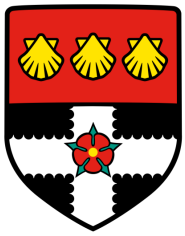

University of Reading

Department of Computer Science

Using Deep Learning and Box Speed Calibration For Predicting Rowing Boat Speed

Robert Henry Young

Supervisor: Dr Varun Ojha

A report submitted in partial fulfilment of the requirements of
the University of Reading for the degree of
Bachelor of Science in *Computer Science*

April 28, 2021

Declaration

I, Robert Henry Young, of the Department of Computer Science, University of Reading, confirm that all the sentences, figures, tables, equations, code snippets, artworks, and illustrations in this report are original and have not been taken from any other person's work, except where the works of others have been explicitly acknowledged, quoted, and referenced. I understand that if failing to do so will be considered a case of plagiarism. Plagiarism is a form of academic misconduct and will be penalised accordingly.

I give consent to a copy of my report being shared with future students as an exemplar.

I give consent for my work to be made available more widely to members of UoR and public with interest in teaching, learning and research.

Robert Henry Young
April 28, 2021

Abstract

Having computers carry out speed predictions on objects in videos has been an interesting topic with many useful applications. This project looks at the application of gathering the metric of how fast a boat is moving in a race to present to a viewer and replace the use of expensive machines which currently do this. The method used to achieve this is to train a Convectional Neural Network using transfer learning to detect rowing boats in the frames of a race. Then use box speed calibration with assistance of a neural network to predict the speeds of the boats. The results show an R2 score of 0.93 when detecting the speeds of the boats. The videos used were also from a moving platform, meaning that the system was able to understand the movement of the environment and still make reliable predictions overcoming some long standing computer vision problems.

Keywords: Deep Leaning, Transfer Learning, Neural Networks, Computer Vision

Contents

List of Figures

List of Tables

List of Abbreviations

CNN	Convolutional Neural Network
Mask RCNN	Mask Region Based Convolutional Neural Network
Yolov4	You Only Look Once Version 4
MLP	Multi layer Perceptron
LSTM	Long Short Term Memory
ROI	Region of Interest
POV	Point of View

Chapter 1

Introduction

Speed prediction has always been a difficult issue to solve in computer vision but also one which could see many real world applications in traffic maintenance and ambient intelligent systems. One application that can be solved is that of gathering metrics in sport and in this case, finding out how fast boats are moving in a rowing race. Solving these problems requires a good understanding of the complex system which the camera is viewing to create predictions about how fast objects will be moving in the scene. Neural Networks and Deep Learning are models which could be a solution to this problem as they are known to be able to model and understand many complex systems to solve real world problems .

1.1 Background

This project is to create a system using a convolutional neural network(CNN) to take a video stream of a boat race at the Olympics, detect the boats in the race and perform a speed prediction on them to help get statistics. It's motivated by the fact that at the moment, gathering live statistics from a boat race like this requires expensive equipment, to do a task which could be done by using CNN's which would be more accessible and inexpensive in comparison. Recently there has been a lot of improvements in the field of object detection with new improved CNN models being created. At the moment there are two main popular object detection models in use, the Mask R-CNN and the YOLOv4 which both have their different merits. Mask R-CNN is generally considered to be more accurate in terms of detection but is slower than YOLOv4 as it produces a Mask as well, making YOLOv4 more practical in video stream in most cases. In terms of speed prediction there are many different ideas about how this should be done, like implementing a pipeline of tracking, alignment and measurement to achieve speed detection, but a more promising option is using a neural network. This technique records compares the pixel speeds of the boxes to the real, known speeds of the objects in question to produce a CNN model which can predict the actual speed from the speed of moving boxes.

1.2 Research Question

The main problem to be solved is that of creating speed predictions on rowing boats from video streams from a moving camera which can then be transferred to other domains or camera angles. This will require object detection to detect the boats in the video streams and then a Neural Network to solve the speed prediction problem.

1.2.1 Object Detection

Object detection is a very trivial issue now due to abundance of new corporate and scientific research that has been carried out in it,so making the best use of the many tools and techniques is important. These tools can be easily replicated and transferred to work on other problems, the only challenge to overcome is the scale of this project as it does not match those projects which have had access to data and resources . Therefor different training techniques will need to be used to create the same useful results. After this the larger problem will need to be tackled, which is to use Neural Networks to create predictions on the speed of the boats in the video streams.

1.2.2 Speed Prediction

The biggest challenge in this report is going to be the speed prediction, as this is less of a reliable technology than object detection with less work in it's area being carried out. While some work has been done on the issue, this is all very specific to the domains and cannot be directly used in this case. The speed prediction system will need to be able to work from moving camera's and be easy to transfer to different races and camera angles. These requirements therefore require a model which can learn more complex problems and for this Neural Networks are a viable option. While other projects have used Neural Networks to solve similar problems Shih et al. (2019) none have done this for isolated instances on moving camera's which is one of the requirements for the problem. There fore the main question is, can Neural Networks be used to make speed predictions on moving platforms and achieve usable results?

1.3 Aims and objectives

Aims The aims of this project are to produce a CNN model which can carry out object detection and to produce a model which can then carry out speed predictions on a video stream of a rowing boat race. This project will hopefully be able to update established techniques for speed detection like box speed calibration Rohit (2019) to be able to use Neural Networks. As well as implementing the speed prediction technique this way, it should be able to work on moving camera angles and be easily transferred to other domains, this factors would allow for the system to be used in real world use cases. As well as this it would be ideal for the boat detection to work in real time, but this may be difficult to achieve given the small scale of the project and as this is not the main research question is an ideal result, not a necessary one as it is already known that boat detectors can run accurate results at a high fps Patrona et al. (2020).

Objectives To achieve these aims, firstly a boat detection model will have to be created, which will be done by first training the model for detection. This will require data collection, pre-processing and then transfer learning on a pre-trained model. A selection of trained models will be created and from this they can all be evaluated and compared to find the best performing model. Next will be the training of the speed prediction model. This part will require implementing another Neural Network model again, however it will be trained to carry out box speed calibration. This training will require the bounding boxes to be passed from the boat detector model, matched with the bounding boxes in the previous frames and then used to

1.4 Solution approach

The approach taken to achieve speed predictions on rowing boats in this project comes in two parts, the boat detection and the speed detection. The boat detection stage detects boats in the video stream. From this it sends the bounding boxes to the speed prediction stage which carried out box speed calibration using Neural Networks.

1.4.1 Boat Detection

An approach which is becoming more and more popular in training CNN's is that of transfer learning. Using this techniques the CNN models will be loaded with their COCO weights and then fine tuned to recognise boats from a smaller data set. The models used will be Yolov4 Bochkovskiy et al. (2020) and Mask RCNN He et al. (2017). Both models will then be evaluated using mAP to work out which one is the most accurate and should be used in the final system.

1.4.2 Speed Detection

The speed detection will be using box speed calibration Rohit (2019) and will use the matched bounding boxes detected from the current and previous frames to make a prediction about how fast to boats are moving in the video. This will be attempted with two different Neural Networks: MLP and LSTM which have both been proven to work well on this type of problem (windowed time series) Gers et al. (2002). Both these models will then be evaluated using R2 Score which is a good metric to use when assessing regression models Van Herwaarden et al. (2014), the best model will then be used in the final system which detects boats and makes predictions on their speeds.

1.5 Summary of contributions and achievements

In terms of contributions, the most ground breaking part of this report is the use of Neural Networks with box speed calibration to achieve speed detection. The use of Neural Networks with this technique allows for the model to have a greater understanding of what is going on in the scene as Neural Networks are effective at solving complex problems like the one in this case which means that speed predictions from a variety of different and moving POV's is potentially possible with the right training. This has not been possible before with previous methods which rely on an almost hard coded/ expert system understanding of a fixed POV to produce speed detection results.

1.6 Organization of the report

This report is organised into seven chapter including this chapter 1. Chapter 2 breaks down previous works in the subject area, reviewing state-of-the-art technologies in section 2.1 with discussions of projects and how the projects relate to this one in the later sections. Chapter 3 discusses the methodology used to produce the technology, discussing how technologies were incorporated and implementations used before discussing how it was all put together. Chapter 4 shows the results from the experiments in the methodologies. Chapter 5 then discusses these results and analyses what they mean and their significance. Then the Chapter 6 finalises the project and talks about what it achieved. Finally the Chapter 7 looks back on the work produced.

Chapter 2

Literature Review

With computer vision and deep learning being some of the most popular, much work has been carried out in the areas. Deep learning especially has seen a lot of recent research carried out with new models being released and improvements being released more and more. As with speed prediction projects, many different techniques have been used, providing a long list of available techniques to use in this project.

