Honours Project - MHW225671

Final Report

2021-2022

Department of Computing

Submitted for the Degree of: BSc Computing

Project Title: Develop and train a machine learning model for the purposes of object detection and recognition in the context of autonomous driving using semi-supervised learning elements and evaluate its accuracy

Name: Blair Nimmo
Programme: BSc Computing
Matriculation Number: S1802408
Project Supervisor: Jacob Koenig
Second Marker: Lisa Liu

Word Count: 10,071
(***excluding contents pages, figures, tables, references and Appendices***)

"Except where explicitly stated, all work in this report, including the appendices, is my own original work and has not been submitted elsewhere in fulfilment of the requirement of this or any other award."

Signed by Student: Blair Nimmo **Date: 29/04/2022**

Abstract

This report is an investigation into the development and use of machine learning models for the purposes of object detection and recognition within the context of autonomous driving vehicles as an Honours project submission for BSc Computing. The background of machine learning and its different forms have been explored as part of a literature review to gain an understanding of the problem area and the direction of the industry of computer vision. The findings from this literature review have been used to establish a problem statement for which this report will be centred on. To satisfy this problem statement a machine learning model for vehicle detection on public roads has been developed as the primary research method, with its required development technologies discussed and justified. The project objectives and implementation lifecycle are then presented and followed throughout the stages of model development from creation to training and testing. Results obtained from the testing stage and the reasons behind them are analysed, and the overall success of the project evaluated against the explored literature and the original problem statement. Potential further work for both the machine learning model developed in this project and the field of computer vision and autonomous driving is considered, and issues surrounding this type of technology are discussed.

Table of Contents

1 Introduction

This chapter will introduce the project and the primary research method selected and highlight its relevance in the context of its topic area. The project background will be examined with an appropriate problem statement formed, which will indicate the expected results from the primary research in the form of a hypothesis. The objectives of both the literature and technology review and the primary research will be identified as a foundation for the project moving forward.

1.1 Project Background

This is a report produced for submission in part fulfilment for Computing Honours. This report will detail the steps taken in developing a software artefact with the functionality of detecting and recognising objects within an image in the context of autonomous vehicles. The software artefact developed will be a machine learning model with elements of semi-supervised machine learning for training purposes. Semi-supervised learning is the branch of machine learning concerned with using labelled as well as unlabelled data to perform certain learning tasks [1] A literature and technology review has been undertaken to gain an understanding of the previous work carried out within this field, including the previous methods used to investigate different forms of machine learning for object detection and recognition within industry. This review has been used as the basis of this project; it will outline the history of semi-supervised machine learning and its relevance to this project and the software technologies selected to carry out the primary research; the development of the machine learning software artefact. The objectives for undertaking a comprehensive literature and technology review have been listed alongside the objectives for the success of the primary research, its evaluation and conclusion. The methods for conducting this project will also be discussed; a justification of the primary research method will be presented, along with its precise nature and the steps taken throughout this report to complete its development. A timeline which was followed for the implementation of the primary research has been established, leading from the literature review through the implementation strategy up to the final completed prototype and the evaluation of its success.

1.2 Project Justification

All autonomous vehicles make use of cameras to view the world around them and take in data. An autonomous vehicle needs to understand the objects it is surrounded by and recognise and identify them in real-time to be able to make decisions and drive safely.

Labelled data can be expensive, time consuming and difficult to obtain which can be a considerable hurdle in the development of object recognition software. The development of a machine learning model aims to address this problem by being trained to detect and recognise objects by taking advantage of an abundance of unlabelled data (which can easily be obtained as a vehicle is driving and recording the real world) to improve learning performance [2] If a successful object recognition solution which makes use of semi-supervised learning throughout training were to be developed by this project, its implementation in autonomous vehicles would drastically cut development timeframes and costs while improving safe driving.

1.3 Project Overview

This section aims to establish the problem statement and the primary research method that will be used to satisfy it. The objectives for both the literature and technology review and the primary research are also presented, along with the primary project aim and hypothesis.

1.3.1 Project Outline

The primary research phase of this project will involve the development and implementation of a machine learning model for detecting and recognising objects, specifically vehicles in different scenarios on public roads, within an image. The model will initially be trained on a labelled dataset and then tested in numerous iterations on identifying and recognising vehicles in subsequent unlabelled datasets.

Following on from this project outline, a problem statement has been formed:

Develop and train a machine learning model for the purposes of object detection and recognition in the context of autonomous driving using semi-supervised learning elements and evaluate its accuracy

From this problem statement a determination will also be made on whether the machine learning model developed will be effective in its prototype state for use in industry within the context of a larger system for autonomous vehicles.

