6 Optimisation

Both models were tested on the same hardware (Intel(R) Core (TM) i7-7500U CPU @ 2.70GHz, 16GB RAM). Both models turned out to be much slower at run-time than initially anticipated due to hardware restrictions. If the application ran on a more powerful CPU or GPU, then it is expected that inference time would be much faster but the difference between them should stay roughly the same. The simplest way to increase inference speed is to choose a smaller model or improve the hardware the model is running on.

While running the inference, a warning message was displayed that indicated the TensorFlow binaries were not optimised for the CPU instruction set being used.

```
Your CPU supports instructions that this TensorFlow binary was not
compiled to use: AVX2
```

Advanced Vector Extensions (AVX2) are CPU operations designed for dot matrix multiplication (Arrows, 2020). Utilising the correct TensorFlow binaries would mean that the library can make use of these CPU operations increasing performance on CPUs with the same instruction set. However, this reduces portability to CPUs with other instruction sets, meaning it might not work on other CPUs during volunteer testing, for this reason AVX2 was not used.

4.7 Real World Testing

This section tests the model in a real-world situation rather than just on the test set. Potential improvements are implemented or discussed where implementation was not possible.

4.7.1 Personal Testing

Testing the models in a real environment showed a heavy dependence on lighting of the room. Experimenting with lighting conditions showed that shining a light at the background made it easier for the model to detect a gesture. This probably helps because it increases the contrast between the hand and the background. This can be seen in the demo video at 4:00 or **figure 4.7**.

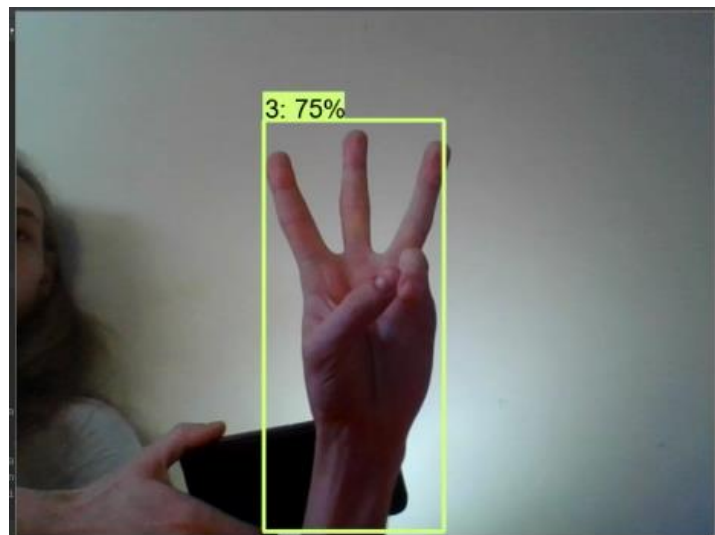


Figure 4.7 – Experimenting with Backlighting

To try and address the lighting issue, the SSD model was re-trained with a random brightness augmentation. This helped improve accuracy of the model against the test set which can be seen

in the confusion matrix 'SSD_Brightness' (**Figure 4.8**). This also helped the model correctly detect the gesture in personal real-world testing when lighting was not optimal, although this is hard to measure without quantitative feedback. Volunteer testing will be conducted to verify these predictions.

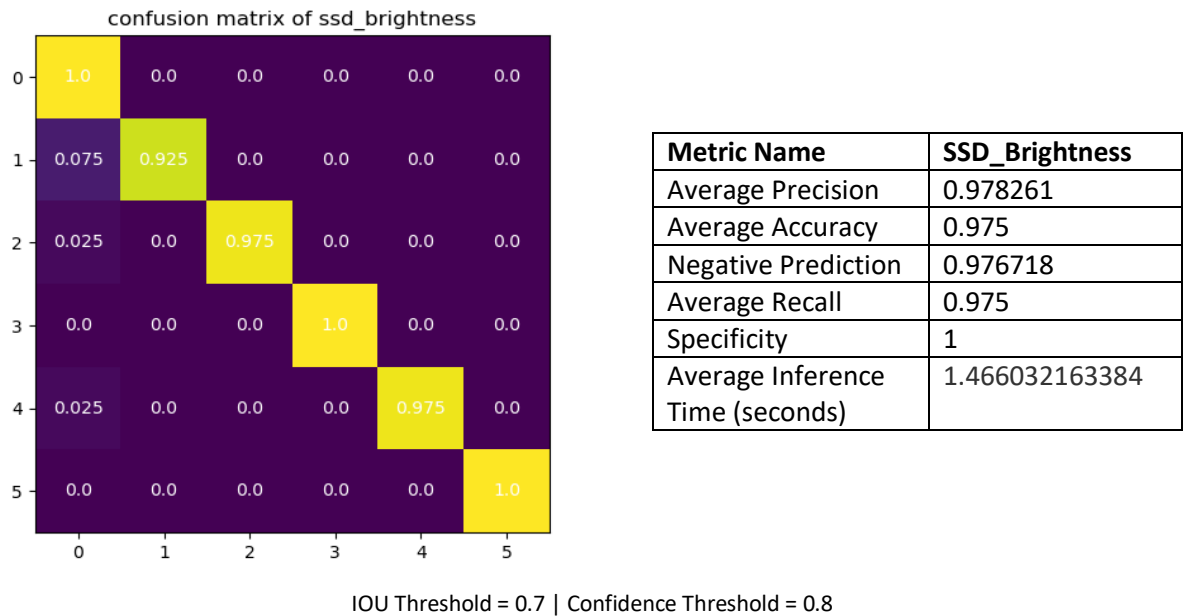


Figure 4.8 – SSD with random brightness

4.7.1.1 Background Subtraction

Since the lighting issue was not fully addressed with the SSD_Brightness model, background subtraction was implemented as an optional feature toggle. This feature takes a snapshot of the current frame and subtracts it from future frames. When a hand is introduced, the background is subtracted leaving the hand highlighted so the model can identify that a hand is present and what kind of gesture is being held.

This feature is not the best for user experience as it requires constant monitoring of the output to ensure only the hand is highlighted. The subtraction frame must also be reset if any other parts of the background or the user move.

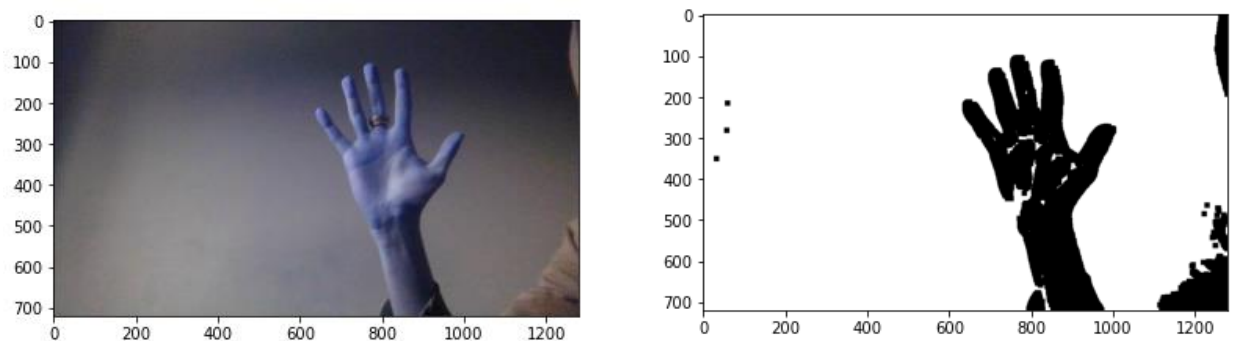


Figure 4.9 – Background subtraction toggle

4.7.1.2 Skin Colour Detection

To reduce the impact of background interference, skin colour detection was implemented. This works well when the background contrasts against the user's skin colour. Using this method, the face is also detected. This feature will not work as well for users with a different skin pigmentation to the developer as it has not been configured to do so.

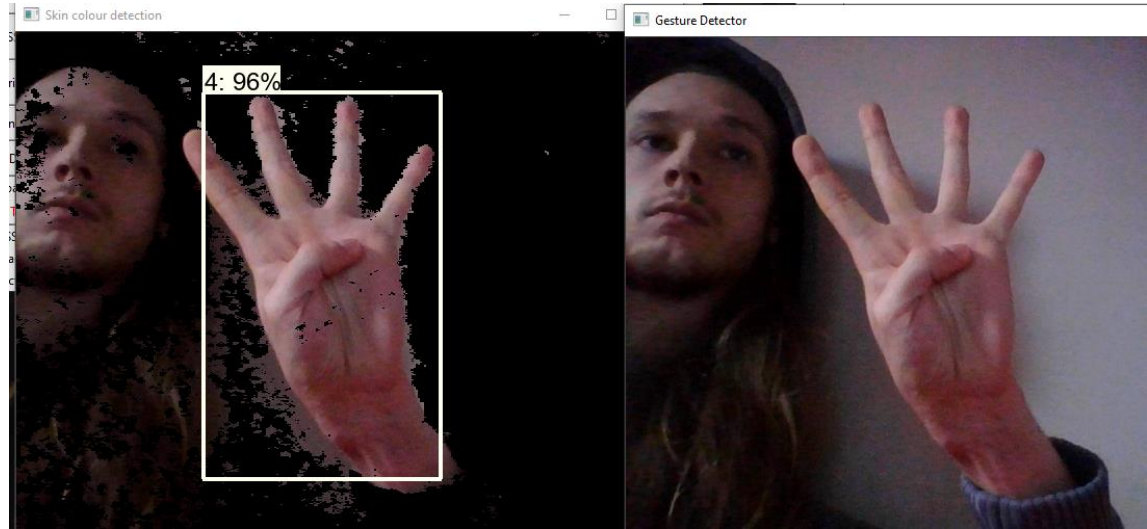


Figure 4.10 – Skin Detection

4.7.2 Volunteer testing

The purpose of volunteer testing was to get feedback on how the models felt to someone who had not been involved in development and to compare the feedback against personal testing of the application. Discovering bugs and exploring ideas for improvements are also areas of interest.

Participants were required to give a score out of 10 for the confidence and another for the accuracy of the bounding box. Also experimenting with pre-processing steps if required.

It should be noted that inference speed was around twice as fast for participant 1 due to a better CPU, which may be why they were able to get better results. More frames processed means more chance of a correct prediction.

4.7.2.1 First Round Testing

In the first round of testing all models when no pre-processing was enabled performed disappointingly. **Figure 4.11** shows SSD_Brightness with background subtraction was the clear winner. Full feedback can be found at **Appendices 8.24-8.25**.

| Model | Pre-processing | Avg Confidence | Avg Bounding Box |
|-------------|----------------|----------------|------------------|
| Faster_RCNN | None | 0.9 | 1 |
| | Background Sub | 0 | 0 |
| | Skin Detection | 0.8 | 0.9 |
| SSD | None | 0 | 0 |
| | Background Sub | 0 | 0 |
| | Skin Detection | 0 | 0 |

| | | | |
|----------------|----------------|-----|-----|
| SSD_Brightness | None | 3.5 | 4.3 |
| | Background Sub | 7.7 | 8.6 |
| | Skin Detection | 2.1 | 2.2 |

Figure 4.11 – Average Results Out of 10

All participants commented on the speed of the application, saying it used a lot of CPU and slowed their computer down. Participant 2 suggested doing the heavy processing on a server and stream it back to the user's device. Considering how slow the models run of an average CPU, this would likely have a speed improvement but would also incur a financial cost.

Participant 1 suggested brightness and contrast sliders, giving more control to the user when combating bad lighting and environmental interference.

While trying to implement the sliders suggested by participant 1, it was discovered that the input to the model was implemented incorrectly being BGR instead of RGB. This turned out to be the main reason why the detector was not performing in well-lit environments and seemed to perform better in the dark. This can be seen best in **appendices 8.11-8.15** where the gesture was appearing on the input with a blue tint.

Examples of the gesture detector working correctly can be found at **appendices 8.16-8.25**.

After fixing the RGB bug, personal testing indicated that the models performed much better when no pre-processing was enabled. The brightness (Gamma) and RGB sliders were still implemented but did not have as much of an impact once the input was in the correct format. (**Figure 4.12**)

Pre-processing steps for all models remained the same, only the standard input had been effected.

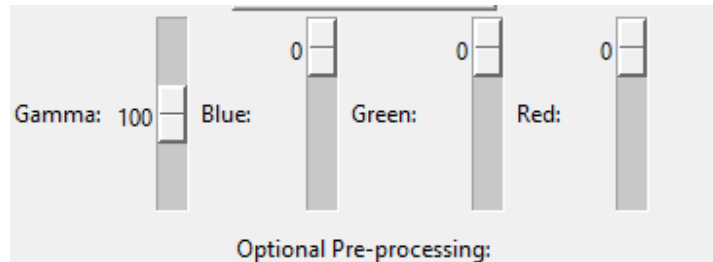


Figure 4.12 – Input Sliders – Tkinter

4.7.2.2 Second Round Testing

A second round of testing was conducted using the same participants, focusing on no pre-processing to confirm the new improvements. Results can be found at **appendices 8.26-8.27**.

Figure 4.13 shows average scores calculated from the feedback forms.

| Model | Pre-Processing | Avg Confidence | Avg Bounding Box |
|----------------|----------------|----------------|------------------|
| Faster_RCNN | None | 7.5 | 7.8 |
| SSD | None | 5.4 | 5.1 |
| SSD_Brightness | None | 7.4 | 8.7 |

Figure 4.13 – Average results - Default settings

Faster_RCNN detected gestures with high confidence and a good bounding box score. If more processing power was available, this would be a very accurate model, but the speed lets it down when used on a CPU.

SSD performed better than previously but still struggled most likely due to poor lighting conditions.