2.1 A review of state-of-the-art technologies in convolutional neural network and Speed Prediction

In terms of convolutional neural network's (CNN), there are two popular models in use: Yolov4 Bochkovskiy et al. (2020) and Mask RCNN He et al. (2017). Mask- CNN is from the two stage detection family of CNN's and breaks the detection into two stages, generating regions of interest (ROI) in the first stage and then classifying the objects in these regions in the second stage. The Yolov4 model belongs to the single stage detectors family, which predicts bounding box coordinates and the classification in one evaluation. While the two stage detection models are meant to result in more accurate detection's, the single stage detectors work quicker and still produce strong results especially when being used on videos. Training these models is just as important as choosing the right one to use when it comes to producing accurate results, and a good technique for doing this at the moment is transfer learning. Transfer learning is a method which allows for accurate training of a CNN model without a large data set to train on by borrowing a pre-trained model, trained on a similar domains data set which is much larger and the fine tuned to achieve classification of the smaller data set Pan and Yang (2010). The specific version to use is that of inductive transfer learning as that of inductive transfer learning as that is the version discussed here and proven to be effective In terms of speed detection in videos, one of the most up to date methods is using a detect and track technique which has achieved strong results. To achieve these strong results, one technique is that of using a Fast R-CNN to detect objects and then a pipeline of tracking, alignment and measurement to track and achieve speed detection Biswas et al. (2019), however other projects have used different techniques which are more in line with the solution approach as shown in Section 2.2.

2.1.1 Mask RCNN

In terms of CNN models, the Mask RCNN is meant to produce some of the most accurate results in modern computer vision He et al. (2017). It is a simple architecture compared to it's over engineered competitors and works very well with transfer learning on smaller data

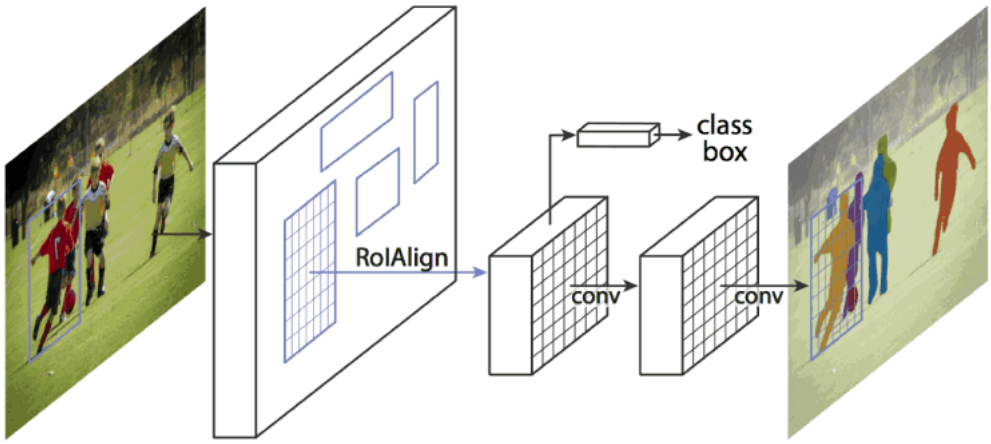

Figure 2.1: Overview of Mask RCNN structure

sets to still produce strong results Sumit et al. (2020). It is an extension of the Faster RCNN model, while adding a small overhead to it, in the second stage of it's detection it runs mask segmentation in parallel with bounding box detection. This means that as well as object detection it produces results for instance segmentation.

This algorithm using two stages to carry out object segmentation, the first stage being a region proposal network (RPN). It's extension previously mentioned is then that in the second stage, in parallel it creates bounding boxes as well as then masks for each object. The masks are created through a fully connected network (FCN) which unlike it's counterpart the fully connected (FC) layers retains it's spatial layout without collapsing into a vector representation, requiring fewer parameters and resulting in better accuracy. As well as this, Mask RCNN replaces the RoIPool layer, which extracts feature maps from regions of interest, with RoIAlign. Change resulted in better performance accuracy due to some of the changes it brings to the process like using bi linear interpolation on RoI bins to avoid hash quantization which can harm results. The architecture of Mask RCNN is shown in Figure 2.1.

2.1.2 Yolov4

In comparison to the Mask RCNN model there is the polar opposite, the Yolov4. This model focuses on being able to be trained and run on a single GPU or VPU and still produce accurate and fast results. Being faster makes it ideal for use in real time scenarios, making it more applicable in a wider range of use cases, especially those which replace human interactions with real time events. It also behaves differently to the Mask RCNN model and while not being as efficient on small data sets and transfer learning it has been proven to easily detect instances of object in images which are smaller or blocked and harder for Mask RCNN to identify Sumit et al. (2020).

The original Yolo model unified the separate components of object detection into a single network. This had the effect of the model being quick enough of real time detection, while having mAP scores which were twice as high as the other real time detectors around at the

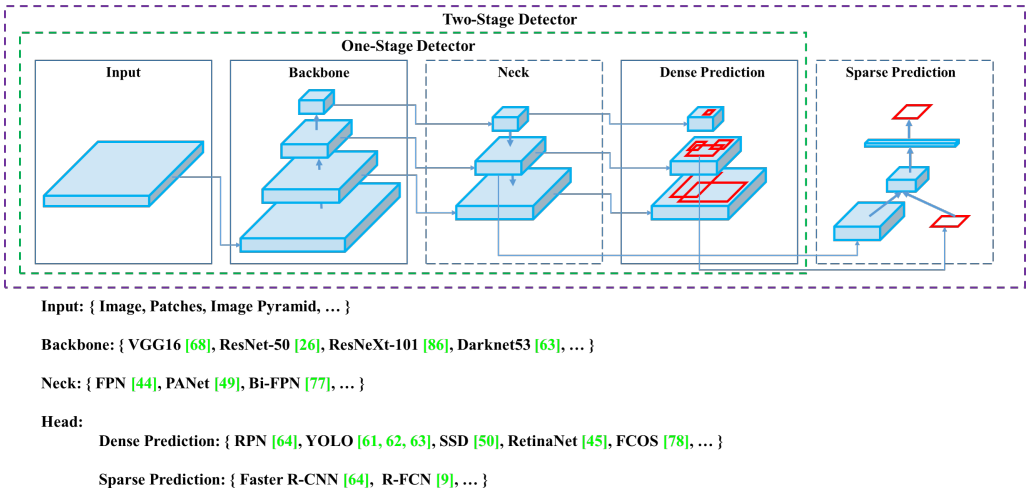

Figure 2.2: Overview of Yolov4 structure

time Redmon et al. (2016). Each new version of Yolo made improvements on it's performance. Firstly Yolov4 (aka Yolo:9000) made the improvements of introducing darknet-19 which is a new version of the network which allowed for the network to perform classification quicker, then used a a word tree to combine data sets in a hierarchical style resulting in more optimised training Redmon and Farhadi (2017). Yolov3 was then released, which introduced a new feature extractor in darknet-53 which outperformed the competitors in both accuracy and speed Redmon and Farhadi (2018). The latest improvement has come in the form of Yolov4, this saw a number of improvements with CSPDarknet53 as the backbone to increase accuracy, included an SPP block to enhance the receptive field of the model, separates features and causes no cost in network operation speed. It also replaced the FPN with the more modern PANet for the neck of the model while retaining the head layer used in Yolov3 to increase the accuracy and performance to what it is now Bochkovskiy et al. (2020). The architecture of Yolov4 is shown in Figure 2.2.

2.1.3 LC-RNN

Another technology to mention is that of LC-RNN: A Deep Learning Model for Traffic Speed Prediction which combines a CNN and RNN into a single model to predict traffic speeds. The architecture made use of both of these types of attributes to create a model which can accurately analyse and predict traffic speed on a road. The model proposed consists of a look-up convolution layers, a recurrent layer, a periodicity extraction layer and a context extraction layer. The combinations of these layers allowed for good use of a range of factors to allow it to achieve a RMSE (root mean squared error) of 5.274, showing that it's predictive capabilities are possible Lv et al. (2018). The structure of LC-RNN is show in Figure 2.3.

2.2 A description of the project in the context of existing technologies and systems

The first system which this project can be compared to is that of Transfer Learning for Instance Segmentation of Waste Bottles using Mask R-CNN Algorithm. This is because the task completed in this project was that of classification of water bottles in images. This work describes the process of using image augmentation and transfer learning to create good results

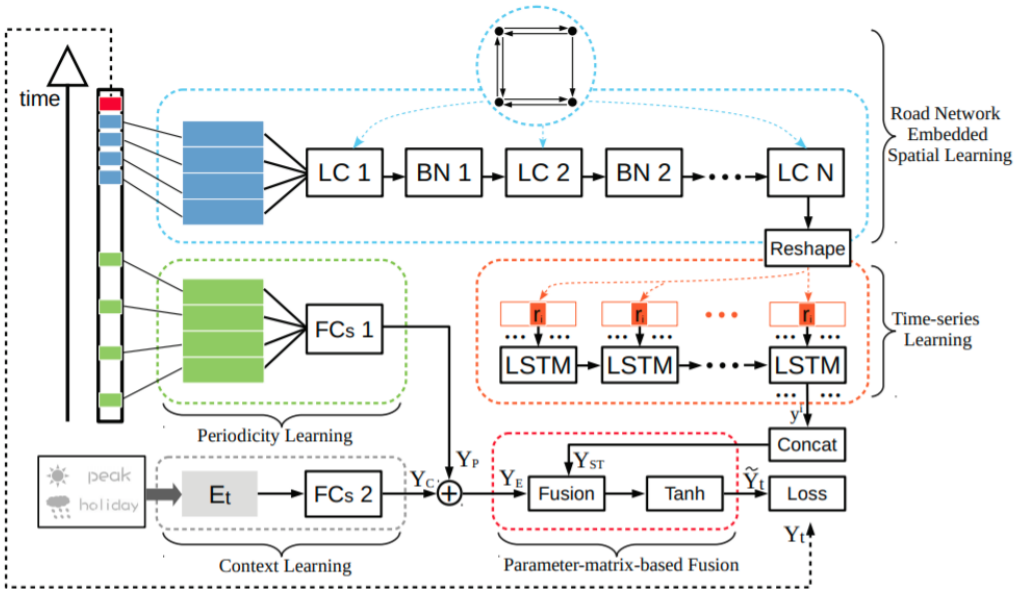

Figure 2.3: Overview of Yolov4 structure

in for the classification of plastic bottles from a small data set. This classification and transfer learning task will be the first challenge as good detection is needed to produce accurate speed predictions .