1.3.2 Literature Review and Primary Research Aims and Objectives

The objectives of the literature and technology review which will be the basis for the primary research are as follows:

- Investigate the history of machine learning methods and justify the use of semi-supervised learning for training purposes

- Examine the use of the train-and-test method in relevant machine learning projects

- Examine the use of semi-supervised learning within the context of autonomous vehicles

- Present and justify the development technologies that will be used for the development and implementation of the primary research

The project objectives which will be milestones for developing the primary research are as follows:

- Develop base model which can identify separate objects within an image using bounding boxes

- Train the model on the initial labelled training dataset of objects

- Test the model on detecting and recognising objects from an unlabelled dataset

- Assign an accuracy score to the images labelled by the model

- Reiterate this process with subsequent unlabelled datasets for testing and evaluate whether the model is consistent and accurate with its labelling

- Draw a conclusion and compare it with the goals of the problem statement and determine the success of the project

1.3.3 Primary Project Aim

The primary project aim is to develop a machine learning model for detecting and recognising objects within an image, with a side goal of establishing the success of using semi-supervised learning for training purposes. Benefits of working machine learning models in the industry of computer vision are substantial; less time would need to be spent on hand-labelling objects and images and therefore development times and costs could be cut massively while more time can be focused on other development issues.

It is expected that during training, the machine learning model will become more accurate with each iteration because as more objects are successfully recognised and labelled the dataset that the model can draw from as experience will be increasingly extensive. If this hypothesis is proved correct, it will lend well to the testing stage of identifying objects within completely new, previously unseen images.

To test this hypothesis, the accuracy score of each iteration of the model training process will be examined chronologically. If the accuracy score is seen to be increasing iteration after iteration, the conclusion that the model is learning will be taken and testing can begin. If the accuracy is seen to be decreasing, the reasons behind this will be explored and rectified and the training process of the model will begin again.

2 Literature and Technology Review

This chapter contains the literature and technology review. The literature review is the groundwork for the rest of the project; it allows for the required understanding of the project area to be acquired and any previous researched carried out by others in the field to be assessed.

The following literature review will investigate further into the background of machine learning and its different learning methods and will detail why a semi-supervised learning approach for the model training has been chosen. The software technologies that have been selected for the development of the primary research will be presented with a justification for their selection and use.

2.1 Investigation of Machine Learning

The project background section briefly defined the semi-supervised machine learning method and justified its use; however, it will be insightful to examine in greater detail the other machine learning methods and their history and explore fully why the chosen learning method is the appropriate choice for the purposes of training the machine learning model in object detection and recognition.

This section will also present the different uses of semi-supervised machine learning within industry, with a focus on the application of this project in the context of autonomous vehicles if the hypothesis is satisfied.

2.1.1 The Machine Learning Methods

Machine learning has been a field of study within computer science since Alan Turing first proposed his thought experiment for machine intelligence in 1950 [3] when he published his paper, Computing Machinery and Intelligence. [4] Michael Paluszek and Stephanie Thomas write “machine learning is a field in computer science where existing data are used to predict, or respond to, future data.” [5] There are three different types of learning within machine learning: supervised, unsupervised, and semi-supervised, which all use different types of existing input and output datasets to predict this future data.

Supervised machine learning (SML) has been the standard for machine learning since the first studies in the 1950s and 1960s after Turing’s initial paper. IBM defines the supervised learning method as the use of labelled datasets to train models to classify data or predict outcomes accurately. [6] There are a variety of uses of SML in the real world, for example Google have implemented SML into their chatbots which are used by companies like DPD and Malaysia Airlines to improve the learning processes of the AI and make them more humanlike and useful in their conversations [7] an example of one of their chatbots is shown below:

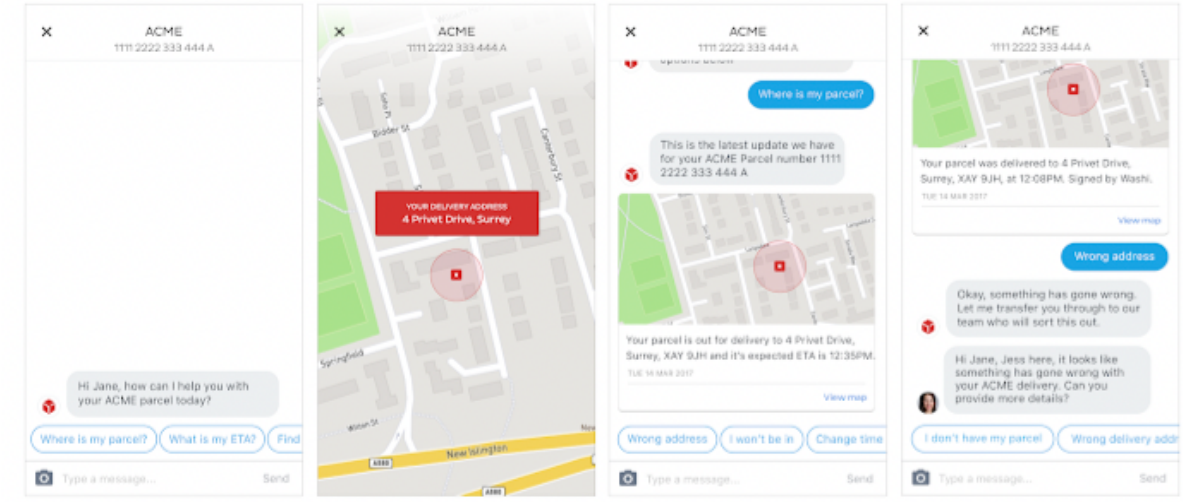


Figure 1: DPD SML Chatbot by Google [8]

Supervised machine learning requires both the input and output data to be pre-labelled; this is impractical in the context of object recognition as not all input objects will share the same characteristics, and some may be even completely foreign to the model. Furthermore, manually coded training data is expensive and labour-intensive to obtain, [9] which makes supervised machine learning a less attractive choice for the development of this project compared to other machine learning methods.