SSD_Brightness had similar accuracy to Faster_RCNN but ran much faster, meaning the user could adjust the gesture in reaction to the feedback and get a successful detection more quickly. The only difference between this model and SSD is the brightness augmentation which helped the model perform in less-than-optimal lighting conditions. The bounding boxes also received a higher average score.

The RGB sliders did not prove useful in testing but were necessary to find a bug detrimental to the application's success. The most useful was the gamma slider, which helped reduce brightness when the webcam was highly exposed to light.

4.7.3 Additional Testing

With standard positioning of the webcam, the user's body and face are often in shot. This can interfere with the model making it predict gestures on clothing or faces. Usually, when a gesture is introduced, this takes precedence over the unintended low confidence prediction, however this was not always the case during volunteer testing. Care should be taken to avoid false negatives because in the context of a sign language detection system, unintended information could be interpreted and conveyed.

Because of the use of a single camera, only a 2D perspective of the gesture can be interpreted. If a finger is hidden behind another due to the y-axis rotation of the hand, then the model is not able to identify the gesture correctly. **Figure 4.14** shows some examples of this.


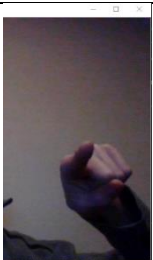
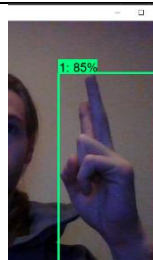
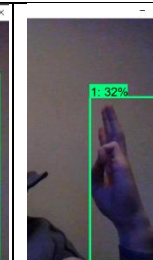
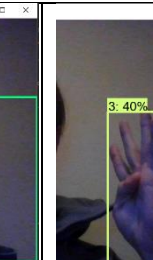
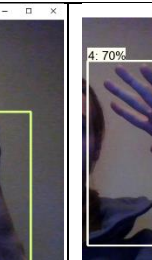
| | | | | | | |
|------------------|---|---|---|--|---|---|
| |  |  |  |  |  |  |
| Predicted | 0 | 0 | 1 - 85% | 1 - 32% | 3 - 40% | 4 - 70% |
| Actual | 1 | 1 | 2 | 3 | 4 | 5 |

Figure 4.14 – Fingers hidden due to orientation

The x-axis rotation of the hand also impacted accuracy; the flatter the gesture is to the webcam the more accurate the prediction seemed to be.

All models struggled to detect gestures when rotated. For example, gestures turned 90 degrees or upside down often get classified incorrectly. **Figure 4.15** shows some examples of this.


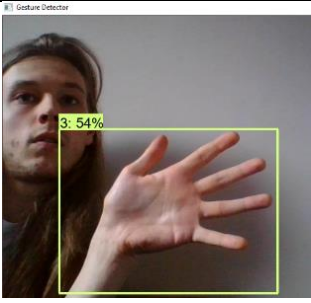
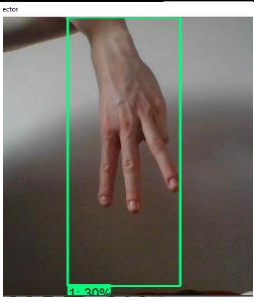
| | | | |
|------------------|---|--|---|
| |  |  |  |
| Predicted | 5 – 94% | 3 – 54% | 1 – 30% |
| Actual | 5 (No Rotation) | 5 (90 Degree Rotation) | 3 (180 Degree Rotation) |

Figure 4.15 – Hand rotation confusing the model

Applying a random rotation augmentation to the training data or using a keypoints approach to gesture detection could improve accuracy in these scenarios.

4.7.4 Summary

Given medium to high light conditions and a contrasting background, the final application can detect gestures with reasonably high accuracy. Refer to the demonstration video at 3:00 for examples of the SSD_Brightness model running as expected or **appendices 8.16-8.25** for screenshots.

5 Conclusion

This project provided an opportunity for an in-depth investigation into the rapidly developing world of machine learning and artificial intelligence. The application delivered as part of this project proves that the goals laid out in the specification were realistic and achievable.

Faster RCNN is the most accurate model but the least efficient when recognising gestures. SSD is more efficient but struggled in low lighting conditions. SSD_Brightness works better in low light environments while maintaining the speed of SSD. Given more time, these models could be improved with more varied training data, additional data augmentation or improved model architectures.

Pre-processing steps did not improve accuracy significantly apart from in poor lighting conditions in the final version of the application.

While object detection algorithms can be used for gesture recognition, a keypoints or image segmentation approach would likely work better, as it is more specific to the anatomy on the hand and allows for more control options.

In conclusion, this project was a success as a working gesture detector was developed using a neural network capable of detecting finger counting gestures 1 through 5. The main drawback of the application is the speed at which predictions are given due to frames being processed on the CPU rather than a GPU. The application works on affordable hardware with just a webcam and demonstrates what is possible in the field of computer vision.

6 Critical Reflection

6.1 Relating to application

The application can detect gestures 1,2,3,4 and 5 in environments with good lighting conditions however there are some ways in which the application could be improved.

Participant 2 suggested allowing the toggling of pre-processing steps while inference is running to make switching between them more seamless.

Participant 1 suggested allowing the user to change the size of the webcam window would make it easier to see predictions being given.

These UI improvements were not implemented due to them being relatively minor and it made more sense to focus on improving model performance.

A significant impact on model accuracy is the quality of the dataset used. Training the models on a diverse dataset with a greater variety of environments, hand sizes and colour would improve generalisability. Collecting this data however would require a large time investment which was not feasible for this project.

Exploring additional data augmentation options such as introducing noise or random rotations could improve performance in the real world. This could also provide a deeper understanding of the augmentations available when training future models. This was not explored due to time limitations regarding re-training and testing the models.

If the models were trained to work on smaller images, this could improve inference speed however the most significant limiting factor of speed is the hardware the application is running on.

The application could be improved by enabling it to detect multiple gestures in a frame, however this resulted in the same gesture being detected twice when only one hand was being displayed. Finding a solution to this would be pursued if more time were available.

Experimenting with more model architectures could have resulted in improved accuracy however it would require a large amount of time to test each architecture. Future innovation in object detection model architectures might mean that future models are more efficient and can be run on more affordable hardware giving similar or better results. This could be another avenue to explore in future.

6.2 Relating to the specification

I successfully investigated how to detect gestures through a video input device using computer vision and a neural network (objective 1).

As a result of delivering the application, my knowledge of machine learning and artificial intelligence has significantly advanced (objective 2). Appropriate languages and frameworks were identified and applied successfully (objective 3).

The dataset was not collected personally but was collated by Sergio Davies which saved a significant amount of time, allowing me to focus on other aspects of the project (objective 4).

An application capable of recognising gestures was delivered but inference was too computationally demanding, so feedback could not be given in a reasonable time as reported in testing (objective 5).

Several models were compared using evaluation metrics and confusion matrices as well as using real world testing and volunteer feedback to ensure the models performed as expected (objective 6).

Feature extraction algorithms were experimented with but did not improve accuracy significantly. More time could have been spent experimenting with these techniques however improvements would be negligible unless the models were re-trained on processed images (objective 7).

Initially it was declared that the application should be able to recognise multiple gesture variants of a represented number however this was not formally tested. From personal testing of the application, sometimes the model could correctly predict this however results were not consistent. Diversifying the training set should help address this problem.

6.3 Future Improvements

The application was only trained to recognise finger counting but the techniques explored in the report should translate to other static hand gestures. It would be interesting to see how the models would perform on a more complex dataset such as ASL. Training the model on ASL would increase the usefulness of the system but would decrease accuracy due to a larger number of possible classifications.

It would be interesting to investigate how dynamic gestures could be interpreted by a machine learning system. For example, being able to “swipe to scroll” or “point to click” on touch screen devices.

The training data consisted of a hand superimposed over a background which meant that the lighting was not realistic. This meant when predicting in the real world, the models sometimes struggled to detect gestures accurately. One way to address this would be to collect training images in environments where the application is likely to be used. This could be in an office or with a person sat at a desk. This, accompanied by greater variations in lighting, should help the model adapt to poor lighting conditions. Collecting and annotating this quantity of data is not feasible in this case due to the project’s time constraints.

Testing showed that it was difficult for the model to differentiate between the hand and the background where lighting was not optimal, or the background was similar to skin colour. Use of a depth sensor could improve this, however it would require an appropriately labelled dataset and more computing power during training and inference.

An image segmentation approach could help improve accuracy and generalisation of the application. This approach would require additional datasets which were unavailable and considerably more computational power during training and inference.

A keypoints approach should increase inference speed while increasing accuracy when parts of the hand are hidden. This would require additional labelled datasets and significant development time that was not available for this project.

Additional pre-processing techniques could help extract relevant information from the frame if more time were available. This could help increase the accuracy and reliability of the system.

It would be interesting to see if it is possible to implement a feature that allows the models to improve over time, either by registering new user gestures or collecting more data relating to existing gestures. This would allow users to correct the model when it is wrong and feedback that data to a server where the model is continually trained and re-distributed to improve performance in a wider range of environments. This kind of system would require a large amount of planning and development time as well as human monitoring to ensure no bad data corrupts the training set.

6.4 Ethical Concerns

Problems could arise if a user has a darker skin colour as the application was only trained and tested on participants with lighter skin colours. In 2019 Joy Buolamwini exposed biases in facial recognition systems created by several large tech companies. These systems were not trained on a diverse enough dataset resulting in error rates of around 31.37% for darker females (Joy Buolamwini, 2019) and in one case misidentifying a college student as a terrorist. The facial recognition systems were withheld from further development or deployment until federal regulations could be passed. This case highlights the importance in training models with a diverse dataset.

The issue of skin colour could be addressed by training the models with a greater variety of skin colours and participants. Using monochrome gloves in the training examples and in practice might help, however this would likely cause discomfort and is not the best for usability as extra materials would be required to use the application.

Because the application was not trained or tested on female hands, problems also may arise when trying to detect them as they often differ in size and finger length compared to male hands. Children's hands may also be hard to recognise as they are smaller than adult hands. This was not tested in this report due to sensitivity surrounding child participants.

This project does not gather any sensitive information about users and all data captured through the webcam is disposed of immediately, which complies with the General Data Protection Regulation (GDPR).

6.5 Personal and Professional development

Over the course of this project, I have completed independent research and development at a level not previously attempted by myself. This has improved my writing, critical evaluation and research skills as well as increasing my confidence when fulfilling future software requirements. I have gathered knowledge and experience surrounding what makes machine learning systems perform well and I am excited to learn more about AI and machine learning.

Reading research papers gave me a greater appreciation of how innovation occurs incrementally. As improvements are made to existing technologies, it is very exciting to see how far machine learning and computer vision has come in recent years. I will certainly be keeping up with developments relating to these fields and hopefully using these ideas to develop solutions that help improve people's quality of life.

Significant time was spent developing the first two prototypes. If starting again with the knowledge I have now, I would allocate more time to research and development. Even though the prototyping approach worked well in this case, I now understand the value of performing targeted research before starting development, especially in a field like machine learning.

7 Bibliography

- Ahmed, R. (2019, Marcy 5). *A Take on H.O.G Feature Descriptor*. Retrieved from Analytics Vidhya: <https://medium.com/analytics-vidhya/a-take-on-h-o-g-feature-descriptor-e839ebba1e52>
- Algorithma. (2018, April 30). *Introduction to Loss Functions*. Retrieved from Algorithma: <https://algorithmia.com/blog/introduction-to-loss-functions#:~:text=What's%20a%20Loss%20Function%3F,ll%20output%20a%20lower%20number.>
- Andreas Savakis, R. S. (2014). *Efficient Eye Detection using HOG-PCA Descriptor*. San Francisco, California, United States: Proc. SPIE 9027, Imaging and Multimedia Analytics in a Web and Mobile World.
- AquaViews. (2018). Scuba Hand Signals to Remember. Retrieved from <https://www.leisurepro.com/blog/scuba-guides/scuba-hand-signals-you-need-to-remember/>
- Arrows, K. (2020, March 29). *Fix: Your CPU Supports Instructions that this TensorFlow Binary was not Compiled to use AVX2*. Retrieved from APPUALS: <https://appuals.com/fix-your-cpu-supports-instructions-that-this-tensorflow-binary-was-not-compiled-to-use-avx2/>
- Aston Zhang, Z. C. (2020). *Dive into Deep Learning*. Retrieved from <https://d2l.ai>
- Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools.
- Brownlee, J. (2016, March 23). *Gradient Descent For Machine Learning*. Retrieved from Machine Learning Mastery: <https://machinelearningmastery.com/gradient-descent-for-machine-learning/>
- Bulao, J. (2021, 03 18). *How Much Data Is Created Every Day in 2020?* Retrieved from Tech Jury: <https://techjury.net/blog/how-much-data-is-created-every-day/#gref>
- Buscema, D. M. (1998). *Back Propagation Neural Networks*. PubMed. doi:10.3109/10826089809115863
- Chollet, F. (2017). *Deep Learning with Python*. Shelter Island, New York: Manning Publications.
- Christian Zimmermann, T. B. (2017). *Learning to Estimate 3D Hand Pose from Single RGB Images*. Freiburg: IEEE International Conference on Computer Vision (ICCV). Retrieved from <https://arxiv.org/abs/1705.01389>
- Copeland, B. (2020, Aug 11). Artificial intelligence. *Encyclopedia Britannica*. Retrieved from <https://www.britannica.com/technology/artificial-intelligence>
- Davies S., L. S.-V. (2020, Jun). Human-Robot_Finger-Counting_Dataset. Retrieved from https://github.com/EPSRC-NUMBERS/Human-Robot_Finger-Counting_Dataset
- Davies S., L. S.-V. (2021, June 6). A Database for Learning Numbers by Visual Finger Recognition in Developmental Neuro-Robotics. *Frontiers in Neurorobotics*. Retrieved from <https://www.frontiersin.org/article/10.3389/fnbot.2021.619504>