The second is that of Measuring Traffic Speed With Deep Learning Object Detection by Rohit Sharma who used box speed detection and Machine Learning to produce a speed prediction model for traffic. This shows that speed prediction is achievable and provides a method for producing these results. He used deep learning to detect the objects, and then moved on to carry out box speed calibration with machine learning to achieve this speed detection which is the second goal of this project Rohit (2019). As the technique used in this project has been used with Machine Learning, it makes sense that this technique would be the best suited technique for use with Neural Networks.

Another project to quickly mention is that of Visual Object Detection For Autonomous UAV Cinematography Patrona et al. (2020) which has achieved very strong results in rowing boat detection. This project managed to achieve mAP scores of up to 76.15% using a variety of Yolo and SSD Liu et al. (2016). This project shows that creating a strong rowing boat detector is achievable.

Finally, a project which can be refereed to is that of Vehicle speed prediction with RNN and attention model under multiple scenarios. This project is notably not a computer vision problem but still makes use of other sensors to show predictive capabilities using LSTM networks. The sensors used are those on a smart car which can record the metrics of other cars around it like acceleration and wheel speed. This data is then used by an LSTM neural network to make predictions on what the detected surrounding cars will do. Then results of this project show that it this technique could be used to perform predictions on a range of attributes like expected maneuvers to achieve RMSE scores of up to 0.7 Shih et al. (2019). While this left the domain of computer vision, it does however show the strength of Neural Networks and specifically Recurrent Neural Networks in making use of time series data to predict things like acceleration and maneuvers, this shows that LSTM networks can work on solving speed problems.

2.3 An analysis of how the review is relevant to the intended system

The review is made relevant to the intended system as it shows how other projects and technologies have achieved some of the goals of this project. The classification problem is shown to have produced strong results from the state-of-the art models and can be trained for a specific task on a small data set accurately. The is also the matter of speed prediction where the technology has been shown to exist in many different forms, whether that is using classic computer vision with detect and track techniques or with machine learning. This literature review shows that these technologies exist and that the goal of speed detection of rowing boats is achievable.

2.4 A critique of existing work compared with the intended work

In terms of object detection in literature there are few issues with this, the models created produce good results and are easy to train and run while upholding these accurate results. Object recognition of objects on water can usually see negative results, but for Mask RCNN, these effects do not have a great impact of the accuracy as shown in Nie et al. (2020). One things to mention in regards to Patrona et al. (2020) is the scale of the project, the scale of the data set and resources available are substantially larger than that provided in this project, meaning that other techniques will be used to achieve boat detection results. The models created were also not made available at the time the project was carried out. Some other critiques lie in the speed detection domain. The systems that have been proposed are created from long pipelines which can be slow and difficult to implement like the example shown here Biswas et al. (2019), which is improved upon in Rohit (2019) by using a two stage process to create accurate results. However these results are limited in the way that they are not very transferable, the camera has to be fixed of this to work which is not possible in a boat race. One way to solve this issue would be to use a NN as used in Lv et al. (2018), which would allow for more contexts to be able to be implemented. However another thing to note is that if box speed calibration is being used, MLP's can be used as well and RNN's due to it being a time-windowed approach Gers et al. (2002). Another thing to note is how the LC-RNN was limited in that it was not working out each individuals speed but current and future average speed of the vehicles in a lane, which will have to be manipulated to carry out speed predictions on individual boats instead.

2.5 Summary

In summary there are two main CNN models which prove to be useful from the literature review, Mask RCNN He et al. (2017) and Yolov4 Bochkovskiy et al. (2020) which are the most up to date and promising for the task of boat detection. As well as this, projects then show that the speed prediction technique of box speed calibration appears to be the best suited technique for integration with the use of Neural Networks. Limitations of projects have been defined. Many speed prediction methods do not have the ability to predict speed from moving cameras and be easily transferable, which is a limitation that can hopefully be overcome with the use of Neural Networks integrated with these other techniques.

Chapter 3

Methodology

3.1 Problem description

The problem is to detect the speeds of boats in an input video stream of a boat race using neural networks. This problem can then be broken down into two further issues. The first being boat detection. This step requires a CNN to be trained to detect boat objects and return bounding boxes. Next a neural network will then have to take these bounding boxes and carry out what is known as box speed calibration, which compares the differences in the bounding boxes to detect what speed the boat is moving at.

Training a CNN to carry out detection of boats requires several steps. Firstly a data set of boats much be found which will need to contain enough boats to train the model on. Next this data set must be labelled correctly, using software to go through the data set and create the correct labels for where boats are in the images. After this is done a CNN will have to be implemented, and then trained on the data. Validation metrics can be used on the model at each stage of it's training to find when it peaks in accuracy.

Creating a speed predictor requires similar steps to creating a boat detector as they are both neural networks and require a similar process. Therefore firstly, a labelled data set is required which can be created using the previously defined detector. Pre processing of the data is then required to make it suitable for training. Then a neural network is required to be created, which will then be trained on the data collected and tested to make sure it has a sufficient accuracy.

After these steps are carried out, a boat speed detector will be ready to implement and deploy.

3.2 Algorithms, tools and technologies

To help create a speed detector, there are a lot of tools which can help accelerate the implementation of this project. These vary from being standalone software to pre written implementations and programming libraries to help create what is needed.

3.2.1 Programming Languages

For the main language, Python is the clear choice. The repositories being used have been written in python, as well as the many libraries that can be used for developing neural networks like Tensorflow and Keras. Despite being famously slow to run it is also easy to write in, meaning that development time will be lower and more time can be spent on experimentation

and fine tuning. As well as Python, Power shell has been used to automate repetitive tasks, allowing for more time to be spent on the other tasks and for more tests to be carried out.

3.2.2 VGG Image Annotator

A software tool was needed to help label the data set which was given. The tool chosen for this was VGG Image Annotator Dutta et al. (2016) as it is easy to use and has methods of creating both bounding box and mask labels. This is required as the two CNN models being used require these two different formats for training. This software was used to label the rowing boat data set Li and Fei-Fei (2007) which came with no labels.

3.2.3 Mask RCNN

Implementing something like Mask RCNN from scratch is a daunting task, luckily it has already been done. There is a git repository in which it has been implemented and exposed for free use Abdulla (2017). This allows for the time consuming implementation to be avoided to more effort can be put into the other sections which are being worked on. To train Mask RCNN, transfer learning was used. This was easy to implement as the new names given did not have to fit the old weights. This allowed for a new class of only rowing boat to be passed along with the training and testing data sets. The model is loaded with weights training on the COCO data set Lin et al. (2014) and then fine tuned using the provided data set Pan and Yang (2009) . This was done in stages of freezing layers and training the rest, so first the head layer was trained, then the 4+ layers and finally all layers. After it had been trained it could then be used to carry out boat detection's in videos and images.

3.2.4 Yolov4

Similar to Mask RCNN, implementing Yolov4 from scratch would have been a time consuming problem. Instead there was also an implementation of it in the form of a Python package Hong (2021). This package exposes a Yolov4 model, which can load weights, be trained and make detection's in videos and images. The model however, does no allow for class names which do not match the model weights. This means that when carrying out transfer learning, the boat class in the COCO data set Lin et al. (2014) must be fine tuned to allow it to detect rowing boats instead of the cruise and leisure boats it was first taught to detect Pan and Yang (2009). With both the Yolov4 and Mask RCNN implementations, there was the option of GPU acceleration which proved to massively speed up the process of training and making predictions Baldini et al. (2014) as it is shown to do. This speed up allows for more experimentation in this case and also helps to push the project towards the real world applications which may require real time processing of the input video streams.

3.2.5 mAP

To carry out evaluations on the models created, mean average precision is a useful technique to provide incite into how well the models predict what objects are where in a scene. To do this there is a useful git repository Cartucho et al. (2018) to help produce results on how well the models have been trained. This requires the results the models to be produced and saved to a text file which is a small addition to the detection. Using this the program then takes these results, compares them to the labels, showing the mAP scores produced and how well the predictions were made visually. This is a very useful tool for presenting the different scores of the models to compare them and also allows for greater analysis.

3.2.6 MLP and LSTM

To use MLP and LSTM models, the Keras python library was used Chollet et al. (2015). This allows for the neural networks to be custom built for the problem being solved. To create the models, the sequential class is used, this allows for all the layers to be contained and fed into one another. Then the layers are required, using the Dense and LSTM layers as provided by the library, the last layers is not given an activation function, as this is a regression problem and a continuous output is needed. After the model is defined it can be compiled and trained. Keras also provides useful tools like early stopping and history callbacks which can be used to help train and analyse these models better. These models can also be loaded and saved using a library called Pickle Van Rossum (2020), allowing these models to be shared and used amongst programs.