Different in method to SML, unsupervised machine learning (UML) uses machine learning model to analyse and cluster unlabelled datasets. [10] UML works on entirely unlabelled inputs, clustering them together based on their shared characteristics and putting forward these clusters into other machine learning models [11]. UML does have a variety of different useful applications, for example customer segmentation where customers are formed into groups based on purchasing/viewing similarities; however, for an object recognition solution such as this project the UML method is not effective because without a single labelled input the model would be unable to identify and draw similarities to recognise objects within the test dataset.

A previously understudied machine learning method by comparison to SML and UML, semi-supervised machine learning (SSML) is quickly becoming the foundation for object detection and recognition. Sebastian Raschka, professor of statistics at the University of Wisconsin states SSML overcomes the problem of supervised learning of "having not enough labelled data" by "adding cheap and abundant unlabelled data." [12] Evidence suggests that in numerous favourable situations in software systems the use of unlabelled data can improve the performance of machine learning models, and there have been numerous studies conducted to confirm this and gain a more in-depth understanding of SSML [13, 14, 15]. In SSML, the model is constructed using a small amount of training data with target labels and a large amount of training test data without labels [16]. This method will be highly effective for training a model to recognise objects, people, vehicles etc. by training on labelled training data and then recognising similarities in the objects presented in the unlabelled test data to identify them.

Researchers at Google and Carnegie Mellon University adopted a similar approach and proposed the use of a self-training SSML method with the goal of training a detector with a

mix of unlabelled and labelled data and evaluating its results. [17] Their project was based on facial recognition within computer vision and focused on a five-step process to detect human eyes within a photo: training the detector with labelled positive and negative examples of the object, running the detector over a weakly-labelled training image searching for a likelihood ratio, assigning a score to the object, and then selecting a subset of the newly labelled example for categorisation. The final step involved restarting at step 1 until all training images had been analysed. Two selection metrics were outlined; the confidence metric which was computed anew using the training dataset at each iteration and the MSE metric which used the training dataset initially whilst incorporating the new weakly-labelled images from previous iterations into the training process.

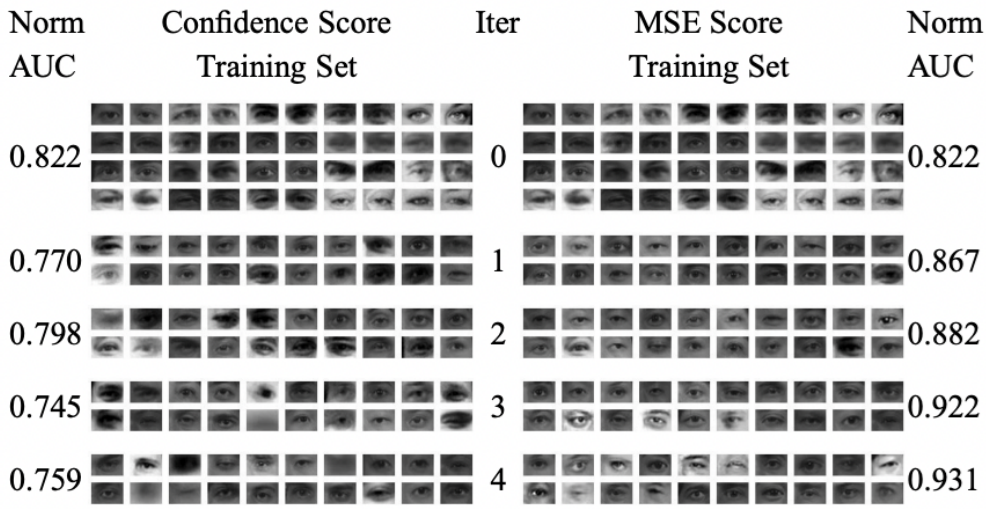


Figure 2: Accuracy of SSML Detector using different Selection Metrics after each iteration [18]

This identifies the accuracy (Norm AUC) of the detector after each iteration using the two selection metrics. It shows that using the MSE selection metric the detector became increasingly accurate with each iteration, which is important because it demonstrates that the detector was teaching itself to better detect eyes within the photo the more times it ran due to it having more experience.

They found that this process of training a detector to search for similarities in training images was effective and fits well with the SSML method for object detection and recognition when using the MSE selection metric.

We can conclude that semi-supervised machine learning will be the most appropriate learning method for the development of the model. Acquiring enough labelled examples for supervised learning is unfeasible due to the cost and time required and at the same time it is unrealistic to expect a moving vehicle to already have every object it will encounter identified and labelled; unsupervised learning would work by clustering objects into groups based on similarities which while useful would not identify individual objects and label them based on similarities with other objects drawn from previous experience, which is the goal of this model.

2.1.2 Machine Learning in Autonomous vehicles

Machine learning is becoming one of the most prevalent fields in the development and manufacturing of autonomous vehicles [19, 20, 21, 22] in recent years. There are many different applications of machine learning for the vehicles, such as risk prediction for safe driving, driver distraction detection and even developing testing frameworks to ensure the machine learning models themselves are functioning appropriately.