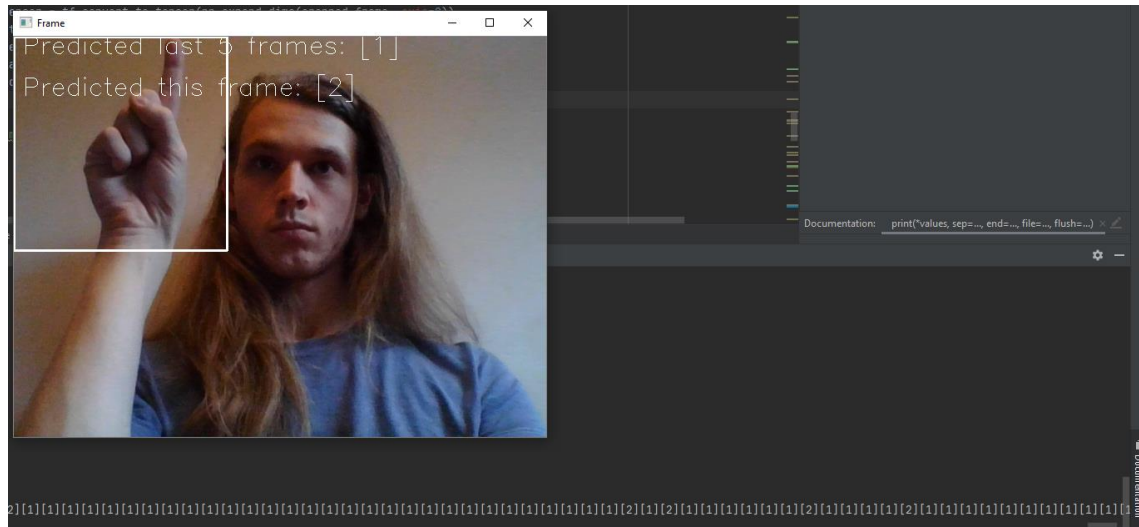
- Diederik P. Kingma, J. L. (2015). *ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION*. ICLR. Retrieved from <https://arxiv.org/pdf/1412.6980>
- Fernando De la Torre, M. J. (2001). *Robust Principal Component Analysis for Computer Vision*. Vancouver, BC: IEEE. doi:10.1109/ICCV.2001.937541
- Fuzhen Zhuang, Z. Q. (2019). *A Comprehensive Survey on Transfer Learning*. CoRR: <http://arxiv.org/abs/1911.02685>.
- Gandhi, R. (2018, 07 9). *R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms*. Retrieved from Towards Data Science: <https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e>
- Gondhalekar, A. (2020, Mar 24). *Data Augmentation — Is it really necessary?* Retrieved from Analytics Vidhya: <https://medium.com/analytics-vidhya/data-augmentation-is-it-really-necessary-b3cb12ab3c3f#:~:text=It%20helps%20us%20to%20increase,Augmentation%20helps%20reduce%20over%2Dfitting>.
- Hamilton Project. (2015, Feb 19). *One Dollars Worth of Computer Power 1980-2010*. Retrieved from The Hamilton Project: https://www.hamiltonproject.org/charts/one_dollars_worth_of_computer_power_1980_2010
- Happiness Ugochi Dike, Y. Z. (2018). *Unsupervised Learning Based On Artificial Neural*. Shenzhen: 2018 IEEE International Conference on Cyborg and Bionic Systems (CBS), 322-327. doi:10.1109/CBS.2018.8612259
- HaptX Inc. (2021, Jan 26). *HaptX Launches HaptX Gloves DK2 to Bring True-contact Haptics to VR and Robotics*. Retrieved from PR Newswire: <https://www.prnewswire.com/news-releases/haptx-launches-haptx-gloves-dk2-to-bring-true-contact-haptics-to-vr-and-robotics-301215041.html>
- He, X. Z. (2015). *Deep Residual Learning for Image Recognition*. Las Vegas: IEEE: 770-778. 10.1109/CVPR.2016.90. .
- Huang J, R. V. (2017). *Speed/accuracy trade-offs for modern convolutional object detectors*. CoRR: <http://arxiv.org/abs/1611.10012>.
- Jolliffe, I. (1986). *Principal Component Analysis*. Springer.
- Joseph Redmon, e. a. (2015). *You Only Look Once: Unified, Real-Time Object Detection*. Washington: Computer Vision Foundation: <http://arxiv.org/abs/1506.02640>.
- Joy Buolamwini, I. D. (2019). *Actionable Auditing: Investigating the Impact of Publicly Naming Biased Performance Results of Commercial AI Products*. Conference on Artificial Intelligence, Ethics, and Society. Retrieved from <https://www.media.mit.edu/publications/actionable-auditing-investigating-the-impact-of-publicly-naming-biased-performance-results-of-commercial-ai-products/>
- Kapkar, N. (2015). TensorFlow with Colab Tutorial. Retrieved from https://github.com/Nkap23/TensorFlow_with_Colab_tutorial

- Karl Weiss, T. M. (2016). *A survey of transfer learning*. Journal of Big Data: <https://doi.org/10.1186/s40537-016-0043-6>.
- Kolungade, S. (2020, Aug 3). *Object Detection, Image Classification and Semantic Segmentation using AWS Sagemaker*. Retrieved from Medium: <https://medium.com/@kolungade.s/object-detection-image-classification-and-semantic-segmentation-using-aws-sagemaker-e1f768c8f57d>
- Kumar, S. (2020, Sep 23). *TensorFlow 2 Custom Object Detection*. Retrieved from https://github.com/Sujith93/Tensorflow2_Custom_objectionDetection/tree/master
- Leroy D'Souza, I. P. (2011). *Kinect to Architecture*. Auckland, New Zealand: IVCNZ. Retrieved from https://www.researchgate.net/publication/266221903_Kinect_to_Architecture
- LIU, Y. H. (2018). *Feature Extraction and Image Recognition with Convolutional Neural Networks*. Glasgow College, University of Electronic Science and Technology of China: IOP Publishing. Retrieved from <https://iopscience.iop.org/article/10.1088/1742-6596/1087/6/062032/pdf>
- Mahapattanakul, P. (2019, Nov 10). *From Human Vision to Computer Vision — Convolutional Neural Network(Part3/4)*. Retrieved from Becoming Human.ai: <https://becominghuman.ai/from-human-vision-to-computer-vision-convolutional-neural-network-part3-4-24b55ffa7045>
- Maneela Jain, P. S. (2013). *Review of Image Classification Methods and Techniques*. Bhopal: International Journal of Engineering Research & Technology (IJERT): 2278-0181. Retrieved from <https://www.ijert.org/research/review-of-image-classification-methods-and-techniques-IJERTV2IS80157.pdf>
- Media Pipe. (2020). MediaPipe Hands. Retrieved from <https://google.github.io/mediapipe/solutions/hands.html>
- Mohammad Sadegh Ebrahimi, H. K. (2018). *Study of Residual Networks for Image Recognition*. Stanford, CA: CoRR: <http://arxiv.org/abs/1805.00325>.
- Ng, A. (2019, 05). DeepLearning.ai. *Convolutional Neural Networks*. Coursera.
- NIDCD, N. I. (2019, March). *American Sign Language*. Retrieved from NIDCD: <https://www.nidcd.nih.gov/health/american-sign-language>
- PAI, A. (2020, Feb 17). *CNN vs. RNN vs. ANN – Analyzing 3 Types of Neural Networks in Deep Learning*. Retrieved from Analytics Vidhya: <https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/>
- Patrikar, S. (2019, Oct 1). *Batch, Mini Batch & Stochastic Gradient Descent*. Retrieved from Towards Data Science: <https://towardsdatascience.com/batch-mini-batch-stochastic-gradient-descent-7a62ecba642a>
- Powell, V. (2015). *Image Kernels*. Retrieved from Setosa: <https://setosa.io/ev/image-kernels/>
- Rini Akmeliawati, M. P.-L. (2007). *Real-Time Malaysian Sign Language Translation using Colour Segmentation and Neural Network*. Warsaw Poland: IEEE.

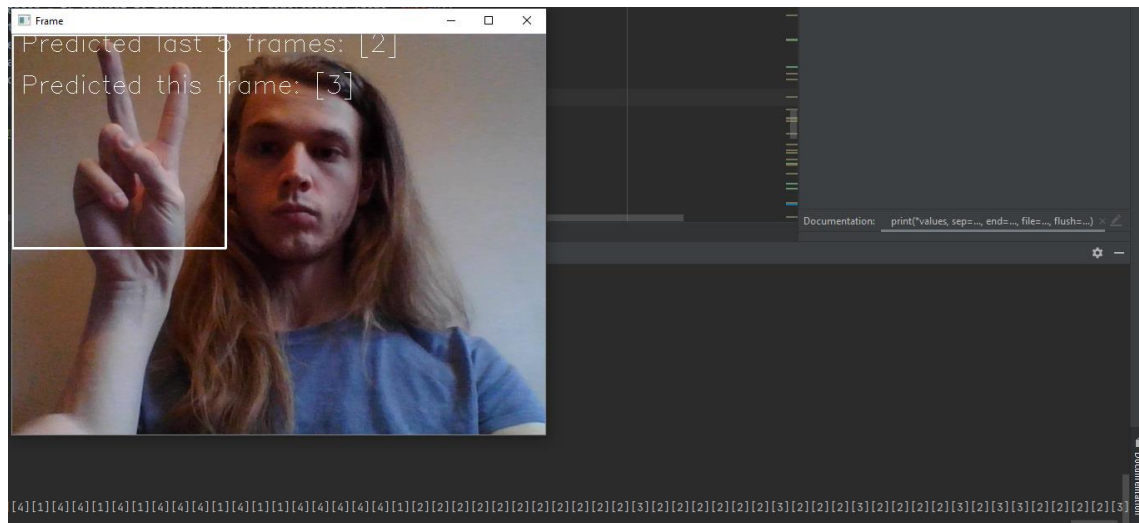
- Rodrigo Vershae, J. R.-d.-S. (2015). *Object Detection: Current and Future Directions*. Frontiers in Robotics and AI. 2. 10.3389/frobt.2015.00029. .
- Rosenblatt, F. (1958). *THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANISATION IN THE BRAIN*. Cornell: Psychological Review: <https://doi.org/10.1037/h0042519>.
- Ross B. Girshick, J. D. (2013). *Rich feature hierarchies for accurate object detection and semantic*. Berkeley: CoRR: <http://arxiv.org/abs/1311.2524>.
- S. Albawi, T. A.-Z. (2017). *Understanding of a Convolutional Neural Network*. Antalya, Turkey: International Conference on Engineering and Technology (ICET). Retrieved from 10.1109/ICEngTechnol.2017.8308186
- SignAll. (2017). Retrieved from SignAll: <https://www.signall.us/>
- Triggs, N. D. (2005). *Histograms of oriented gradients for human detection*. San Diego, CA, USA, 2005, pp. 886-893 vol. 1, doi: 10.1109/CVPR.2005.177.: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05).
- Tsung-Yi Lin, M. M. (2014). *Microsoft {COCO:} Common Objects in Context*. CoRR. Retrieved from <http://arxiv.org/abs/1405.0312>
- Wei Liu, D. A.-Y. (2015). *SSD: Single Shot MultiBox Detector*. Cornell: CoRR: <http://arxiv.org/abs/1512.02325>.
- Wilderness Arena. (2015, June 1). *How to use hand and arm signals to communicate silently*. Retrieved from Wilderness Arena: <https://www.wildernessarena.com/environment/signaling/standard-military-hand-arm-signals-visual-signaling>
- Yongxin Zhou, J. B. (2007). *Atlas-Based Fuzzy Connectedness Segmentation and Intensity Nonuniformity Correction Applied to Brain MRI*. IEEE.
- Zhihao Jia, M. Z. (2018). *Beyond Data and Model Parallelism for Deep Neural Networks*. CoRR. Retrieved from <https://dblp.org/rec/journals/corr/abs-1807-05358.bib>
- Zhong-Qiu Zhao, S.-t. X. (2018). *Object Detection with Deep Learning: A Review*. CoRR <http://arxiv.org/abs/1807.05511>.