Both models were used in this problem as sometimes as it is a windowed time series problem in the form of box speed calibration. With this sort of problem the natural response is to use a Recursive Neural Network (RNN), but can also be solved by Multi Layer Perceptron models (MLP's) Gers et al. (2002).

To record how well these two different models perform, R2 score is an ideal option Van Herwaarden et al. (2014). R2 score is a metric of how well the model can describe the regression problem. To implement the use of R2 score, a Python library called Scikit-learn was used Pedregosa et al. (2011) which is a library containing a wide variety of tools for Statistics and Machine Learning. The R2 score from the metrics library was used by passing it the actual labels of the speeds and the predicted labels of the speeds, it the calculates the R2 score between these labels and then produces the final result which it prints out.

3.3 Implementations

Even with the tools and technologies provided there is still a lot of implementation to be carried out to piece these together to achieve the goal that had been set out. For this project many things had to be created for different reasons like making technologies integrate with one another or to create automation to allow for a greater freedom when experimenting so that more can be gained from it. The integration stages are very important as there are a lot of separate but essential technologies which need to be connected manually using new code.

3.3.1 Mask RCNN

The Mask RCNN repository used did not have an inbuilt method to produce a set of results which could be used for evaluation. For this reason a function was required which could use the trained model to carry out predictions on a validation set and then output those predictions to a place where they could be accessed by the mAP repository Cartucho et al. (2018). In this function, it takes a model and loads each image in the validation data set. For each image it makes a prediction on the details of the boats, then writes these results to a file for it to be used in the mAP repository.

```
1
2 def eval(model):
3     import glob
4     import cv2
5
6     build_ground_truth()
7
8     val_dir = ROOT_DIR + '\datasets/rowing2/rowing_val/*.jpg'
9
```

```
    jpgFileList = glob.glob(val_dir)

    for img_file in jpgFileList:
        print(img_file)
        image = cv2.imread(img_file)
        r = model.detect([image], verbose=0)[0]

        print(r['rois'])
        print(r['scores'])

        length = len((ROOT_DIR +  '\datasets/rowing2/rowing_val/'))
        dest = img_file[length:-4]

        filestr = "mAP/input/detection-results/" +  dest + ".txt"
        fileObj = open(filestr, "w")

        for rois, score in zip(r['rois'], r['scores']):
            string = "boat " + str(score) + " "  + str(rois[1]) + " " +
    str(rois[0]) + " " + str(rois[3]) + " " + str(rois[2]) + "\n"

            fileObj.write(string)

        fileObj.close()
```

Listing 3.1: Code used to expose Mask RCNN predictions for evaluation program

3.3.2 Yolov4

The label format given by VGG was different from the labels required by the Yolov4 implementation. Because of this, a python program was required to take in the labels and transform them so they can be used with the Yolov4 module. This code works by loading in the CSV file which holds the initial labels and unpacks each images labels. With this information it gets the image and it's dimensions, which are used to convert the bounding box points from pixel values to decimal values.

```
import pandas as pd
import json
import cv2
import os

val = pd.read_csv('rowing2/validation_set.csv')

fileobj = open("val_rowing.txt", "w")

prev_file = None

for f, r in zip(val['filename'].tolist(), val['region_shape_attributes']):

    img_dir = "rowing2/rowing_val/" + f
    img = cv2.imread(img_dir)

    if img is None:
        raise Exception("Could not load img")

    img_h, img_w, _ = img.shape

    data = json.loads(r)

    try:
```

```
        x = data['x']
        y = data['y']
        w = data['width']
        h = data['height']
    except:
        string = "\n"+ str(f)
        fileobj.write(string)

        continue

    print(x, y, w, h)
    print(img_h, img_w)


    if "x" not in data:
        string = str(f) + "\n"
        fileobj.write(string)
    else:
        x, y, w, h = x/img_w, y/img_h, (x+w)/img_w, (y+h)/img_h
        print(x, y, w, h)

        if(w > 1):
            w = 1.0
        if(h > 1):
            h = 1.0

        if ((x > 1) or (y > 1) or (w > 1) or (h > 1)):
            raise Exception("Dimension is greater than one")

        if f == prev_file:
            string = " 8," +  str(x) + "," +  str(y) + "," + str(w) + ","
    + str(h)
        else:
            string = "\n"+ str(f) + " 8," +  str(x) + "," +  str(y) + ","
    + str(w) + "," + str(h)


        prev_file = f

        print(string)
        print()
        fileobj.write(string)


fileobj.close()
```

Listing 3.2: Code used to reformat labels for Yolov4 training

3.3.3 Evaluation

To help speed up the boat detection evaluation process, Power Shell automation was used. Due to the many times this needs to be repeated it is essential that the evaluation process is automated, so that more experiments can be carried out and better results can be found. The Power Shell script runs the python scripts which carry out detect ions on a validation set and save the results to a file. The script then moves those results to the mAP repository and runs the main files, in two ways either showing the images or not depending on user input.

```
python "eval_model.py"
Copy-Item "mAP\input" -Recurse "../mAP" -Force
```

```
3
4 $see_images = Read-Host -Prompt 'Do you wish to see evaluation images? (y/
    n)'
5 if ( 'y' -eq $see_images )
6 {
7     python..\mAP\main.py
8 }
9 else {
10     python ..\mAP\main.py -na
11 }
```

Listing 3.3: Code used to automate evaluation task

3.3.4 Speed Predictor Model

One bit of code required by for speed predictions to be carried out properly is one which is able to group the different bounding boxes in consecutive frames. To do this the two lists of bounding boxes are classified on which one is smaller. Using this the y values are used to decide which bounding boxes match those in the next consecutive frame. The Y value is used because the boats are divided by lanes which is an important method of tagging as shown in Zhu et al. (1996). The method then returns a Numpy array Harris et al. (2020) which contains the linked bounding boxes.

```
1
2
3 def match_boats(previous_boats, new_boats):
4
5     ret = np.zeros((0, 8))
6
7
8     if len(previous_boats) > len(new_boats):
9         small_list = new_boats
10         big_list = previous_boats
11     else:
12         small_list = previous_boats
13         big_list = new_boats
14
15     for b in small_list:
16         opt = big_list[0]
17         opt_dist = np.linalg.norm(b[0] - big_list[0][0])
18         for x in big_list:
19             dist = np.linalg.norm(b[0]-x[0])
20
21             if dist < opt_dist:
22                 opt = x
23                 opt_dist = dist
24
25         ret = np.append( ret ,np.reshape((np.append(b, opt)), (1, 8)),
    axis = 0)
26
27
28     return ret
```

Listing 3.4: Code used to pair boats in consecutive frames

With the boats matched, the new boat objects now need to be labelled with their speeds. Due to the POV chosen for this, the boats can be identified by where there are in the y axis of the frame Zhu et al. (1996). The splits (aka speeds) for how fast the boat have been travelling can be found in the graphics of the videos provided, as the samples are at the 500m

marks there is time values for each boat, which can be put into a dictionary and used to label all the boats on where they are in the y axis. The code below does require editing between clips due to how the boats speed change. Also to node the code is contained in a block in a Jupyter notebook Kluyver et al. (2016).

```
1
2  speeds = {
3      "GER": 1.31,
4      "GBR": 1.32,
5      "NED": 1.32,
6      "CAN": 1.34
7  }
8  labels = np.zeros((boats.shape[0]))
9
10 for i, (_, _, y2, _, _, _, _, _) in enumerate(boats):
11     if y2 >= 715:
12         labels[i] = speeds['GBR']
13     elif (y2 < 715) and (y2 >= 662):
14         labels[i] = speeds['CAN']
15     elif (y2 < 662) and (y2 >= 606):
16         labels[i] = speeds['NED']
17     else:
18         labels[i] = speeds['CAN']
```

Listing 3.5: Code used to pair boats in consecutive frames

3.4 Experiment Design

With all the relevant tools, technologies, algorithms and extra implementations in place, it's then time to integrate everything together to create a system which can carry out accurate speed predictions. Within this there are two main components, the boat detection which then leads into the speed detection by passing bounding boxes of the detected boats which can be used in box speed calibration.

3.4.1 Boat Detection

Firstly the boats data set must be scaled so that the pixels are consistently 800x600 as this is essential for good training of CNN's and that the images still have a large enough size Hashemi (2019). Next, with the boats data set labelled using the VGG software Dutta et al. (2016) and the Yolov4 labels re-formatted the models can be trained. GPU acceleration is used with a RTX 2070 to speed up training times on the models Baldini et al. (2014). The training is carried out using transfer learning, by loading weights pre trained on the COCO data set Lin et al. (2014) and then fine tuning them on the new labelled data set Pan and Yang (2009). The Mask RCNN model is trained at different stages, each stage freezing certain layers and training the rest to allow for more control over the training to produce greater results Jaikumar and Ojha (2020). Finally with the models trained it is then time to evaluate how well they perform on new images. To do this the automation script is used to automatically run the python evaluation scripts and get the mAP scores from the mAP repository Cartucho et al. (2018). Depending on the mAP scores the models are then evaluated and the model with the best scores are then used to carry out boat detection's on a sample of videos from the 2012 men's eights second heat. The camera angles used in the samples are taken from a higher point of view (POV) in which the boats are clearly divided by their lanes and there is little overlapping, which is an ideal POV for a speed detection problem Zhu et al. (1996). The

bounding boxes detected in each frame are then saved to a file to be pre processed for speed detection in the next stage.