As semi-supervised machine learning is researched further its benefits in autonomous vehicles are becoming clearer. Using SSML new approaches and systems are being developed for autonomous vehicles, for example analysing lighting conditions and other adverse effects [23], fatigue driving detection [24] and vision-based distance measurement [25].

This demonstrates that the autonomous vehicle industry is making increased use of different forms of machine learning models to add new functionality to their cars, making them safer, more efficient and improving user experience. This will likely remain the trending direction of software development in this area which makes working object recognition software using SSML highly relevant.

2.2 Development Technologies

This section of the literature review will present the software development technologies selected for the primary research section of the project and justify their use. It will be beneficial for the success of the primary research to have a clear sense of the purpose of each of the technologies and their respective roles within the development phase.

2.2.1 TensorFlow

TensorFlow is an end-to-end open-source platform for developing machine learning models created by Google. It has been selected as a primary software for developing the machine learning model that will be trained using the datasets to recognise and label unknown objects. TensorFlow has significant benefits to its use; it is free to use, has a robust debugging system, takes a graphical approach to visualising data as images and is compatible with Python and Jupyter notebooks, two other development technologies selected and used for this project. An alternative to TensorFlow would be OpenCV; however, OpenCV is used for this project for other development purposes, whereas TensorFlow is primarily used for developing the machine learning model [26] and it will continue to integrate smoothly with the other development technologies presented.

2.2.2 OpenCV

OpenCV is an open-source computer vision and machine learning library [27] which will be imported for this project to make use of its relevant libraries and learning algorithms for implementing and training the primary research model for object detection and recognition. The primary learning algorithm that will be implemented from OpenCV is the selective search segmentation algorithm, which is used to identify and localize objects within an image. This algorithm and its history is expanded upon further during the Model Implementation section where it is first loaded into the Jupyter notebook. With more than

2000 unique algorithms used for the purposes of computer vision, OpenCV will be a vital development technology for the Execution and Development stage of this project.

2.2.3 Keras

Keras is another development technology that has been selected for the development of the primary research method. Keras is an API that provides essential deep learning frameworks making use of Python [28] and TensorFlow, two other essential development technologies that have been previously selected and justified. Keras is also used frequently with Kaggle, the development technology that has been chosen to provide the image datasets that will be used for the training and testing stages of this project. During the Model Implementation and Model Training stages, Keras will be used to employ Convolutional Neural Networks such as VGG16 for the purposes of training the model on object detection and recognition algorithms. Sequential APIs are also used from Keras to group the model layers and their inputs and outputs in a linear fashion through VGG16.

2.2.4 Python

Python was chosen as the development language to carry out the development of the project because it is a highly readable and concise language. Compared to other languages that could have been used, like Java, JavaScript or C, Python already has an extensive selection of libraries and frameworks ready to use that will be advantageous in developing the machine learning model. Python programs in general are also a lot quicker to develop than other languages, which will allow more focus to go onto the training, testing and evaluation phases of the project where the model will independently label objects [29]. Python is also the language used with Jupyter notebooks, which is another development technology that has been selected for this project so the use of Python as the primary coding language makes sense.

2.2.5 Jupyter Notebooks

Jupyter notebooks is a key development technology that was selected as the primary development environment for constructing and implementing the machine learning model. Jupyter notebooks provides an interactive environment for developing applications using Python – the chosen development language for this project – and can integrate the different libraries Python possesses to develop the model, visualise and eventually analyse the results gained from the test data. Jupyter notebooks are also able to seamlessly import another key development technology, TensorFlow, as a Python library. An alternative to Jupyter notebooks which was originally envisioned for this project was Google Colab, however once the Execution and Development stage commenced it became clear Google Colab was not compatible with the hardware being used for the project, and Jupyter notebooks provided a similar development environment which was more interactive and seamless to use. The decision to change development environment from Google Colab does mean that the access to using powerful external GPUs over the internet is no longer possible, however this was only an inconvenience as on-hand hardware would suffice, however it would take longer to train the model.

2.2.6 Kaggle Datasets

The technology selected for finding an appropriate dataset to use for training and testing the machine learning model is Kaggle, which is an online community hub for users to share, download and discuss datasets. With over 50,000 online datasets, finding a suitable option for the purposes of this project was straightforward, and a dataset containing approximately 1000 training images and a separate 175 testing images of vehicles [30] was chosen. This number of images is sufficient for this project, and perhaps somewhat bloated in terms of the training dataset which will be addressed during the Model Implementation stage.

2.2.7 GitHub

GitHub is an open-source version control system [31] that allows multiple different versions of the model throughout the development stage to be securely stored online. GitHub can be directly linked to other development technologies like TensorFlow and Jupyter notebooks which will make saving and accessing multiple iterations of the model development quick and seamless. Having access to multiple iterations and versions of the model will make development easier and allow an evolving view of the model at its different stages which can each be used at any time.

3 Method

This chapter will present the research method, the preparatory work to be taken before the Execution and Development stage commences and the steps taken to conclude the development of the machine learning model prototype. This will include the project lifecycle and the remaining development stages and plan. The evaluation process will also be described along with the final conclusion.

3.1 Research Method

The purpose of this project is to design and implement a machine learning model for the purposes of detecting and recognising vehicles within an image. To carry out the development of the prototype the develop and test research methodology will be employed which Oates outlined in his 2005 book “Researching Information Systems and Computing” [32].