8 Appendix

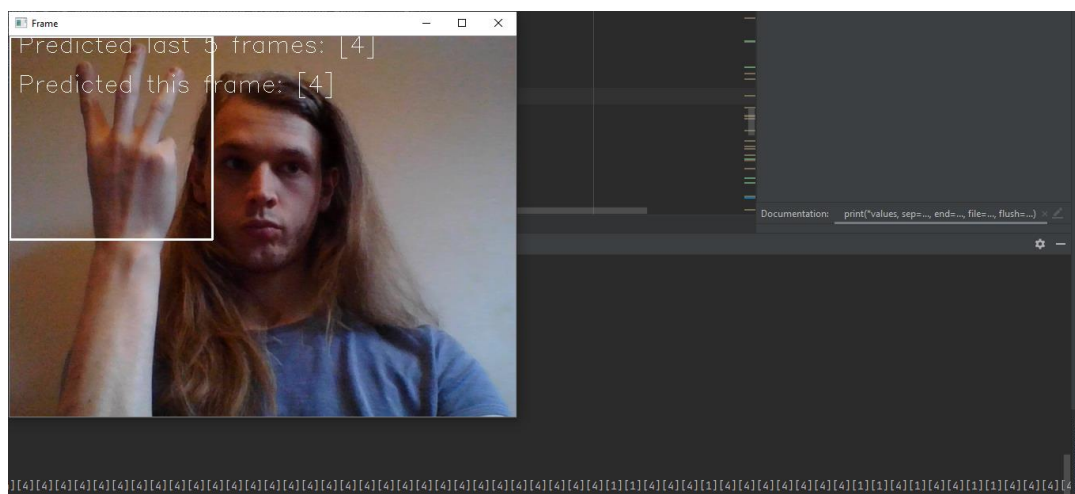
8.1 Prototype 1 – Image Classification (Gesture 1)



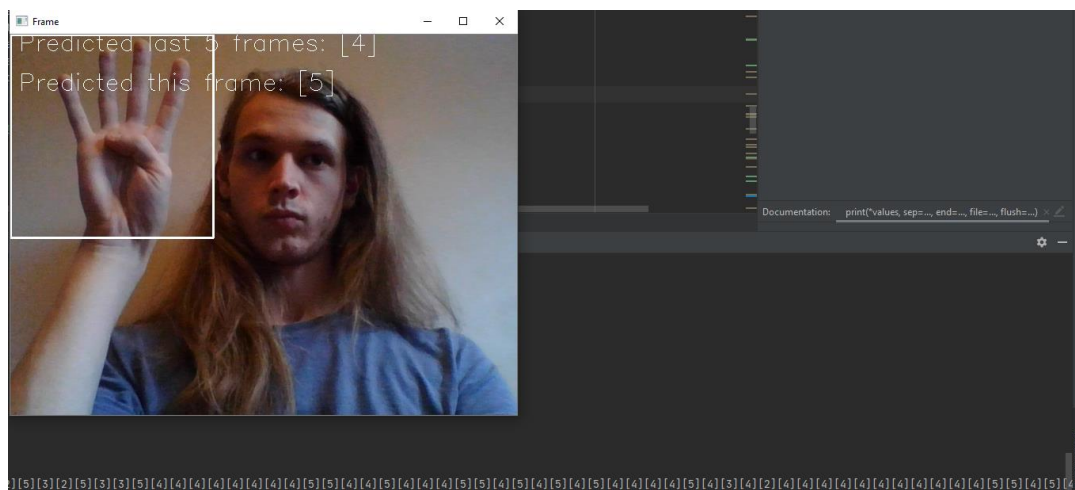
8.2 Prototype 1 – Image Classification (Gesture 2)



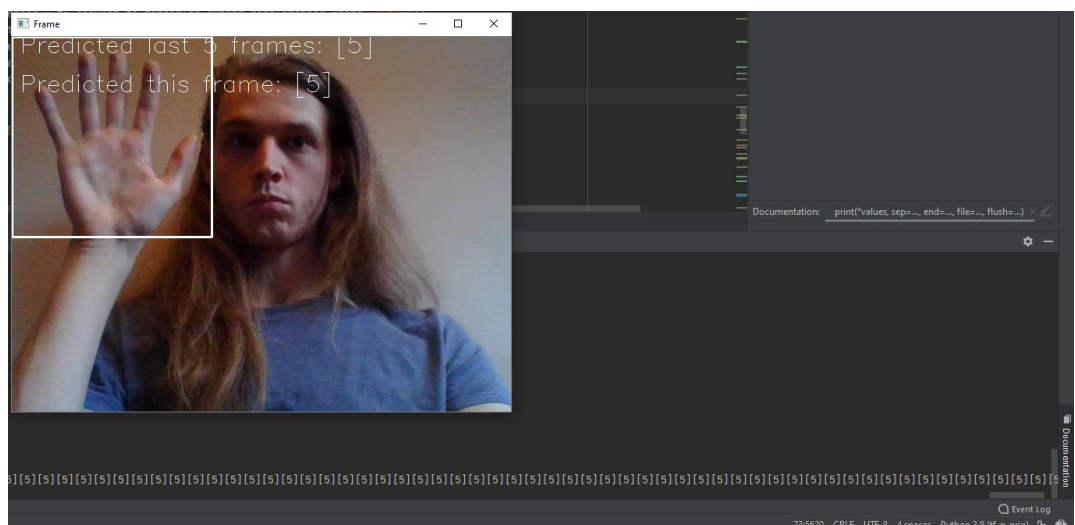
8.3 Prototype 1 – Image Classification (Gesture 3)



8.4 Prototype 1 – Image Classification (Gesture 4)



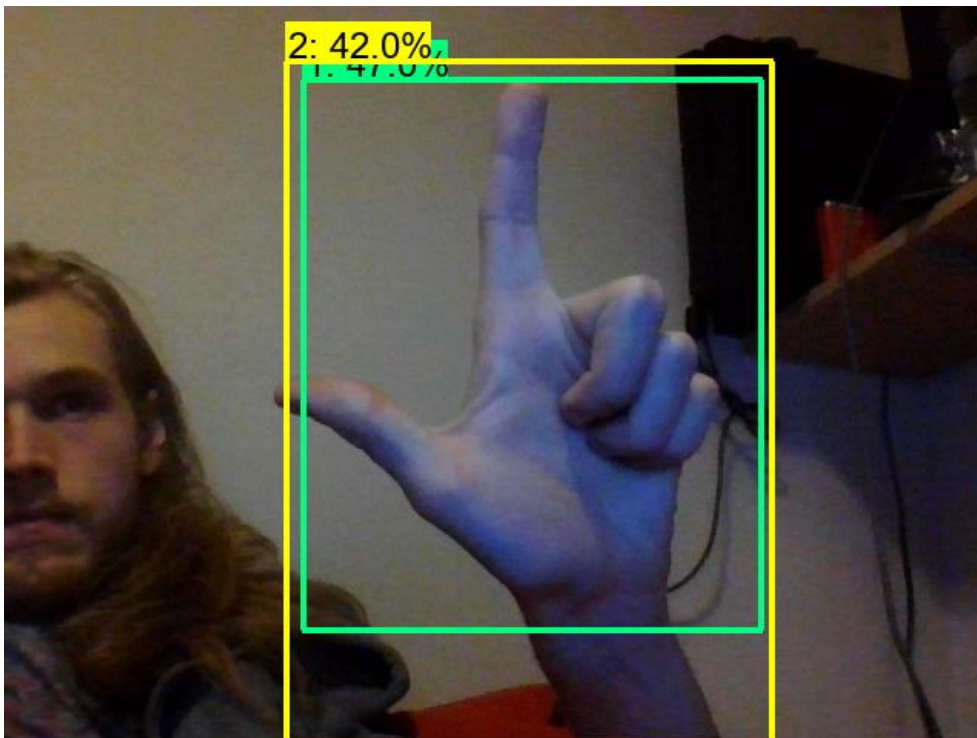
8.5 Prototype 1 – Image Classification (Gesture 5)



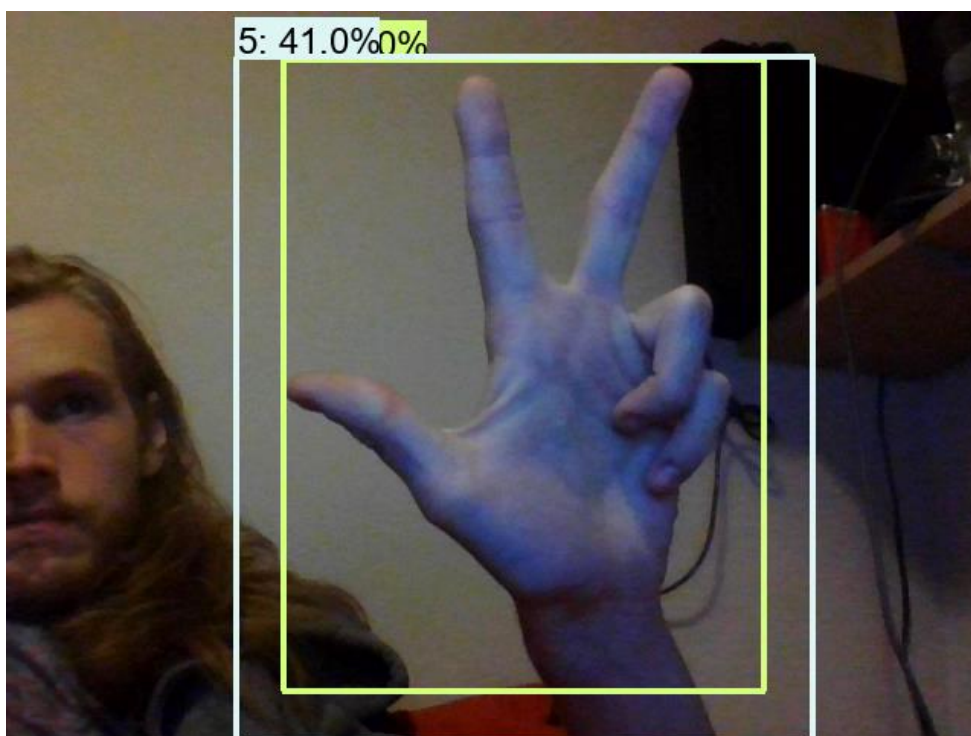
8.6 Prototype 2 – Object Detection from Scratch (Gesture 1)



8.7 Prototype 2 – Object Detection from Scratch (Gesture 2)



8.8 Prototype 2 – Object Detection from Scratch (Gesture 3)



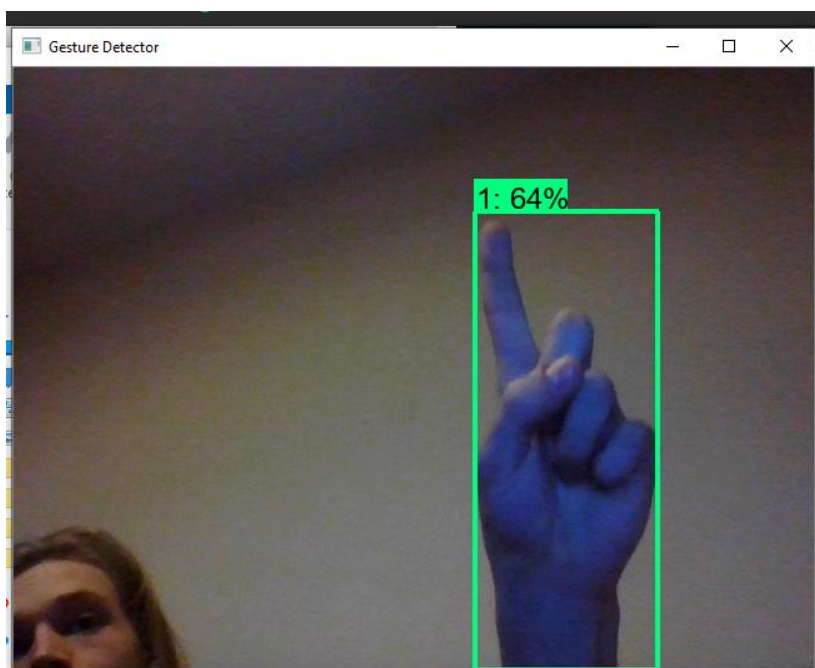
8.9 Prototype 2 – Object Detection from Scratch (Gesture 4)



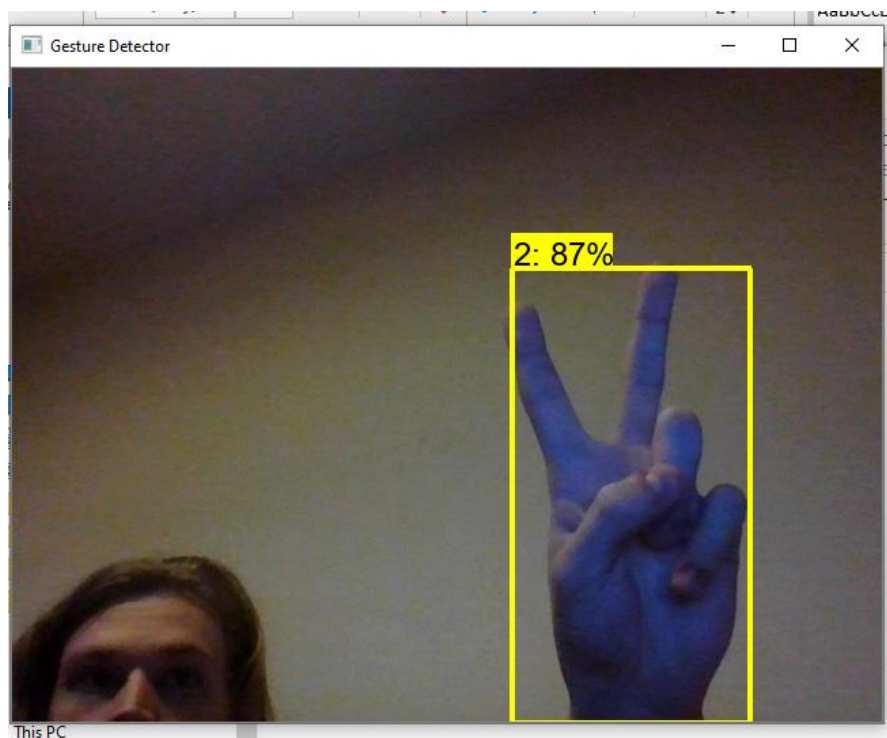
8.10 Prototype 2 – Object Detection from Scratch (Gesture 5)