3.4.2 Speed Detection

Now that the bounding boxes have been generated, they can now be labelled with the required speeds as implemented before. The data given is first scaled to allow for better training of the Neural Network's Bishop (1996). With the labelled data in place the models then need to be generated which are MLP and LSTM Gers et al. (2002) using the Python library Keras Chollet et al. (2015). With the models generated they will then be trained on a sub set of the data set for training and evaluated on another sub set of the data for validation Bishop (1996). The validation set is used at each epoch to measure how well the model performs on this subset. Using this the model can also make use of early stopping, in which it will detect when the validation scores are not improving and then stop the training of the model Sarle (1996). As well as this a history callback is used to allow to user to see how training is performing and make reasonable adjustments. After the model has been trained it can then be evaluated using R2 Score. This score measures how well a model describes a regression problem Van Herwaarden et al. (2014). Then depending on the score achieved two things can happen, either the score is undesirable and the model is retrained, otherwise the model is then saves using a Python library called Pickle Van Rossum (2020) to be later used to detect speeds in a video.

3.4.3 Bringing It All Together

With the two models created and made, it is now time to carry out boat speed inference on videos of rowing races. As the videos used so far have been from similar POV's, the video used the race segment of the finishing line which continues this trend of looking at the race from above. The video is first passed to the chosen and trained CNN model which detects the bounding boxes of rowing boats in the frames which are first normalised to work with the models. Then the 'match boats' functions pieces the frames together which are then passed to the speed detection stage. First the bounding boxes are scaled with their labels by the same transformer used in the pre processing of the speed data so that the results are consistent. Then predictions are made on how fast each bounding box pair has travelled. The bounding box detection's and speed predictions are then drawn onto the video and either saved or immediately shown so that the user can see how fast each boat is travelling, successfully carrying out speed predictions on a boat race. Fig. **??**

3.5 Summary

Two main types of components made up the overall system and experiments. The components which were pre made and those other parts which were implemented specifically for the system. The components which were pre made covered the CNN's and main libraries and tools used to implement the required technologies. As well as this there were useful tools to help with the generation of data and other activities along the way. The other set of components are the ones manually implemented for this system which allow all the technologies to be integrated together to produce the final system and carry out the required tasks. Overall the system detects boats using CNN's after being trained with transfer learning. Then the Neural Network's carry out boat speed calibration after being trained as well to made predictions on boat speeds.

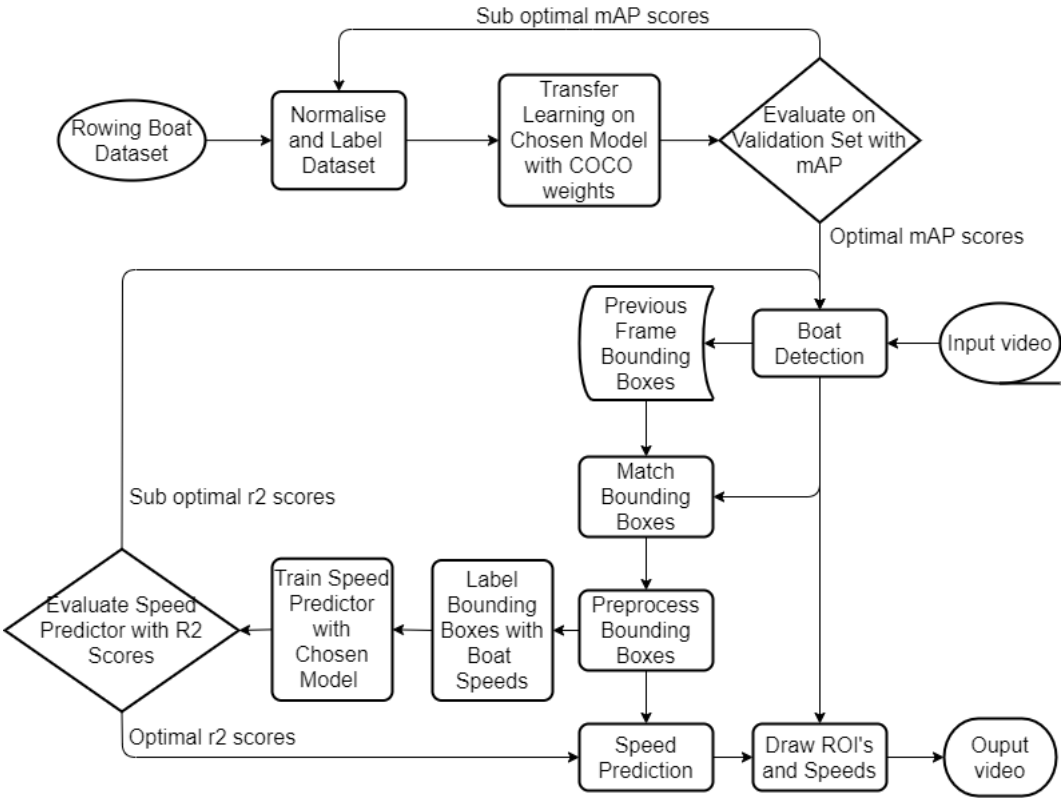

Figure 3.1: Overview of methodology

Chapter 4

Results

As the project was made up of two different major components, it makes sense to make use of carrying out component testing on each part of the system and then an integration or system test to make sure that overall the system works and can be used to carry out the task. This is also useful as each stage of the project has it's own metrics, as boat detection uses mAP while speed detection uses R2 score. This scrutiny at each step also allows for a deeper understanding into the entire system and how it's behaviours can be analysed.

4.1 Boat Detection Results

Two main CNN models were trained and tested in this experiment, the Mask RCNN and Yolov4 models. These models are very different, Mask RCNN is better suited to transfer learning Sumit et al. (2020) and is well know for being one of the most accurate models in use at the moment He et al. (2017). However Yolov4 is another model which prefers a different domain, it's main uses are in live situations due to it's high frame rate Bochkovskiy et al. (2020) but has also shown behaviours like being able to detect objects which other competitors can not Sumit et al. (2020). For the scenario, it would have been ideal for Yolov4 to have been implemented, however the small data set mean that it is more likely that Mask RCNN will work better. For this reason they both had to be tested to see if Mask RCNN was needed, or if the quicker Yolov4 was able to be used.

4.1.1 Mask RCNN

When training the Mask RCNN model, it only took training up to the 4+ layers for it to peak. This will be because the convolutional and lower layers were already able to extract features from the images because the model had already been trained on the coco weights Lin et al. (2014) especially because boats are already in the COCO data set. The head layers did require training as the output was very different - only one class was being detected instead of the entire COCO data set. With the training of several different combinations of layers done and their weights saved they could then be evaluated using mAP. The training of the HEAD layer individually saw little improvement but when the 4+ layers were trained it saw a great jump in accuracy, all shown in the table below. After this was trained, further training saw very little improvement beyond what was created in epoch 90.

With the training done for this model, the detection results can be seen visually in a video stream. Due to the good point of view the video is coming from Zhu et al. (1996) the model seemed to make almost perfect detection's on the boats in the race, with no false positives being seen. These behaviours are exhibited in Figure 4.1. The only drawback from these

Table 4.1: Mask RCNN mAP scores

Mask RCNN	
Model	Score (%)
M30_HEAD	31.22
M60_HEAD	31.82
M90_4+	**51.11**
M105_4+	49.17
M105_ALL	49.56

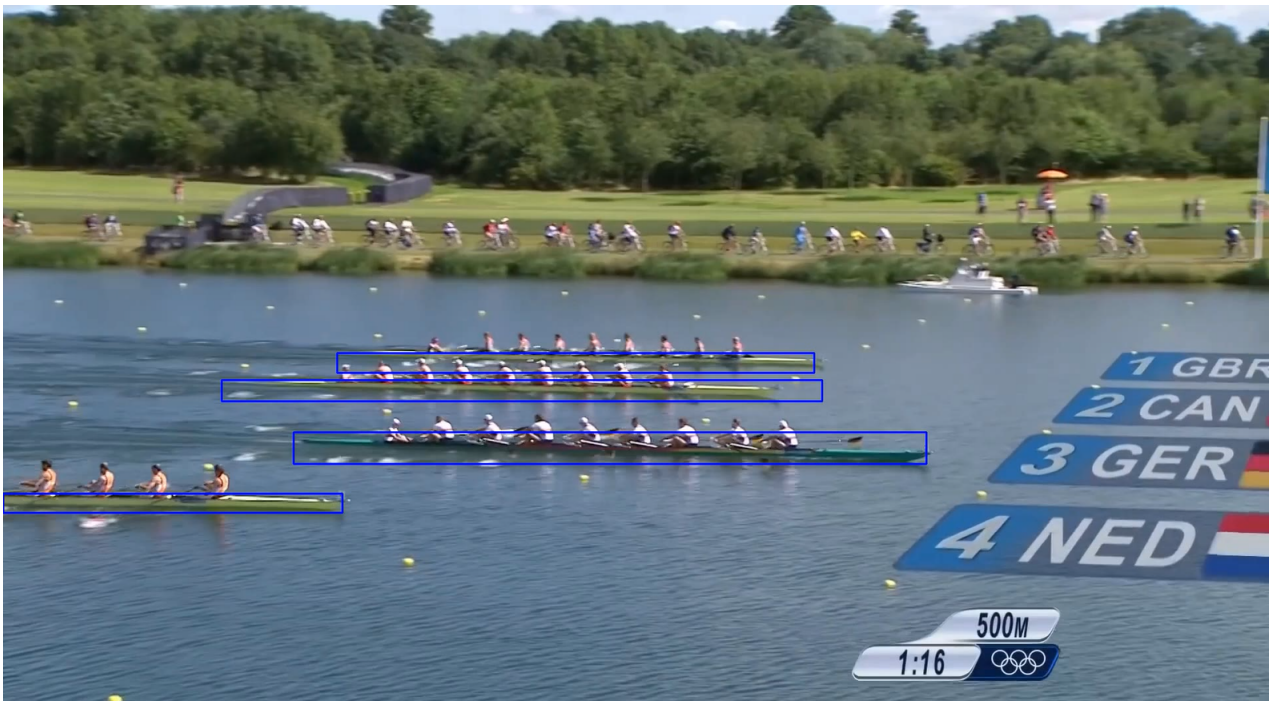

Figure 4.1: Evidence of Mask RCNN Behaviours

results is however, that it cannot be used for real time detection as it runs to slowly to be used this way.