This methodology involves developing a project from scratch before testing it against pre-defined objectives with the goal of satisfying the original problem statement.

The problem statement has been established at the beginning of this report followed by the literature review; now the development methods will be outlined and then the prototype developed which will be tested and have these testing results evaluated to determine the success of this project.

3.2 Development of the Machine Learning Model

This section details the project lifecycle, as well as the preparatory work to be conducted for the project before the Execution and Development stage commences, the development stages and plan for the full model prototype.

3.2.1 Project Lifecycle

Defining the project lifecycle is a key moment in the development of a project. The model will be developed using an agile development methodology based on an iterative process, which will allow for numerous iterations throughout the development of the base model. Once the model has been developed, it will be trained on a training set of labelled images and the iterative process will be used again to conduct testing on multiple datasets of unlabelled images. The agile and iterative development methodologies have several benefits over other lifecycles such as the waterfall methodology: the use of agile and iterative development embraces change, allows the project to work with an unknown end result and allows for continuous improvement throughout development [33].

The figure below is a Gantt chart that was developed to outline the project lifecycle of the Execution and Development stage of this project. The Gannt chart is laid out in weeks from the start of development of the machine learning model, which was in December 2021 at the end of the first trimester of the Honours year. The different sections of the chart detail the stages of development which are themselves comprised of the more detailed stages presented in the Development Stages section to follow.

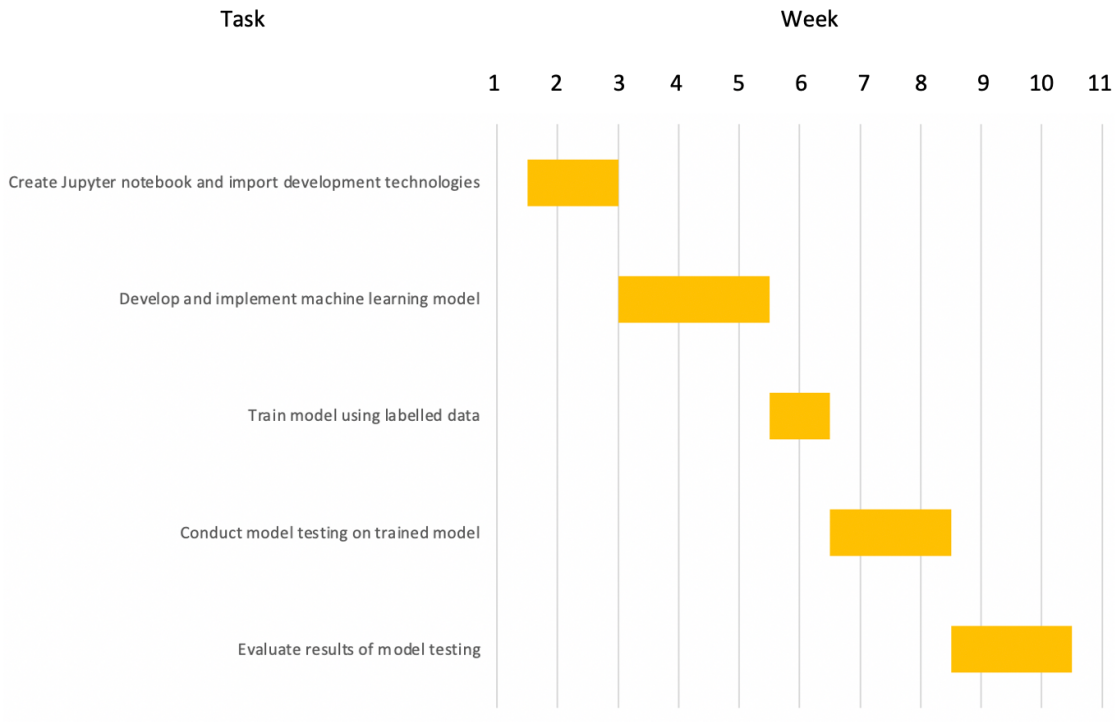


Figure 3: Project Lifecycle Gantt chart

3.2.2 Preparatory Work

To prepare for the development stage of the machine learning model, the technologies that will be used have been identified with their respective uses justified and their integration with each other described. From this section the project lifecycle and development stages have also been presented.

An initial test development has been conducted from examples within TensorFlow to become familiar with the software and the practices used to develop a machine learning model. This test development has provided the background required to proceed with the full creation of a working prototype and is a solid foundation from which to continue development.

3.2.3 Development Stages

The development stages of the primary research are as follows:

- Create Jupyter notebook, import TensorFlow and other software components and link together
- Develop model using iterative development cycle
- Import training and testing datasets
- Conduct training of model using the labelled training dataset
- Conduct first testing iteration of model on initial unlabelled testing dataset
- Conduct second testing iteration of model on random unlabelled testing dataset
- Record accuracy score
- Stop after 5-10 testing iterations

3.2.4 Development Outline

Following the literature and technology review, the machine learning model will be developed following the previously defined development stages. This is expected to take approximately four weeks for completion of the implementation of the model, with the training phase and the multiple iterations of the testing phase taking an additional three weeks. The analysis and evaluation of the results acquired from the testing phase are expected to take roughly two weeks, leaving the entire Execution and Development stage at nine weeks total. This target kept the project development moving forward while also leaving time for development errors.