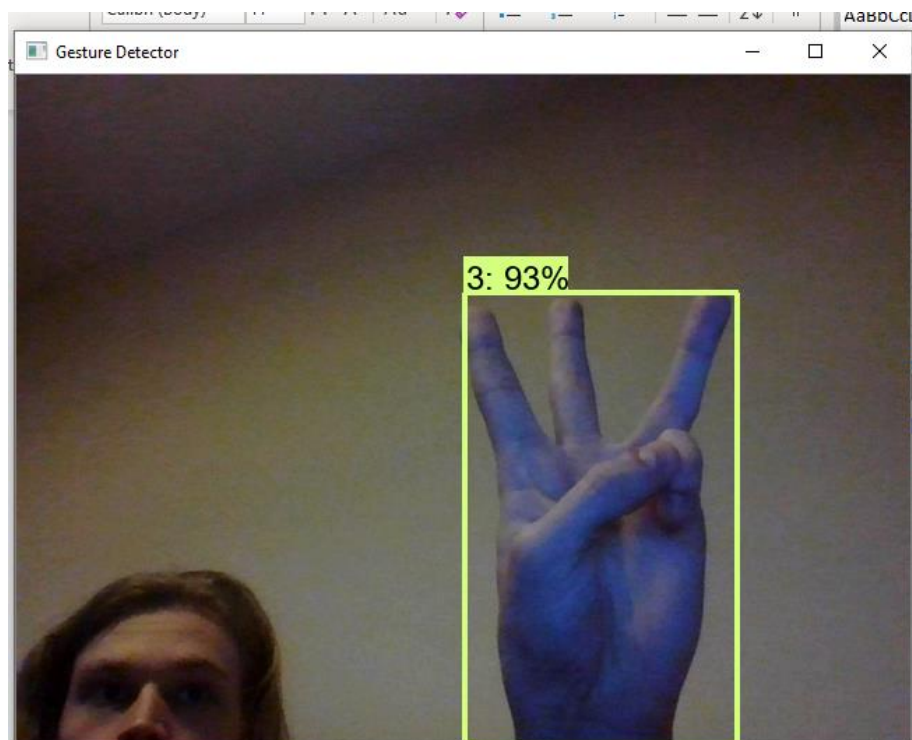
8.11 Prototype 3 - BGR Input (Gesture 1)



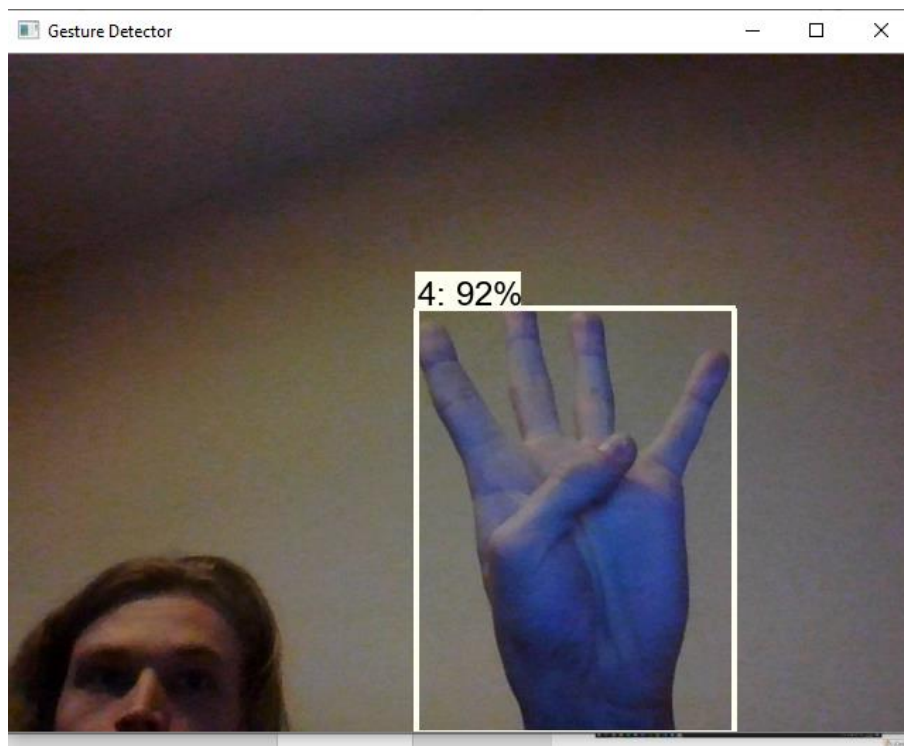
8.12 Prototype 3 - BGR Input (Gesture 2)



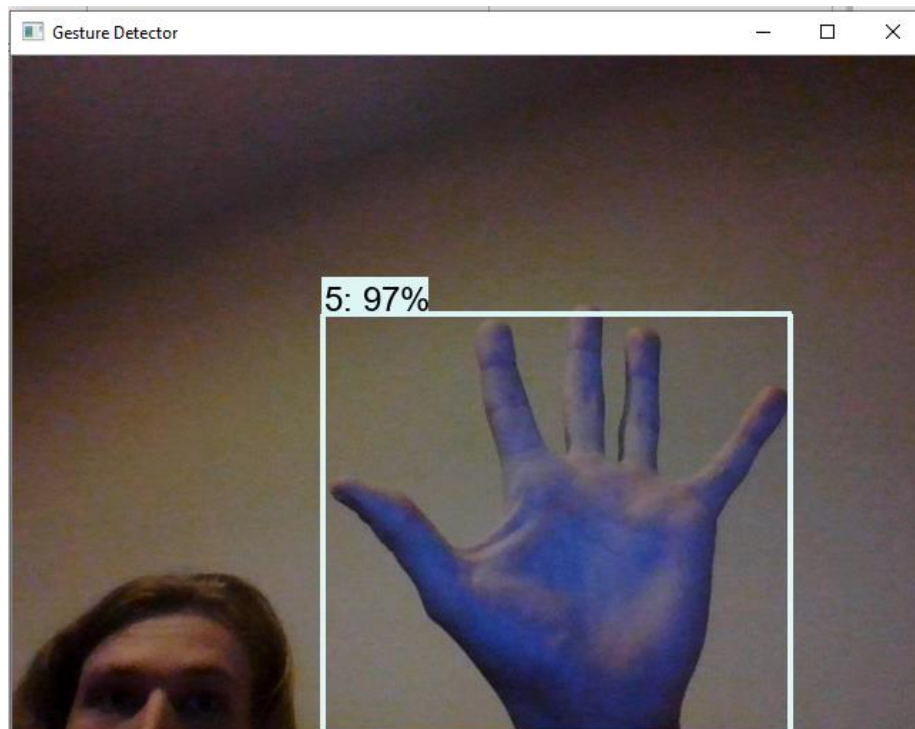
8.13 Prototype 3 - BGR Input (Gesture 3)



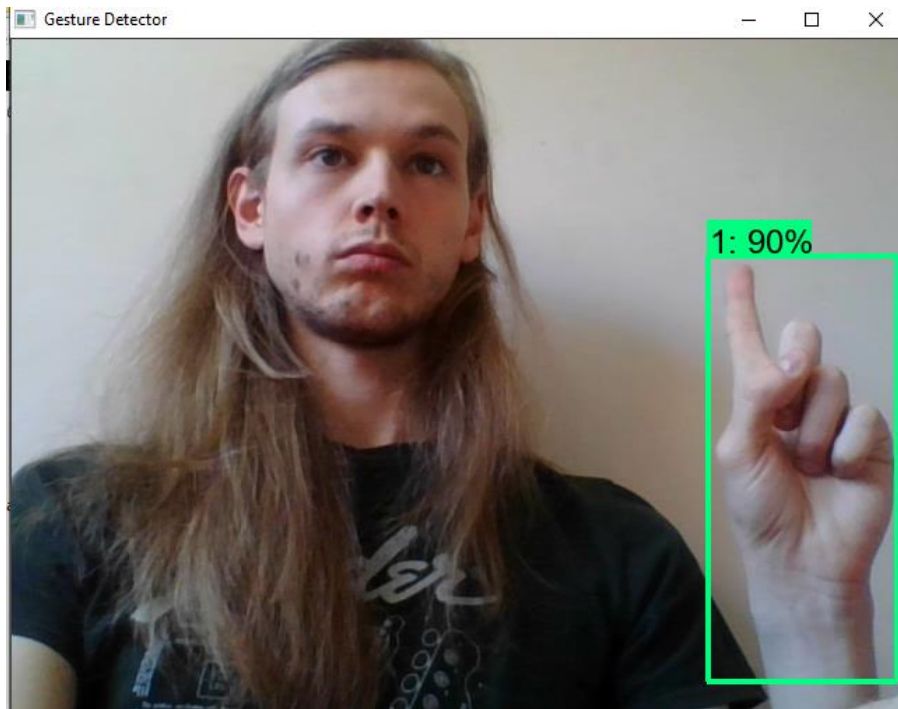
8.14 Prototype 3 - BGR Input (Gesture 4)



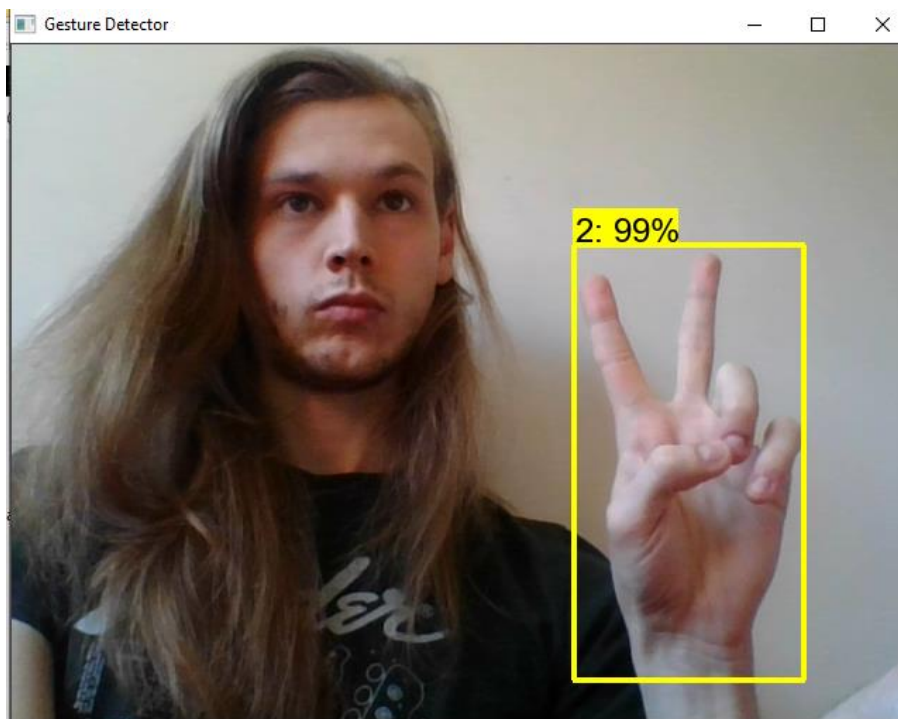
8.15 Prototype 3 - BGR Input (Gesture 5)



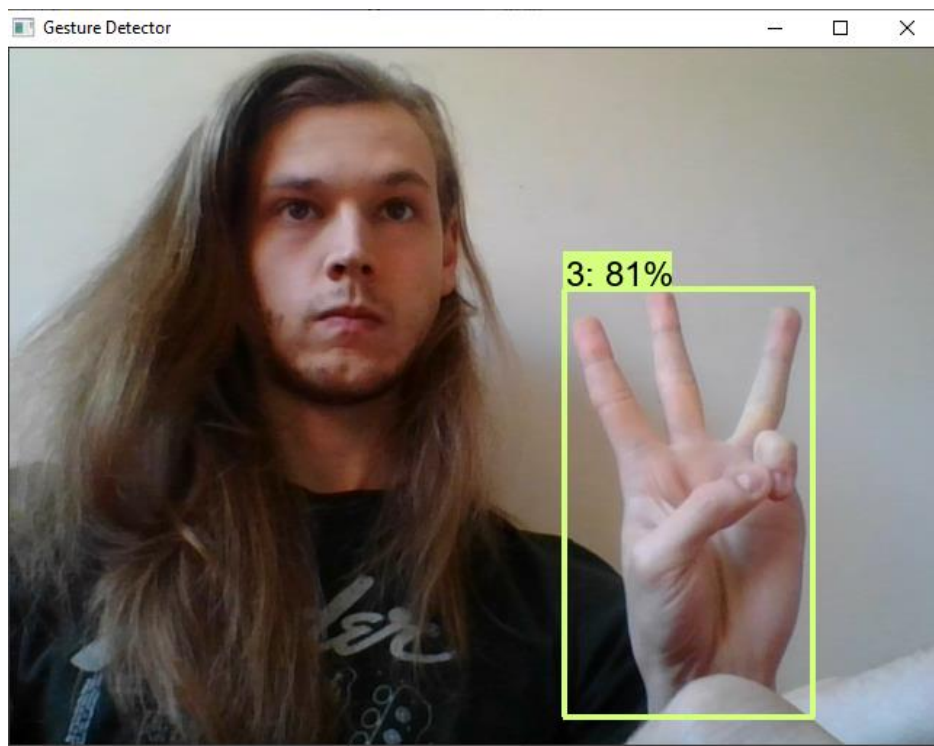
8.16 Final application - RGB - Faster RCNN (Gesture 1)