4.1.2 Yolov4

Training Yolov4 on such a small data set was difficult, mainly for the reason that it does not respond well to small data sets Sumit et al. (2020). Due to this and the model already having been trained on the COCO data set, it was very quick for it to hit the maximum mAP scores within 6 epochs. After 6 epochs the accuracy dropped and then the model froze as improvements were not being made and it was avoiding over fitting. With a larger data set it would have been nice to see the model work well and would allow for live use if the system. This would be an improvement to make in the future, maybe combining multiple data sets of rowing boats.

Like the testing of Mask RCNN, it is also good to visually examine what how the model is behaving. And as expected from the mAP results, it was making a lot of mistakes. Most of the boats were not being detected by the model, with the detection's looking glitchy and

Table 4.2: Yolov4 mAP scores

Yolov4	
Model	Score (%)
M4	25.51
M6	**25.68**
M8	24,58
M10	24.58

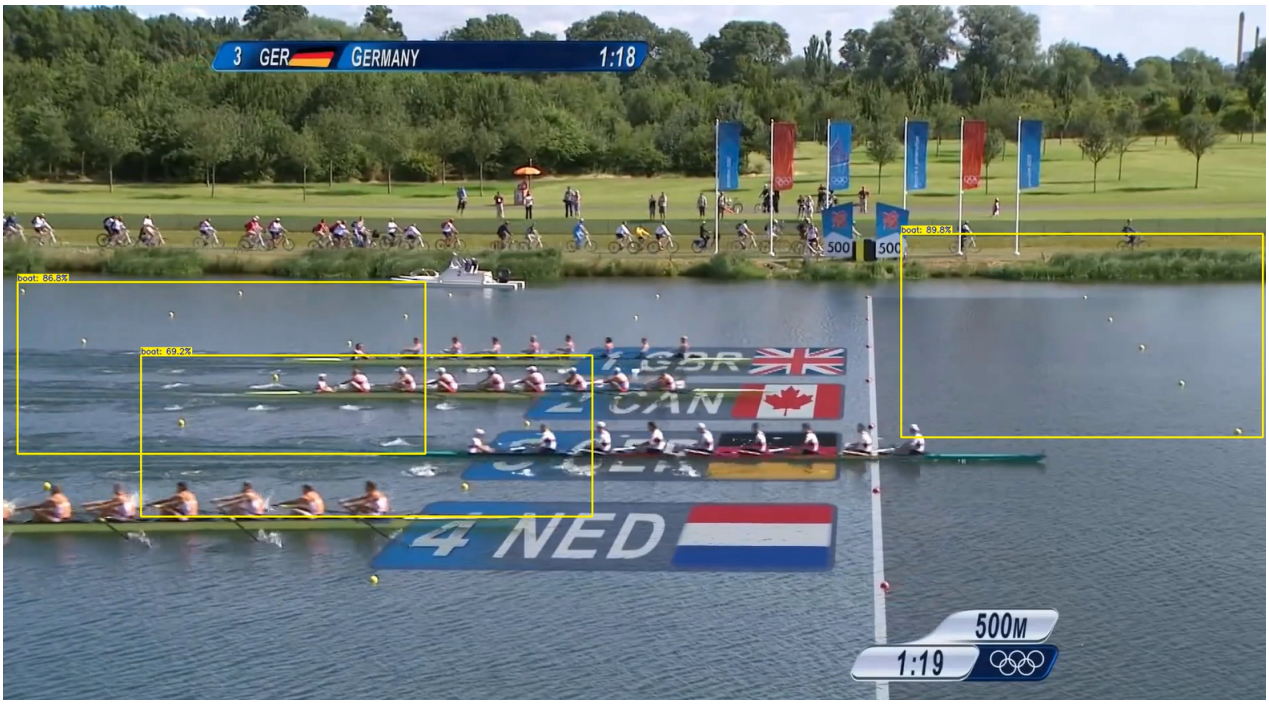

Figure 4.2: Evidence of Yolov4 Behaviours

inconsistent. As well as this there were examples of false positives being detected. An example of some of the behaviours of Yolov4 are shown in Figure 4.2 While being able to be run in real time which was the goal, Yolov4 was unusable after being trained on the data set which is known to be an issue with the Yolov4 model Sumit et al. (2020).

4.2 Speed Detection Results

With the ideal boat model and weights identified, the next stage was identifying which speed detection model works best for box speed calibration on rowing boats. The models used for speed detection were MLP and LSTM as both have been proven to work well on windowed time series problems Gers et al. (2002). Both models were trained on the same boat speed data set but the LSTM one had an extra temporal dimension, as it is designed for time series problems Hochreiter and Schmidhuber (1997). With both the models trained on the same data set they were then scored using r2 score which is useful for regression problems as it scores the model on how well it represents the problem space Van Herwaarden et al. (2014).

Using R2 score as a metric, it is clear that MLP out performed LSTM in this task. Being a simple model with a well chosen time window, MLP managed to learn the problem better

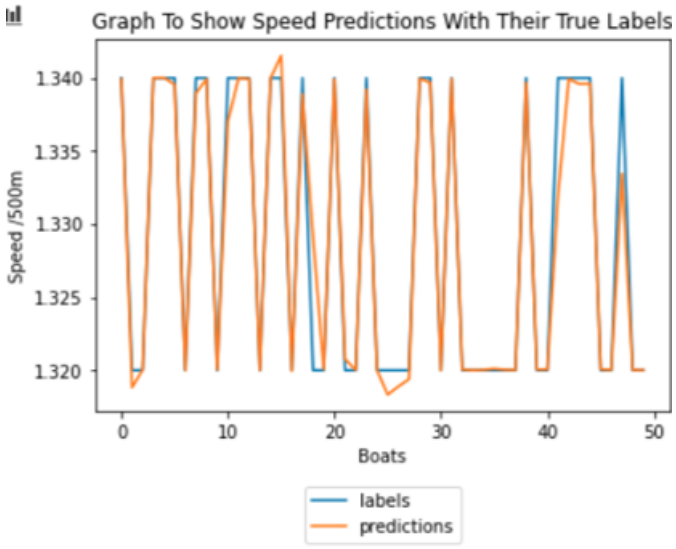

Figure 4.3: Graph of Regression Problem

than LSTM and is therefore better design for detecting the speeds of the rowing boats in the races. However LSTM may have performed better given a different setup, perhaps if it was using it's own memory instead of that given to it in the box speed calibration problem. This would be an issue to explore in future works. But for now, MLP is showing strong scores for being able to detect the boat speeds in a boat race. This is also shown in Figure 4.3 which shows the predicted speed and actual speed prediction trends being very similar and accurate.

Table 4.3: NN R2 Scores

Speed Detection R2 Scores	
Model	Score (%)
MLP	**0.93**
LSTM	0.72

4.3 Summary

Both stages of the system have been evaluated, with each experimental option being compared. In the boat detection sections the results showed Mask RCNN producing better mAP scores and also having better behaviours than Yolov4. Then in the boat speed prediction evaluation both LSTM and MLP models did well, but MLP ended up having better LSTM scores when solving this problem. Overall the system's best performing components were the Mask RCNN model for boat detection and then the MLP model for speed predictions. These components can then be used in the final system of which a frame of the output can be seen in Figure 4.4.