Once the development outline provided in this section has been followed, a conclusion can be made on whether the problem statement was satisfied. This report was produced both throughout and after the Execution and Development stage.

3.3 Evaluation

The evaluation will be carried out following the results of the testing stage of the machine learning model, by taking the accuracy scores of each iteration of the testing phase and determining whether the model is able to detect, recognise and label unlabelled images accurately and consistently.

If the machine learning model implemented is able to perform its intended function and the training stage is found to be increasing the accuracy of the model, this will be a positive sign for the outcome of the project and lead into the conclusion. However, if the model is found to either be decreasingly accurate at labelling images or unable to perform its intended function of detecting and recognising unlabelled objects accurately and consistently, this will be a

negative sign for the outcome of the project and considerations will need to be taken to address this in the conclusion of this report.

3.4 Analyse Score Results and Determine Conclusion

From the evaluation of the Execution and Development stage and the testing results, a conclusion on the success of the machine learning model and this project as a whole will be made.

If the model can perform its intended function, the project will be deemed a success. Further conclusions will be made on the success of implementing a semi-supervised machine learning method for the purposes of testing, and whether another learning method could have been employed at this stage.

However, if the model is found to not be able to perform its intended function, this project will be deemed a failure and it will be necessary to conduct an investigation into the reasons for this failure and determine whether improvements can be made to the model. Any reasons for this failure will be presented alongside a reasonable plan and timescale for rectifying these reasons.

4 Execution and Development

This chapter of the report will present the execution and development stage of the machine learning model for the project, beginning with the project specification. The design and implementation of the model will be described, followed by the training method and model testing and results. This chapter will also describe the decisions made during implementation and training of the model and the reasoning behind each decision made.

4.1 Project Specification

The project specification will give a general summary of the goals during development that were established in the Methods section and link the implementation steps back to the problem statement and its goal.

The aim of this project is to develop a machine learning model to satisfy the problem statement and determine whether the model will maintain a consistent accuracy score at identifying and labelling unlabelled objects (namely other vehicles) in images as it learns from experience. To meet this aim, the machine learning model will be created using the develop and test methodology and implemented in Jupyter notebook, making use of the dataset of images previously selected in the Development Technologies section from the website Kaggle. This dataset will provide the library of training and testing images that the model will make use of.

To determine whether the problem statement is satisfied, the model will be implemented then trained on the Kaggle dataset using relevant Python libraries and machine learning algorithms. Following training, two sets of images from the Kaggle dataset and images acquired from Google will be used to test the accuracy of the model. The images dataset for testing taken from Kaggle will provide a baseline for testing, and a second dataset will be formed from images taken from Google for the purposes of testing the accuracy of the model on images entirely foreign to the model and original dataset. This dataset will be detailed further in the Model Design section. Once testing is complete, the accuracy score of each of the images from both datasets will be taken and presented in a table. An evaluation of the testing results will take place to determine a conclusion on the success of the project in relation to the original problem statement.

4.2 Model Design and Implementation

This section will layout the design of the machine learning model using the chosen technologies outlined in the Development Technologies section and based on the previously outlined Project Specification. The implementation stage of the project will follow, along with a presentation of the training and testing stages with their results.

4.2.1 Model Design

This subsection will describe the design of the model and the techniques that will be used during implementation.

Data

A dataset containing training and testing images of vehicles has been selected using Kaggle, which will be used for the development and implementation of the machine learning model. This dataset also consists of a .csv file, which contains the parameters of bounding boxes relevant to images from the training images set (with values being xmin, ymin, xmax, ymax) which will be used during the training stage of development to draw these bounding boxes around the vehicles within the images from the training set.

The data from the training dataset will be processed prior to the training stage, using the train_test_split method from scikit-learn to separate the training images dataset into training and testing subsets, with the testing set having a size of 33% of the initial training set. Splitting the original dataset is appropriate for this project as it is sufficiently large for the purposes of training the model and helps to avoid overfitting the model during the training stage and making it overly optimistic. This will also address the issue of the size of the training dataset being bloated.

One final testing dataset with 5 testing images was manually formed by selecting unrelated images containing vehicles from Google for the purposes of a final testing run for the trained model. This will ensure the success of the testing stage as it will provide a small set of images completely foreign to the original dataset taken from Kaggle, putting the machine learning model in as close to a real-world scenario as possible. This dataset does not require to be processed prior to testing due to its small size. This dataset will be referred to as the Google dataset throughout the remainder of this report.

Libraries and Learning Models

The machine learning model developed for this project will implement several Python libraries and employ different deep learning models to create the model and satisfy the problem statement. These will each be discussed in detail as they become relevant during the Model Implementation subsection, but include:

- TensorFlow
- OpenCV
- Selective Search Segmentation using OpenCV
- Intersection Over Union for refining bounding boxes
- Train_test_split for separating the training dataset into training and testing subsets
- VGG16 using Keras for object classification and detection

4.2.2 Model Implementation

This subsection will detail the approach taken within Jupyter notebook to implement the machine learning model based on the project specification and model design outlined in the previous sections.