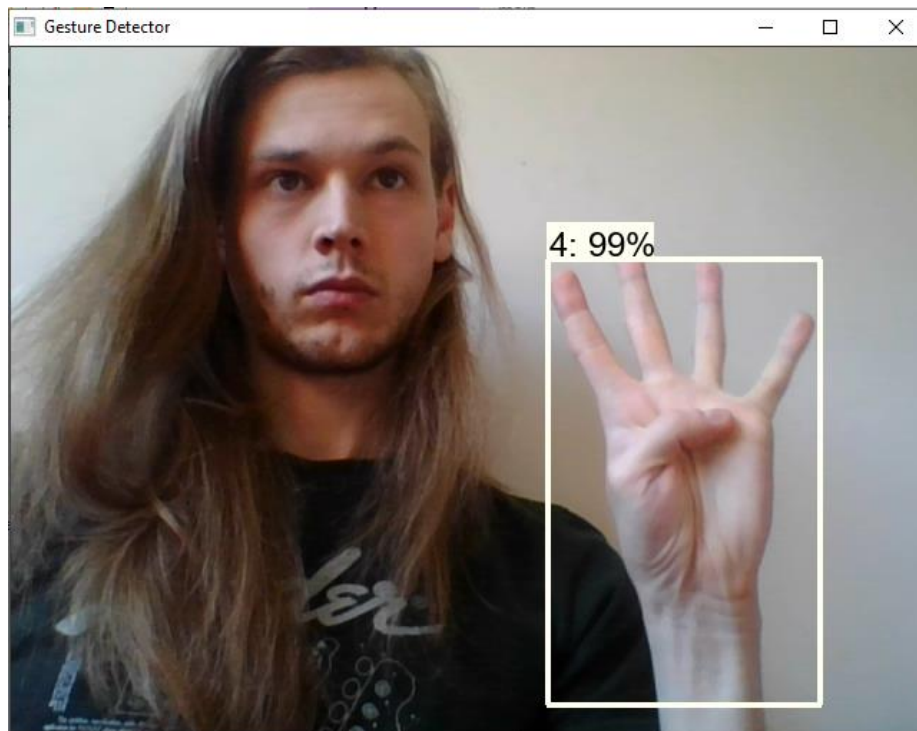
8.17 Final application – RGB - Faster RCNN (Gesture 2)



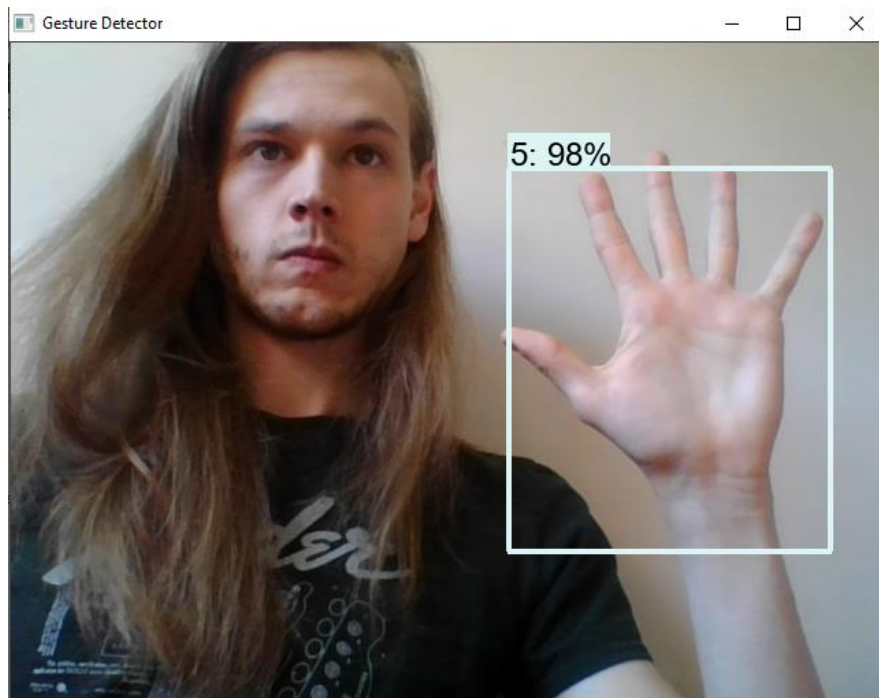
8.18 Final application - RGB - Faster RCNN (Gesture 3)



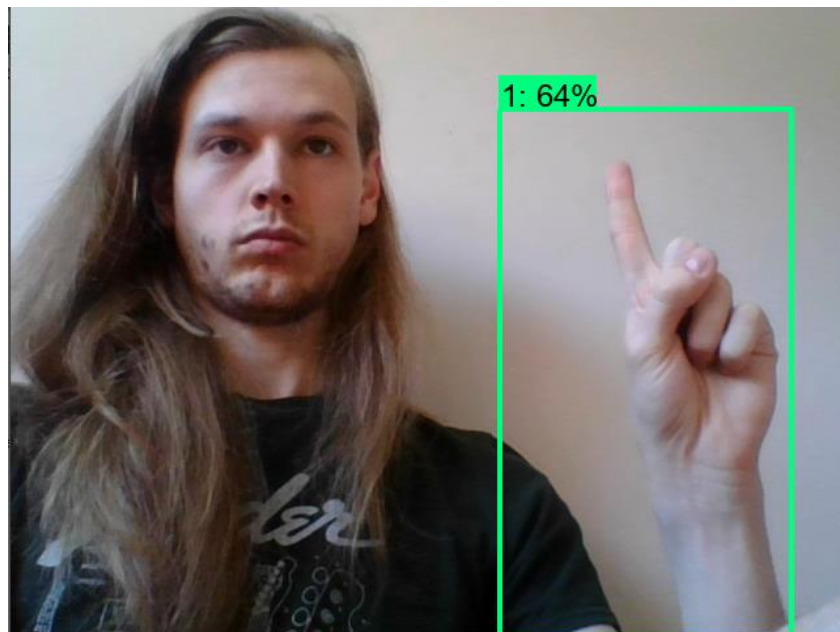
8.19 Final application - RGB - Faster RCNN (Gesture 4)



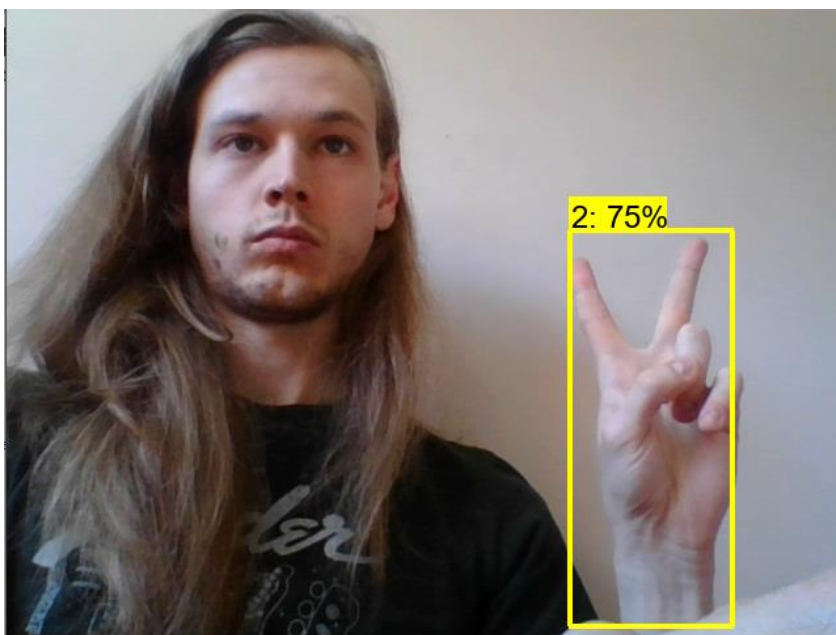
8.20 Final application - RGB - Faster RCNN (Gesture 5)



8.21 Final application - RGB – SSD Brightness (Gesture 1)



8.22 Final application - RGB – SSD Brightness (Gesture 2)



8.23 Final application - RGB – SSD Brightness (Gesture 3)



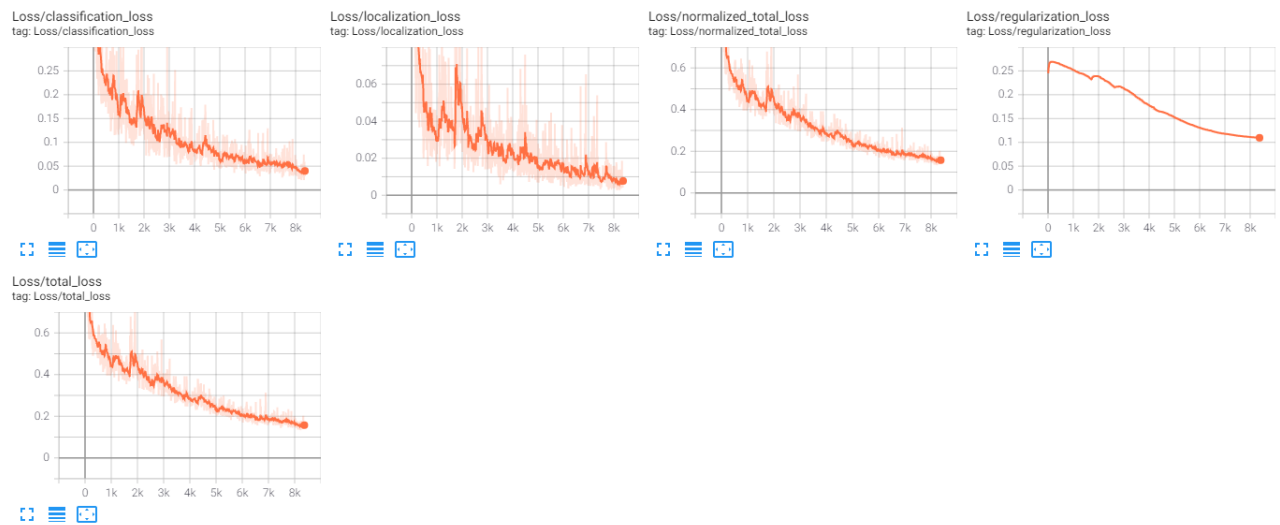
8.24 Final application - RGB – SSD Brightness (Gesture 4)



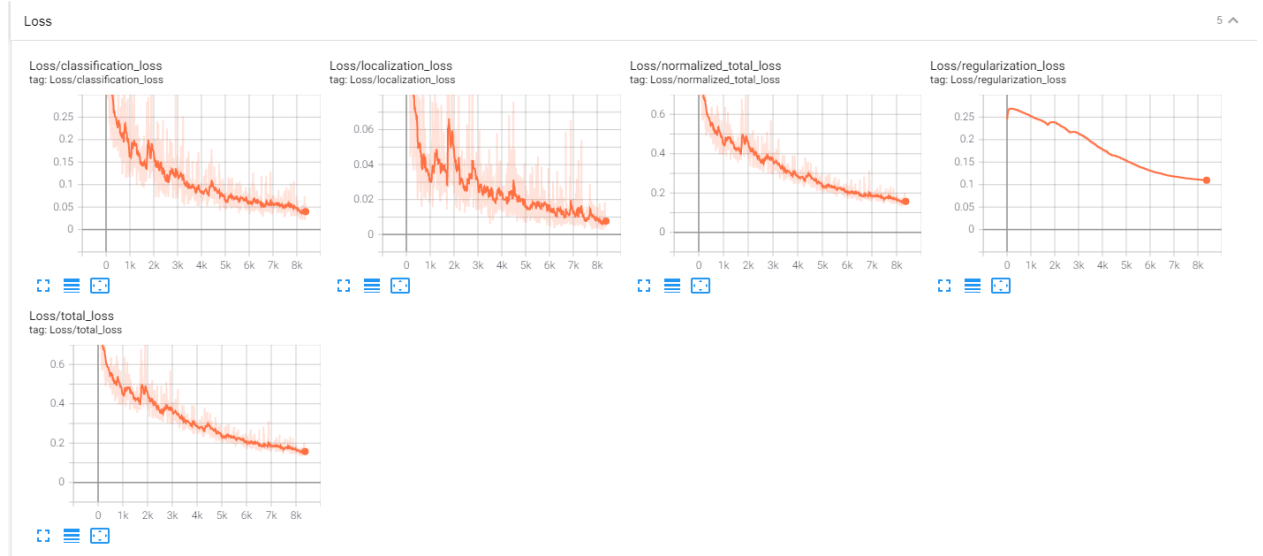
8.25 Final application - RGB – SSD Brightness (Gesture 5)



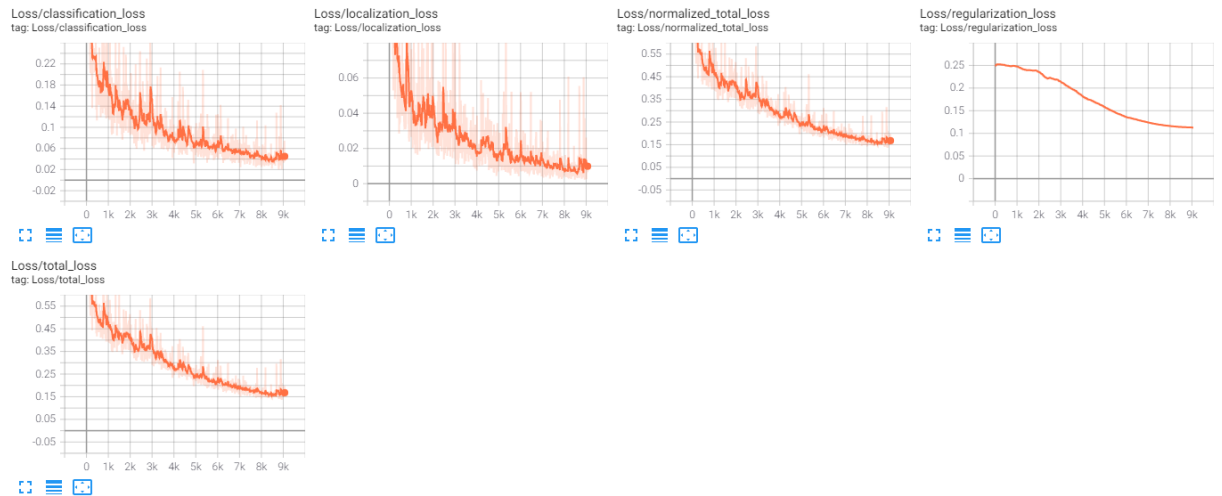
8.26 SSD TensorBoard



8.27 Faster RCNN TensorBoard



8.28 SSD Brightness TensorBoard



8.29 Participant 1 - Round 1

Gesture Detector Questionnaire

For each of the models in the application, test gestures 1,2,3,4,5 and report on the accuracy and responsiveness of each. Examples of the gesture can be found on the final page of this document.