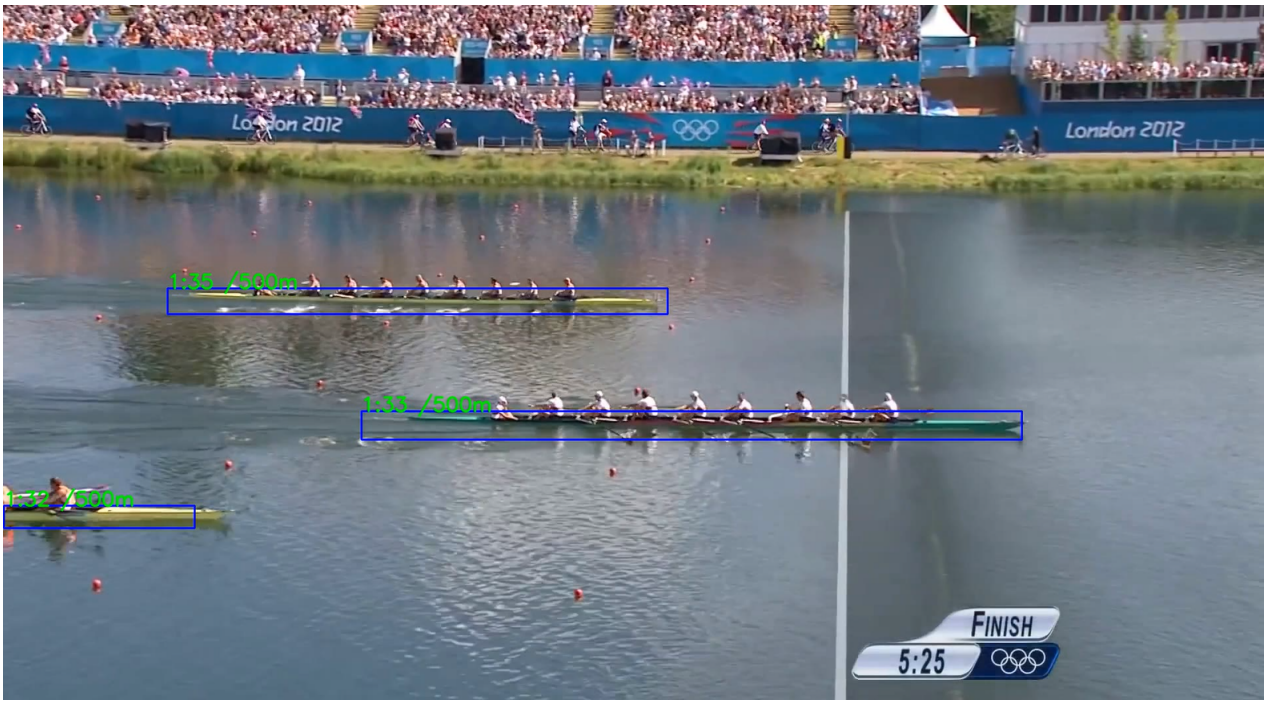

Figure 4.4: Proof of concept of Speed Prediction System Working

Chapter 5

Discussion and Analysis

With the results collected from the tests and metrics, the system and it's components can be analysed and the effectiveness of the system discussed. From a quick look at the strong results it is clear that something which is applicable has been created. As there are two main parts to the system and each is their own significant piece of work they can be discussed separately, allowing for a greater analysis of the work done. The significance of the findings is another important matter as well as the system was build with a use in mind and the limitations of the system for this use need to be discussed.

5.1 Boat Detection

The detection component of the system does achieve a high accuracy from using Mask RCNN. This was achieved due to transfer learning Pan and Yang (2010) and how well the model reacts to this process Sumit et al. (2020). The results created from the model show consistent and accurate results in detecting rowing boats and is very reliable, especially in the video stream chosen for the project. However where Mask RCNN has triumphed Yolov4 has failed. The model did not take well to the small data set as was expected but unusable. While it is quick, the detection's were unreliable and unusable in detecting rowing boats with false positives and inconsistent results. With a larger data set and training styles more specific to Yolov4 Bochkovskiy et al. (2020) this may have performed better, but as more focus was put on Mask RCNN and the techniques which allow that to perform best Yolov4 struggled to keep up.

5.2 Speed Detection

Both MLP and LSTM models performed well in this problem with even the results generated from LSTM being usable in this problem. However MLP managed to take win over LSTM in how well it learnt and made predictions on the speed regression problem. This may have been because of the short time window given to the problem, literally two frames difference which MLP has been know to perform well under Gers et al. (2002). LSTM also has the possibility of working well for methods other than box speed calibration in speed detection Shih et al. (2019) so it would be interesting to see how LSTM performs with a different speed detection technique compared to box speed calibration. However MLP using box speed calibration is more than enough to achieve accurate speed results.

5.3 Significance of the findings

The first key result gathered was that of the accuracy achieved for a rowing boat detector. Where other projects have achieved better results, this was also those projects single focus and there was a lot more infrastructure behind it which would have been unreasonable for this project. For example Patrona et al. (2020) achieved a best AP (average precision) score of 76.15%, which used a custom data set of 40786 images which were collected and labelled. As the data set created in this project was smaller by a magnitude of 156 achieving an mAP score of 51.11% is quite an impressive score given the context. This shows how useful transfer learning is in creating good usable models in small projects and how it can be used to make AI and deep learning more accessible.

The main result however is how the use of deep learning has been used to make speed predictions from a moving camera. While being able to achieve a strong final R2 score, this has been achieved in the past. What is different however is how the speed prediction is able to be carried out from a variety of different camera angles. The cameras being able to move and still provide good speed detection's is something that has been difficult to do in the past Hochreiter and Schmidhuber (1997) but using Neural Networks and their ability to understand complex problems Bishop (1996) this has been overcome. The use of box speed calibration Rohit (2019) has also helped with the process. This has allowed for the relevant information to be extracted to be given to the model in a non computationally expensive way and also allowed for the system to be transferable to other angles as the simple system allows for ease of use and a quick learning process.

5.4 Challenges

Discuss the key limitations and potential implications or improvements of the findings.

One challenge with the system at the moment is it's performance in terms of speed. The Mask RCNN model runs at 5 fps when on good hardware which means that the detection's will be performed slowly and it would make annotations and speed predictions glitchy and out of time with what is going on. This effect would make it unsatisfactory for every day users. However with a larger data set like the one found in Patrona et al. (2020) used on quicker models like Yolov4 which runs at 65 fps on good hardware Bochkovskiy et al. (2020), it would be more viable for main stream use.

Another difficulty is the CNN model accuracy, which again is lacking due to the small data set provided by Li and Fei-Fei (2007). This resulted in not only poor results for Yolov4 but also imperfect results for Mask RCNN. The use of transfer learning did provide good results in the context but it would be ideal to improve this accuracy. Improving the accuracy would allow for view from unsatisfactory camera angles which would help allow for consistent use through out the race because at the moment it only makes detection's at the 500m marks of the race as this is where the camera angle is most ideal for collecting metrics like this. As well as extending the system to carry out predictions of boat speeds in the Men's eights division, it would also be good to extend the system to work on other races like singles and Women's events. Another area it could include could be the Paralympics. It would be interesting to see how the system takes on new races and events, as it is a Neural Network so extending it should be easy, but as well as this it would be interesting to see if box speed calibration can be used in one event, and then immediately be used on another with little or no changes.

Anther limitation at the moment is that the system only collects one metric which is predicted boat speed. In rowing there are other useful metrics like stroke rate and lead in terms of boat length which would be interesting to include. These metrics are very important

to viewers of a race to be able to understand what is happening in the race and would therefore be useful expansions to the system and also interesting problems to explore.

5.5 Summary

With the results being shown in the previous Chapter 4 they are explained by the situations in which the technologies were used. Mask RCNN is better suited to work on smaller data sets and hence produced better results and then MLP had been shown to outperform LSTM in windowed time series problems like the one being solved. The findings have been shown to be significant with the speed detection working one a moving camera which has been a long standing challenge of speed predictions in computer vision, and being easily transferable due to the nature of the technique being used while retaining strong results. One challenge main challenge shown however is the limitation of not being able to be deployed on real time projects due to the use of Mask RCNN to gain better results.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The project was to use deep neural networks to detect rowing boats and make predictions about their speeds. It was to use a Convolutional Neural Network to detect rowing boats in a video stream, then from this use a technique to make predictions on how fast the boats are moving using another Neural Network. The detect rowing boats, the two CNN networks trialled were Yolov4 Bochkovskiy et al. (2020) and Mask RCNN He et al. (2017). These were to both be trained and compared to decide which one should be used in the system. Next was to create a speed prediction model, to do this both LSTM and MLP models were used to try and predict the speeds of the boats in the video streams using a technique called box speed calibration Rohit (2019).

The boat detecting components produced strong results for the given context. With a small data set of images of boats, good results are hard to create, but using transfer learning Pan and Yang (2010) a strong accuracy was achieved with an mAP score of 51.11. This shows how computer vision and deep learning are becoming more and more accessible to smaller projects with less time and funding, as in this case where a boat detector was created which was reliable enough to build a good system on top of. As well as transfer learning being key to the success of the detector the Mask RCNN model He et al. (2017) was another technology which worked well with transfer learning. However well the boat detector performed, that was not the main objective of the project but a key component. This main goal was to implement speed predictions using a Neural Network. From looking at previous work on the topic, it was found that box speed calibration was a technique which had had over machine learning techniques used on it and proved to be one of the most promising methods Rohit (2019). As with this, MLP has been known to work well in solving windowed time series problems such as the speed prediction problem being tackled Gers et al. (2002) and proved to be the most reliable Neural Network what evaluated compared to other achieving an R2 score of 0.93%. With both the components the system was successfully fed a video feed and was able to identify boats in the video as well as their speeds. As well as this the system was able to do this from moving cameras and also was easy to reproduce making it more transferable than other techniques.