The project Jupyter notebook was created and placed into a local folder alongside the relevant training and testing datasets from Kaggle and the .csv file containing the list of solution bounding boxes for each training image. The relevant Python libraries, primarily including TensorFlow and OpenCV (among others) were then imported to the notebook, and the os.listdir command was run on both datasets in the notebook to ensure they could reliably be found within the parent folder.

The .csv file used for training purposes referenced in the Model Design subsection was then read into the notebook inside a variable called Data, and the first 5 entries displayed within the notebook for clarification:

```
In [5]: Data.head()
```

Out[5]:

	image	xmin	ymin	xmax	ymax
0	vid_4_1000.jpg	281.259045	187.035071	327.727931	223.225547
1	vid_4_10000.jpg	15.163531	187.035071	120.329957	236.430180
2	vid_4_10040.jpg	239.192475	176.764801	361.968162	236.430180
3	vid_4_10020.jpg	496.483358	172.363256	630.020260	231.539575
4	vid_4_10060.jpg	16.630970	186.546010	132.558611	238.386422

Figure 4: Initial entries of train_solution_bounding_boxes.csv

The figure above shows the beginning of the Data variable (.csv file), where the images listed in the first column are images from the training dataset, with the following columns listing the parameters for the bounding boxes around each of the vehicles in the images. The length of the Data variable was read to be 559, which when checked with the .csv file in Microsoft Excel is confirmed to be the correct number of entries used for training later in development.

Three images from the training dataset were then displayed with their related bounding box parameters to confirm the notebook can find and display the images from the dataset in the notebook and connect them with their associated bounding box parameters, as shown below:

```
Photo shape: (380, 676, 3)
Name,xmin,ymin,xmax,ymax: ['vid_4_1000.jpg' 281.2590449 187.0350708 327.7279305 223.225547]
```

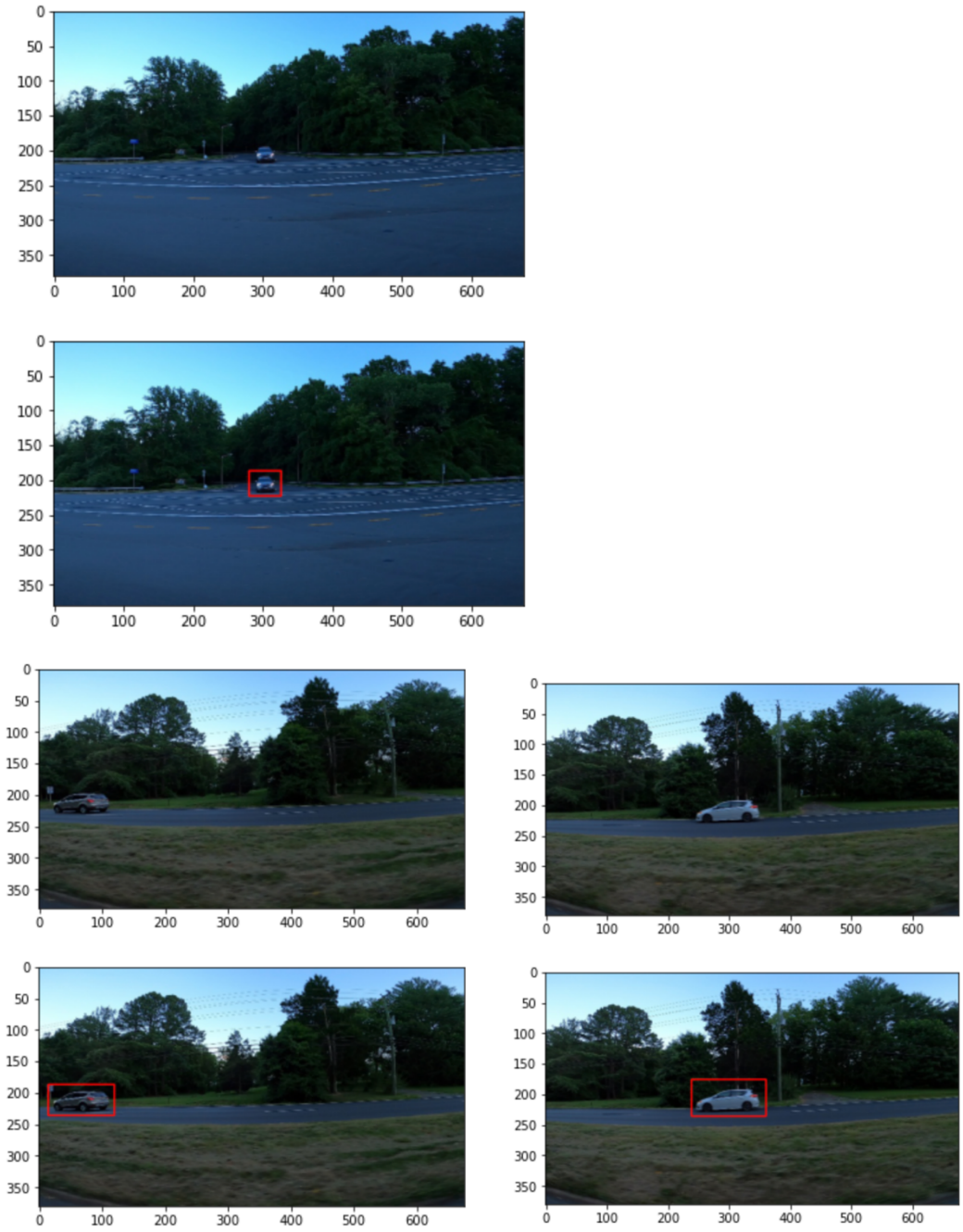


Figure 5: Examples of images before and after bounding box

The figure above demonstrates the coding of the bounding boxes around the vehicles within each image, all taken from the training images dataset and the train_solution_bounding boxes .csv file. The output at the top of the figure pertains to the first of the three images.