Start the application choose, choose a model, and hold up a gesture.

For each model and gesture give a score out of 10 representing the accuracy of the prediction and bounding box as well as reporting on overall responsiveness.

Pre-processing steps are there to assist in extracting the gesture from the image. Also report on these methods compared to the default settings.

| Faster_RCNN | Default | | Background Subtraction | | Skin Detection | |
|--------------------|------------|--------------|------------------------|--------------|-------------------|--------------|
| | Confidence | Bounding Box | Confidence | Bounding Box | Confidence | Bounding Box |
| One | - | - | - | - | - | - |
| Two | - | - | - | - | - | - |
| Three | - | - | - | - | - | - |
| Four | - | - | - | - | Detected 5 at 79% | 8 |
| Five | 9 (93%)* | 10 | - | - | 79% | 9 |
| Overall | 2 | | 0 | | 4 | |

*Reverse hand

| SSD | Default | | Background Subtraction | | Skin Detection | |
|------------|------------|--------------|------------------------|--------------|----------------|--------------|
| | Confidence | Bounding Box | Confidence | Bounding Box | Confidence | Bounding Box |
| One | - | - | | | | |
| Two | - | - | | | | |
| Three | - | - | | | | |
| Four | - | - | | | | |
| Five | - | - | | | | |
| Overall | 0 | | 0 | | 0 | |

| SSD_Brightness | Default | | Background Subtraction | | Skin Detection | |
|-----------------------|------------------|------------------------|------------------------|--------------|----------------|-----------------------|
| | Confidence | Bounding Box | Confidence | Bounding Box | Confidence | Bounding Box |
| One | 33% | 2 | 94% | 10 | - | - |
| Two | 76% | 10 | 94% | 10 | - | 1 |
| Three | ~32%, detected 2 | 9 | 77% | 9 | - | - |
| Four | 93% | 8 (chopped fingertips) | 98% | 10+ | 61% | 6 (no thumb included) |
| Five | 83% | 8 (chopped fingertips) | 65% (7 for accuracy) | 10 | - (detected 4) | 5 |
| Overall | 8 | | 10 | | 4 | |

(This will be anonymised)

Participant name: 1

Date: 16/03/2021

Any additional comments:

This application is CPU intensive – the average peaked around 87% on AMD 3600 3.6GHz CPU. Perhaps worth doing some further optimisation, and if that is not possible, having the option to limit the resource usage (at the cost of performance) would be handy.

Few bits around the environment and my setup: the webcam that I own has a wide-angle lens and a tendency to over-expose the image. For these reasons, some of the tests had to be in the dark (so

that the screen glare would highlight the skin against the dark background) and some required awkward positions to get out of the shot and only have my hand visible.

Each test was done generously – starting off with five fingers going down to one. This was done on both sides of the right hand. Variables such as distance from the camera, orientation and contrast were all fine-tuned to give each gesture as good of a chance as possible.

There were cases with the background subtraction pre-processor where the model just would not pick up the gesture under seemingly perfect conditions (high contrast, large spaces between fingers).

The same tests were then also done on a laptop with a built in 720p webcam. Right off the bat, the first thing I noticed was the performance decline – RCNN model took between 1.72 and 2.25s to infer each frame. Setting the performance plan to favour better performance did improve things slightly, but not enough to make it nice to use.

The results were very much similar to the above – RCNN did not produce any reliable results, with SSD following closely behind (no pre-processors were used). SSD with brightness model, however, shone bright, with each gesture being recognised with high confidence and the bounding box encapsulating the whole of the hand as expected. This went a step further with the background subtraction pre-processor.

So, just to summarise the findings, SSD_Brightness combined with background subtraction produces the best results across the two device types that I have tested. The model also has the best *time to infer* performance too, therefore it is safe to say that the model is best out the three.

As for the further improvements, having some controls to allow the user to manually adjust the contrast and exposure of the camera. Further training could be done in order to allow the models to differentiate differently sized fingers and different skin tones too. Adding the ability for the computer to recognise combinations of fingers rather than the defined gestures would also be interesting, though I am unsure of the amount of complexity this would entail.

Very interesting project, I very much enjoyed testing it.

8.30 Participant 2 – Round 1

Gesture Detector Questionnaire

For each of the models in the application, test gestures 1,2,3,4,5 and report on the accuracy and responsiveness of each. Examples of the gesture can be found on the final page of this document.

Start the application choose, choose a model, and hold up a gesture.

For each model and gesture give a score out of 10 representing the accuracy of the prediction and bounding box as well as reporting on overall responsiveness.

Pre-processing steps are there to assist in extracting the gesture from the image. Also report on these methods compared to the default settings.

(This will be anonymised)

Participant name: 2

Date: 18th March 2020

| Faster_RCNN | Default | | Background Subtraction | | Skin Detection | |
|-------------|------------|--------------|------------------------|--------------|----------------|--------------|
| | Confidence | Bounding Box | Confidence | Bounding Box | Confidence | Bounding Box |
| One | 0 | 0 | 0 | 0 | 0 | 0 |
| Two | 0 | 0 | 0 | 0 | 0 | 0 |
| Three | 0 | 0 | 0 | 0 | 0 | 0 |
| Four | 0 | 0 | 0 | 0 | 0 | 0 |
| Five | 0 | 0 | 0 | 0 | 0 | 0 |
| Overall | 0 | 0 | 0 | 0 | 0 | 0 |

| SSD | Default | | Background Subtraction | | Skin Detection | |
|---------|------------|--------------|------------------------|--------------|----------------|--------------|
| | Confidence | Bounding Box | Confidence | Bounding Box | Confidence | Bounding Box |
| One | 0 | 0 | 0 | 0 | 0 | 0 |
| Two | 0 | 0 | 0 | 0 | 0 | 0 |
| Three | 0 | 0 | 0 | 0 | 0 | 0 |
| Four | 0 | 0 | 0 | 0 | 0 | 0 |
| Five | 0 | 0 | 0 | 0 | 0 | 0 |
| Overall | 0 | 0 | 0 | 0 | 0 | 0 |

| SSD_Brightness | Default | | Background Subtraction | | Skin Detection | |
|----------------|------------|--------------|------------------------|--------------|----------------|--------------|
| | Confidence | Bounding Box | Confidence | Bounding Box | Confidence | Bounding Box |
| One | 0 | 0 | 7 | 7 | 0 | 0 |
| Two | 0 | 0 | 5 | 7 | 0 | 0 |
| Three | 0 | 0 | 7 | 7 | 2 | 2 |
| Four | 4 | 6 | 9 | 9 | 8 | 8 |
| Five | 0 | 0 | 6 | 7 | 5 | 5 |
| Overall | 2 | 2 | 7 | 7 | 6 | 6 |

Any additional comments

- *First two models didn't seem to detect gestures at all so had no confidence that the application was working until the third model.*
- Third model seems to detect several gestures after an average of 1.2 seconds, however had to stretch the gap between fingers for the number 3 (as close to the image provided) suspect that the application needs a larger and more varied data set of hand sizes, finger shapes and finger gaps replicating the gesture to identify them more accurately with more confidence.
- I was using a low-quality camera which might be affecting the applications performance.

- The Application stops responding when loading a model. This may be resolved by queuing the model loading while keeping the UI on a separate thread.
- It would be easier to use the application if you could toggle the pre-processing options during inference.
- Application seemed to struggle in daytime lighting conditions.
- Application struggles in a crowded environment
- Application was lagging considerably so maybe look improving it's performance. Consider using a server to do the heavy processing.(cloud based architecture)

8.31 Participant 1 –Round 2

Participant name: 1

Date: 24/03/2021

| Faster_RCNN | Default | | Background Subtraction | | Skin Detection | |
|-------------|------------|--------------|------------------------|--------------|----------------|--------------|
| | Confidence | Bounding Box | Confidence | Bounding Box | Confidence | Bounding Box |
| One | 88% | 8 | | | | |
| Two | 99% | 10 | | | | |
| Three | 95% | 10 | | | | |
| Four | 94% | 10 | | | | |
| Five | 99% | 10 | | | | |
| Overall | 10 | | | | | |

| SSD | Default | | Background Subtraction | | Skin Detection | |
|---------|------------|--------------|------------------------|--------------|----------------|--------------|
| | Confidence | Bounding Box | Confidence | Bounding Box | Confidence | Bounding Box |
| One | - | - | | | | |
| Two | 38% | 6 | | | | |
| Three | 38% | 7 | | | | |
| Four | 78% | 8 | | | | |
| Five | 81% | 6 | | | | |
| Overall | 6 | | | | | |

| SSD_Brightness | Default | | Background Subtraction | | Skin Detection | |
|----------------|------------|--------------|------------------------|--------------|----------------|--------------|
| | Confidence | Bounding Box | Confidence | Bounding Box | Confidence | Bounding Box |
| One | 57% | 9 | 90%+ | 10 | | |
| Two | 92% | 10 | 90%+ | 10 | | |
| Three | 91% | 8 | 90%+ | 10 | | |
| Four | 94% | 9 | 90%+ | 10 | | |
| Five | 95% | 9 | 90%+ | 10 | | |
| Overall | 9 | | 10 | | | |

Any additional comments:

This time round I was really impressed by how accurate and fast the RCNN model was compared to its previous iteration. Comparing the base model results with the test run from 16/03/2021, the current iteration has performed extremely well with a 10 instead of a two, with all gestures recognised. This improvement did come at a cost – the inference time has increased by around 150ms to approximately 860ms per frame. That said, the model recognised each gesture nearly instantly in less-than-ideal lighting and background conditions.

I did not only test the gestures documented below – I also had a go at variations such as index and third finger for two, index, middle and pinky for 3 and so on, all of which worked exactly as expected. Wonderful.

SSD model did not fill me with a lot of confidence, but compared to the previous set of results, it still did miles better. Nearly all of the gestures were recognised to a fairly high confidence, but the bounding box did tend to bounce around a little. By bounce around I mean it would increase and reduce in size with every slight movement. This may have possibly been caused by colours in the background that somewhat resembled skin. In addition, if I came into the shot, the model would sometimes recognise my face as a 1...

SSD brightness with and without background subtraction remains as the most usable of the three. In terms of confidence and accuracy, RCNN is just a little better. But when it comes to inference times, SSD Brightness shines with each frame taking under 400ms to process (this is at least half the speed it takes to do the same with RCNN). I also tried this model with the background subtraction pre-processor just to see if the performance has been affected. It has, but in a good way. Each gesture was now being recognised even faster than the previous set of results. And to top that, the confidence and bounding box were really accurate.

All in all, this iteration has been a pleasure to test. The UI improvements with the addition of camera setting adjustments has come in handy.

SSD Brightness with and without background subtraction remains the most effective and usable of the 3.

8.32 Participant 2 – Round 2

Participant name: 2

Date: 25/03/2021

| Faster_RCNN | Default | | Background Subtraction | | Skin Detection | |
|--------------------|------------|--------------|------------------------|--------------|----------------|--------------|
| | Confidence | Bounding Box | Confidence | Bounding Box | Confidence | Bounding Box |
| One | 8 | 8 | 0 | 0 | 0 | 0 |
| Two | 7 | 7 | 0 | 0 | 0 | 0 |
| Three | 4 | 7 | 0 | 0 | 0 | 0 |
| Four | 9 | 9 | 0 | 0 | 0 | 0 |
| Five | 7 | 7 | 0 | 0 | 0 | 0 |
| Overall | 8 | 8 | 0 | 0 | 0 | 0 |

| SSD | Default | | Background Subtraction | | Skin Detection | |
|------------|------------|--------------|------------------------|--------------|----------------|--------------|
| | Confidence | Bounding Box | Confidence | Bounding Box | Confidence | Bounding Box |
| One | 6 | 4 | 0 | 0 | 0 | 0 |
| Two | 7 | 7 | 0 | 0 | 0 | 0 |
| Three | 7 | 6 | 0 | 0 | 0 | 0 |
| Four | 6 | 3 | 0 | 0 | 2 | 2 |
| Five | 4 | 4 | 0 | 0 | 0 | 0 |
| Overall | 6 | 6 | 0 | 0 | 0 | 0 |

| SSD_Brightness | Default | | Background Subtraction | | Skin Detection | |
|-----------------------|------------|--------------|------------------------|--------------|----------------|--------------|
| | Confidence | Bounding Box | Confidence | Bounding Box | Confidence | Bounding Box |
| One | 9 | 8 | 6 | 9 | 4 | 8 |
| Two | 6 | 8 | 6 | 9 | 7 | 8 |
| Three | 7 | 9 | 5 | 9 | 8 | 8 |
| Four | 9 | 9 | 8 | 9 | 8 | 8 |
| Five | 0 | 8 | 9 | 9 | 6 | 7 |
| Overall | 8 | 8 | 7 | 7 | 7 | 7 |

Any additional comments

- SSD (Default). Bounding box seem to be larger then RCNN. Kept identifying head and shoulders as one finger
- SSD Brightness (Default). Bounding box seem to be larger then RCNN. Kept identifying head and shoulders as one finger
- SSD Brightness (Background subtraction). Tight bounding box but lost confidence every second inference between gestures 1-3
- SSD Brightness (Default). Couldn't identify 5 finger gesture but performed well on the rest
- SSD (Skin Detection). Bounding box seemed to only identify head and shoulders as one finger and ignore hand except for the four finger gesture
- Pre processing buttons were disabled while running inference
- There's no stop inference which is annoying
- Button labels should toggle enable/disable when selected
- Used left hand this time to test the data being used included gestures from both left and right hands.
- No indication why sliders are there. Shouldn't the application detect the dominant colour balance? (or whatever their utility is supposed to be)

Previous Unresolved Comments

- Application was lagging considerably so maybe look improving it's performance. Consider using a server to do the heavy processing.(cloud based architecture)
- I was using a low-quality camera which might be affecting the applications performance.