6.2 Future work

On of the key improvements the future work should carry on from this project is improving the boat detection. It has been built to be accurate as shown in the Chapter 4 and work

well in post race analysis as it is slow. This can be improved specifically by creating a model which would allow this project to run in real time as this is a key problem holding it back from being applicable to the real world so that a end user can view live output and gain a better understanding of what is going on in terms of boat speed. It would also make the output seem more up to date as it would be able to update within the seconds that pass to create a smoother experience. However this improvement is one which is readily available given more support and which is know to be solvable.

The main future work to be carried out is then expanding the speed detection model to understand the speeds of boats when looking at different points of view. Currently the model is very effective at predicting speeds as shown in the Chapter 4, and it also works when the camera is moving so that a camera is able to scan across the race. However it has only been trained to do this from a single camera currently. To expand the system in to include more camera angles maybe the speed detection model can be expended to take in the camera angle as an input when making predictions about speed to help the system understand from where the view is coming from. As well as this it will need more camera angles to use in training and testing. Doing this would allow for the system to achieve more coverage of a race and mean that the investment in this system would result in the reward of the entire race having speed detection carried out.

Another improvement that would be nice to see would be to have the system work on different races. While the system is very good at detecting rowing boats and predicting their speeds, it would be interesting to see if it could be expanded to other races like canoeing or even car races as the models used can detect a variety of different objects which would be useful in their different domains. This would allow for more users to be able to make use of the speed metrics presented by the system and remove the cost of expensive equipment to the hosts of the races who can instead use this system to get the speed metrics they want cheaply and reliably.

Chapter 7

Reflection

One of the main skills learnt from carrying out this project was that of creative problem solving, this project pushed me to think for myself and be critical about what solutions would work. From this my research skills were improved as to carry out this critical decision making I had to gain a strong understanding about the cutting edge technologies being used along with their strengths, weaknesses and behaviours. This information would then be used to guide decisions made throughout the project and create the outcome which was achieved. As this was a large project another skill which was improved was that of organisational skills. This came from detailing the requirements and using an agile approach to completing the project which allowed for flexibility which proved useful as initial ideas changed and the project had to adapt around them. Using a Kan ban board was very useful and keeping a weekly journal allowed for good time management. One challenge not overcome was that of changing the CNN models to better suit the situation and even solve the speed prediction problem as well with the right adjustments, while this would have been ideal it was not necessary for the project to work and there was already a lot of work to do within the given time frame. If more time was available I would have done more fine tuning of the CNN models to gain stronger results to then compare to other projects because the results are not that impressive when compared to other projects which focus on achieving strong accuracy for detectors. The initial plan also deviated when it became clear that the data set did not allow for the Yolov4 model to perform well enough to be used, which was the initial idea so that it could run in real time. This resulted in Mask RCNN being used as well as even though the mask was unnecessary it performs very well on the data set available.

References

Abdulla, W. (2017), 'Mask r-cnn for object detection and instance segmentation on keras and tensorflow', `https://github.com/matterport/Mask_RCNN`.

Baldini, I., Fink, S. J. and Altman, E. (2014), Predicting gpu performance from cpu runs using machine learning, *in* '2014 IEEE 26th International Symposium on Computer Architecture and High Performance Computing', IEEE, pp. 254–261.

Bishop, C. (1996), *Neural networks for pattern recognition*, Oxford University Press.

Biswas, D., Su, H., Wang, C. and Stevanovic, A. (2019), 'Speed estimation of multiple moving objects from a moving uav platform', *ISPRS International Journal of Geo-Information* **8**(6), 259.

Bochkovskiy, A., Wang, C.-Y. and Liao, H.-Y. M. (2020), 'Yolov4: Optimal speed and accuracy of object detection', *arXiv preprint arXiv:2004.10934* .

Cartucho, J., Ventura, R. and Veloso, M. (2018), Robust object recognition through symbiotic deep learning in mobile robots, *in* '2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', pp. 2336–2341.

Chollet, F. et al. (2015), 'Keras', `https://keras.io`.

Dutta, A., Gupta, A. and Zissermann, A. (2016), 'Vgg image annotator (via)', *URL: http://www. robots. ox. ac. uk/~ vgg/software/via* .

Gers, F. A., Eck, D. and Schmidhuber, J. (2002), Applying lstm to time series predictable through time-window approaches, *in* 'Neural Nets WIRN Vietri-01', Springer, pp. 193–200.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del R'ıo, J. F., Wiebe, M., Peterson, P., G'erard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C. and Oliphant, T. E. (2020), 'Array programming with NumPy', *Nature* **585**(7825), 357–362.
URL: *https://doi.org/10.1038/s41586-020-2649-2*

Hashemi, M. (2019), 'Enlarging smaller images before inputting into convolutional neural network: zero-padding vs. interpolation', *Journal of Big Data* **6**(1), 1–13.

He, K., Gkioxari, G., Dollár, P. and Girshick, R. (2017), Mask r-cnn, *in* 'Proceedings of the IEEE international conference on computer vision', pp. 2961–2969.

Hochreiter, S. and Schmidhuber, J. (1997), 'Long short-term memory', *Neural computation* **9**(8), 1735–1780.

Hong, H. (2021), 'tensorflow-yolov4', `https://github.com/hhk7734/tensorflow-yolov4`.

Jaikumar, Punitha Vandaele, R. and Ojha, V. (2020), 'Transfer learning for instance segmentation of waste bottles using mask r-cnn algorithm'.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S. and Willing, C. (2016), Jupyter notebooks – a publishing format for reproducible computational workflows, *in* F. Loizides and B. Schmidt, eds, 'Positioning and Power in Academic Publishing: Players, Agents and Agendas', IOS Press, pp. 87 – 90.

Li, L.-J. and Fei-Fei, L. (2007), What, where and who? classifying events by scene and object recognition, *in* '2007 IEEE 11th international conference on computer vision', IEEE, pp. 1–8.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. and Zitnick, C. L. (2014), Microsoft coco: Common objects in context, *in* 'European conference on computer vision', Springer, pp. 740–755.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. and Berg, A. C. (2016), Ssd: Single shot multibox detector, *in* 'European conference on computer vision', Springer, pp. 21–37.

Lv, Z., Xu, J., Zheng, K., Yin, H., Zhao, P. and Zhou, X. (2018), Lc-rnn: A deep learning model for traffic speed prediction., *in* 'IJCAI', pp. 3470–3476.

Nie, X., Duan, M., Ding, H., Hu, B. and Wong, E. K. (2020), 'Attention mask r-cnn for ship detection and segmentation from remote sensing images', *IEEE Access* **8**, 9325–9334.

Pan, S. J. and Yang, Q. (2009), 'A survey on transfer learning', *IEEE Transactions on knowledge and data engineering* **22**(10), 1345–1359.

Pan, S. J. and Yang, Q. (2010), 'A survey on transfer learning', *IEEE Transactions on Knowledge and Data Engineering* **22**(10), 1345–1359.

Patrona, F., Nousi, P., Mademlis, I., Tefas, A. and Pitas, I. (2020), Visual object detection for autonomous uav cinematography, *in* 'Proceedings of the Northern Lights Deep Learning Workshop', Vol. 1, pp. 6–6.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011), 'Scikit-learn: Machine learning in Python', *Journal of Machine Learning Research* **12**, 2825–2830.

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. (2016), You only look once: Unified, real-time object detection, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 779–788.

Redmon, J. and Farhadi, A. (2017), Yolo9000: better, faster, stronger, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 7263–7271.

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

Rohit, S. (2019), 'Measuring traffic speed with deep learning object detection'. [Online; posted 16-January-2019.
URL: *https://medium.com/datadriveninvestor/measuring-traffic-speed-with-deep-learning-object-detection-efc0bb9a3c57*

Sarle, W. S. (1996), 'Stopped training and other remedies for overfitting', *Computing science and statistics* pp. 352–360.

Shih, C.-S., Huang, P.-W., Yen, E.-T. and Tsung, P.-K. (2019), Vehicle speed prediction with rnn and attention model under multiple scenarios, *in* '2019 IEEE Intelligent Transportation Systems Conference (ITSC)', IEEE, pp. 369–375.

Sumit, S. S., Watada, J., Roy, A. and Rambli, D. (2020), In object detection deep learning methods, yolo shows supremum to mask r-cnn, *in* 'Journal of Physics: Conference Series', Vol. 1529, IOP Publishing, p. 042086.

Van Herwaarden, S., Grachten, M. and De Haas, W. B. (2014), Predicting expressive dynamics in piano performances using neural networks, *in* 'Proceedings of the 15th Conference of the International Society for Music Information Retrieval (ISMIR 2014)', International Society for Music Information Retrieval, pp. 45–52.

Van Rossum, G. (2020), *The Python Library Reference, release 3.8.2*, Python Software Foundation.

Zhu, Z., Yang, B., Xu, G. and Shi, D. (1996), A real-time vision system for automatic traffic monitoring based on 2d spatio-temporal images, *in* 'Proceedings Third IEEE Workshop on Applications of Computer Vision. WACV'96', IEEE, pp. 162–167.

Appendix A

Appendix

The project code comes in two main git repositories, one contains the code for the Mask RCNN and speed prediction system and the other contains the code for the Yolov4 implementation.

Mask RCNN Speed Prediction Repository:

`https://csgitlab.reading.ac.uk/am001697/individual_project_speed_predictions.git`

Yolov4 Repository:

`https://csgitlab.reading.ac.uk/am001697/individual_project_yolov4`