8.33 Project Specification

PROJECT SPECIFICATION - Project (Technical Computing) 2020/21

| | |
|--------------------------|---|
| Student: | Jack Thickett |
| Date: | 24/09/2020 |
| Supervisor: | Sergio Davies |
| Degree Course: | Computer Science |
| Title of Project: | Classification of hand gestures using a neural network and computer vision |

Elaboration

Hand gestures are used widely even before computer vision. They are easily understood by humans however getting a computer to recognise gestures is much harder. Advances in deep learning are improving computers ability to detect human like communication. The main aim of this project is to train a neural network on a set of gestures so that it can recognise those gestures given a video stream.

This trained network then could be used to translate sign language into English or be the basis of gesture-controlled software such as a game, robotics or smart devices in the home such as a stereo system or lights. Gesture recognition could also be useful in situations where surface contamination might be a concern. For example in a medical environment or fast food restaurant where a particular device might be used by many different people, gestures could be used to interface with a computer so that users don't spread or catch contaminants by having to touch a touch screen or keyboards and mouse.

There are many different approaches to gesture recognition, part of my research will be finding out what techniques/algorithms work best for this application. I will begin with building a convolutional neural network for classification then evaluating and iterating from there to improve modal accuracy. I will consider using established pre-trained networks for this task and try to develop them for the problem area.

Project Aims

Objectives

- Investigate how to detect a gesture through video input using computer vision and neural networks.
- Develop a knowledge and understanding of the field of machine learning and artificial intelligence.
- Research the most appropriate languages and frameworks for this application.
- Collect sufficient data to train and develop a neural network powered program.
- Develop an application that can detect gestures from a video input device and give feedback in a reasonable time.
- Evaluate the performance of various architectures and select the most suitable approach.
- Experiment with feature extraction algorithms to improve the accuracy of the application.

Project deliverable(s)

The deliverable for this project will be a Python desktop application that uses the TensorFlow framework for neural network and OpenCV to capture frames from the users' webcam. Users will be able to show gestures that have been trained into the neural network to their webcam and the neural network will predict what gesture it is.

Action plan

| Task | Details | Submission Date |
|----------------------|--|--------------------------|
| Find a Supervisor | Discuss feasibility with project supervisor | 9 th October |
| Preliminary research | Undertake research to confirm that the project is feasible. Download all software needed. (Python/TensorFlow/OpenCV/PyCharm) | 15 th October |

| | | |
|--|--|---------------------------|
| | Research into neural networks and computer vision – Pick an approach that my research shows work the best. | |
| Project Specification and Ethics form | Flesh out specification to clearly describe the aims and objectives | 23 rd October |
| Information Review | Pool research and write information review | 20 th November |
| Develop Prototype | Build a small version of the application to get to grips with the frameworks I will be using and write any useful functions. | 20 th December |
| Develop Application | Build and test deliverable to meet specification. Test the application to a satisfactory level of accuracy. | 30 th January |
| User testing | Allow people to test application. Provide participant information sheet and | 5 th February |
| Contents Page | | 15 th February |
| Draft Evaluation/ Draft Report | | 19 th March |
| Finish and Submit report and deliverable | | 15 th April |
| Demo of work | | 29 th April |

BCS Code of Conduct

I confirm that I have successfully completed the BCS code of conduct on-line test with a mark of 70% or above. This is a condition of completing the Project (Technical Computing) module.

Signature: J.Thickett

Publication of Work

I confirm that I understand the "Guidance on Publication Procedures" as described on the Bb site for the module.

Signature: J.Thickett

GDPR

I confirm that I will use the "Participant Information Sheet" as a basis for any survey, questionnaire or participant testing materials. This form is available on the Bb site for the module and as an appendix in the handbook.

Signature: J.Thickett

8.34 Ethics Form

UREC2 RESEARCH ETHICS PROFORMA FOR STUDENTS UNDERTAKING LOW RISK PROJECTS WITH HUMAN PARTICIPANTS

This form is designed to help students and their supervisors to complete an ethical scrutiny of proposed research. The University [Research Ethics Policy](#) should be consulted before completing the form. The initial questions are there to check that completion of the UREC 2 is appropriate for this study. The final responsibility for ensuring that ethical research practices are followed rests with the supervisor for student research.

Note that students and staff are responsible for making suitable arrangements to ensure compliance with the General Data Protection Act (GDPR). This involves informing participants about the legal basis for the research, including a link to the University research data privacy statement and providing details of who to complain to if participants have issues about how their data was handled or how they were treated (full details in module handbooks). In addition the act requires data to be kept securely and the identity of participants to be anonymized. They are also responsible for following SHU guidelines about data encryption and research data management. Information on the [Ethics Website](#)

The form also enables the University and College to keep a record confirming that research conducted has been subjected to ethical scrutiny.

The form may be completed by the student and the supervisor and/or module leader (as applicable). In all cases, it should be counter-signed by the supervisor and/or module leader, and kept as a record showing that ethical scrutiny has occurred. Some courses may require additional scrutiny. Students should retain a copy for inclusion in their research projects, and a copy should be uploaded to the relevant module Blackboard site.

Please note that it may be necessary to conduct a health and safety risk assessment for the proposed research. Further information can be obtained from the College Health and Safety Service.

Checklist Questions to ensure that this is the correct form

1. Health Related Research with the NHS or Her Majesty's Prison and Probation Service (HMPPS) or with participants unable to provide informed consent

| Question | Yes/No |
|--|--------|
| 1. Does the research involve? | No |
| • Patients recruited because of their past or present use of the NHS | |
| • Relatives/carers of patients recruited because of their past or present use of the NHS | No |
| • Access to data, organs or other bodily material of past or present NHS patients | No |
| • Foetal material and IVF involving NHS patients | No |
| • The recently dead in NHS premises | No |
| • Prisoners or others within the criminal justice system recruited for | No |

| | |
|---|----|
| health-related research* | |
| <ul style="list-style-type: none"> Police, court officials, prisoners or others within the criminal justice system* | No |
| <ul style="list-style-type: none"> Participants who are unable to provide informed consent due to their incapacity even if the project is not health related | No |
| 2. Is this a research project as opposed to service evaluation or audit? <i>For NHS definitions of research etc. please see the following website</i> http://www.hra.nhs.uk/documents/2013/09/defining-research.pdf | No |

If you have answered **YES** to questions **1 & 2** then you **MUST** seek the appropriate external approvals from the NHS, Her Majesty's Prison and Probation Service (HMPPS) under their independent Research Governance schemes. Further information is provided below.

<https://www.myresearchproject.org.uk>

NB College Teaching Programme Research Ethics Committees (CTPRECS) provide Independent Scientific Review for NHS or HMPPS research and initial scrutiny for ethics applications as required for university sponsorship of the research. Applicants can use the IRAS proforma and submit this initially to their CTPREC.

1. Checks for Research with Human Participants

| Question | Yes/No |
|--|--------|
| 1. Will any of the participants be vulnerable? <i>Note: Vulnerable' people include children and young people, people with learning disabilities, people who may be limited by age or sickness, people researched because of a condition they have, etc. See full definition on ethics website</i> | No |
| 2. Are drugs, placebos or other substances (e.g. food substances, vitamins) to be administered to the study participants or will the study involve invasive, intrusive or potentially harmful procedures of any kind? | No |
| 3. Will tissue samples (including blood) be obtained from participants? | No |
| 4. Is pain or more than mild discomfort likely to result from the study? | No |
| 5. Will the study involve prolonged or repetitive testing? | No |
| 6. Is there any reasonable and foreseeable risk of physical or emotional harm to any of the participants? <i>Note: Harm may be caused by distressing or intrusive interview questions, uncomfortable procedures involving the participant, invasion of privacy, topics relating to highly personal information, topics relating to illegal activity, or topics that are anxiety provoking, etc.</i> | No |
| 7. Will anyone be taking part without giving their informed consent? | No |
| 8. Is it covert research? <i>Note: 'Covert research' refers to research that is conducted without the knowledge of participants.</i> | No |
| 9. Will the research output allow identification of any individual who has not given their express consent to be identified? | No |

If you have answered **YES** to any of these questions you are **REQUIRED** to complete and submit a UREC 3 or UREC4). Your supervisor will advise. If you have answered **NO** to all these questions then proceed with this form (UREC 2).

General Details

| | |
|--|--|
| Name of student | Jack Thickett |
| SHU email address | B8028388@my.shu.ac.uk |
| Course or qualification (student) | Computer Science |
| Name of supervisor | Sergio Davies |
| email address | Sergio.Davies@shu.ac.uk |
| Title of proposed research | Classification of hand gestures using a neural network and computer vision |
| Proposed start date | 1 st November |
| Proposed end date | 15 th April |
| Background to the study and scientific rationale for undertaking it. | Investigate computer vision techniques for image classification and train an effective neural network that can be used to classify static gestures in real-time allowing users to control computer systems with just gestures. |
| Aims & research question(s) | <p>To evaluate the performance of neural network architectures and image processing techniques in identifying hand gestures.</p> <p>To get feedback from participants on ease of use of the application as well as any ideas for features that might improve the quality of the application.</p> |
| Methods to be used for: 1.recruitment of participants, 2.data collection, 3. data analysis. | <p>Participants will be recruited friends and family to test the software and make sure it works as expected.</p> <p>No data will be stored from participants. Only live video data will be taken in by the webcam and then discarded when it is no longer in use.</p> <p>Data is analyzed by passing it through the trained neural network then output is given of what gesture is being shown.</p> |
| Outline the nature of the data held, details of anonymisation, storage and disposal procedures as required. | The application does not require any data from the participants apart from a video input which is disposed of as soon as the result is calculated. |

3. Research in Organisations

| Question | Yes/No |
|---|--------|
| 1. Will the research involve working with/within an organisation (e.g. school, business, charity, museum, government department, international agency, etc.)? | No |
| 2. If you answered YES to question 1, do you have granted access to conduct the research? <i>If YES, students please show evidence to your supervisor. PI should retain safely.</i> | N/a |
| 3. If you answered NO to question 2, is it because: A. you have not yet asked B. you have asked and not yet received an answer C. you have asked and been refused access. <i>Note: You will only be able to start the research when you have been granted access.</i> | N/a |

4. Research with Products and Artefacts

| Question | Yes/No |
|--|--------|
| 1. Will the research involve working with copyrighted documents, films, broadcasts, photographs, artworks, designs, products, programmes, databases, networks, processes, existing datasets or secure data? | Yes |
| 2. If you answered YES to question 1, are the materials you intend to use in the public domain? <i>Notes: 'In the public domain' does not mean the same thing as 'publicly accessible'.</i> <ul style="list-style-type: none"> <i>Information which is 'in the public domain' is no longer protected by copyright (i.e. copyright has either expired or been waived) and can be used without permission.</i> <i>Information which is 'publicly accessible' (e.g. TV broadcasts, websites, artworks, newspapers) is available for anyone to consult/view. It is still protected by copyright even if there is no copyright notice. In UK law, copyright protection is automatic and does not require a copyright statement, although it is always good practice to provide one. It is necessary to check the terms and conditions of use to find out exactly how the material may be reused etc.</i> <i>If you answered YES to question 1, be aware that you may need to consider other ethics codes. For example, when conducting Internet research, consult the code of the Association of Internet Researchers; for educational research, consult the Code of Ethics of the British Educational Research Association.</i> | Yes |
| 3. If you answered NO to question 2, do you have explicit permission to use these materials as data? <i>If YES, please show evidence to your supervisor.</i> | N/a |

| | |
|--|-----|
| 4. If you answered NO to question 3, is it because: A. you have not yet asked permission B. you have asked and not yet received and answer C. you have asked and been refused access. <i>Note You will only be able to start the research when you have been granted permission to use the specified material.</i> | N/a |
|--|-----|

Adherence to SHU policy and procedures

| | |
|--|------------------|
| Personal statement | |
| I can confirm that: <ul style="list-style-type: none"> I have read the Sheffield Hallam University Research Ethics Policy and Procedures I agree to abide by its principles. | |
| Student | |
| Name: Jack Thickett | Date: 04/10/2020 |
| Signature: J.thickett | |
| Supervisor or other person giving ethical sign-off | |
| I can confirm that completion of this form has not identified the need for ethical approval by the FREC or an NHS, Social Care or other external REC. The research will not commence until any approvals required under Sections 3 & 4 have been received and any necessary health and safety measures are in place. | |
| Name: Sergio Davies | Date: 26/10/2020 |
| Signature: Sergio Davies | |
| Additional Signature if required by course: | |
| Name: | Date: |
| Signature: | |

Please ensure the following are included with this form if applicable, tick box to indicate:

| | Yes | No | N/A |
|--|-------------------------------------|--------------------------|-------------------------------------|
| Research proposal if prepared previously | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Any recruitment materials (e.g. posters, letters, etc.) | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| Participant information sheet | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Participant consent form | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Details of measures to be used (e.g. questionnaires, etc.) | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

| | | | |
|--|--------------------------|--------------------------|-------------------------------------|
| Outline interview schedule / focus group schedule | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| Debriefing materials | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| Health and Safety Project Safety Plan for Procedures | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> |