Each figure shows the original image before adding the bounding box, and then the images with the bounding boxes trained onto them.

To address the issue of first localizing where in an image objects are located, the software algorithm that would perform this function was required to be selected. The decision was made to implement the selective search segmentation algorithm from the OpenCV library. Selective search is an algorithm implemented in OpenCV and first introduced in the 2012 paper Selective Search for Object Recognition [32] by J.R.R Uijilings, K.E.A van de Sande et al. Selective search was created as an improvement to using sliding windows over image pyramids as a method of localizing where an object is located within an image. Using sliding windows as a method for determining where an object is located within an image for this project is possible but is comparatively slow and much more computationally exhaustive than the selective search segmentation algorithm using OpenCV. There is also a history of inaccuracy when localizing an object within an image using sliding windows and image pyramids, both through false positives and the method failing to identify and localize objects entirely. It is for these reasons that the selective search algorithm was chosen to be used for the implementation of the model over the alternatives.

An image was then chosen from the training images dataset for testing the selective search segmentation algorithm at this stage, and all the possible bounty boxes based on where the objects may be located within the image were imposed on top of the image for display:

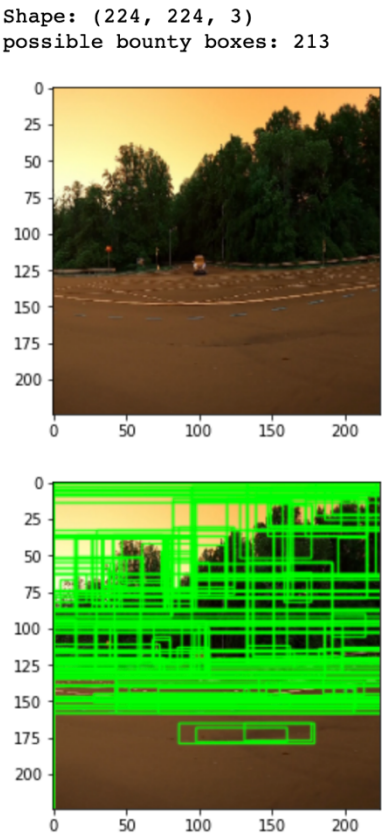


Figure 6: Example of bounding boxes surrounding all detected objects prior to segmentation

The figure above demonstrates all the possible bounding boxes that could be where the objects are localized within the image, using the selective search segmentation algorithm

from OpenCV prior to training the machine learning model to identify the vehicle as the object to search for.

To narrow down the number of bounding that could be the vehicle the model is searching for, the Intersection Over Union (IOU) evaluation metric was implemented. IOU is an evaluation metric that was employed for the implementation of the machine learning model to ensure the predicted bounding boxes that are applied onto an image match as closely as possible to the true location of the vehicle within the image. IOU works by acquiring a ratio between the area of overlap – the area where two or more bounding boxes overlap – and the area of union – the total area covered by two or more overlapping bounding boxes. Dividing the area of overlap by the area of union will give the intersection over union evaluation metric that will assist in finding the correct area of localization of the vehicle within an image, using the data from all the bounding boxes. IOU will be used in cooperation with the selective search segmentation algorithm used in the previous step to achieve this.

After reading in all the relevant data to the Jupyter notebook and implementing the algorithms to find the most accurate bounding boxes, the remaining images from the training dataset are loaded into the notebook alongside their associated parameters for the accurate bounding boxes from the .csv file:

```
vid_4_9620.jpg 541 1644
vid_4_9720.jpg 542 1623
vid_4_9760.jpg 543 1537
vid_4_9760.jpg 544 1537
vid_4_9760.jpg 545 1537
vid_4_9740.jpg 546 1599
vid_4_9700.jpg 547 1630
vid_4_9780.jpg 548 1477
vid_4_9780.jpg 549 1477
vid_4_980.jpg 550 1099
vid_4_9800.jpg 551 1513
vid_4_9800.jpg 552 1513
vid_4_9820.jpg 553 1386
vid_4_9840.jpg 554 1565
vid_4_9860.jpg 555 1497
vid_4_9880.jpg 556 1450
vid_4_9900.jpg 557 1499
vid_4_9960.jpg 558 1380
vid_4_9980.jpg 559 1379
```

Figure 7: Training images being loaded and matched to relevant bounding box parameters

The figure above shows the last of the 559 training images that have accurate parameters for the bounding boxes being loaded into the notebook. These images and their associated bounding boxes will be used during the training stage of the machine learning model.

The dataset containing training images has been split into a smaller training set and a testing set for validation. This was implemented using the train_test_split method from scikit-learn, which is a technique used for evaluating the performance of a machine learning algorithm or model, by splitting one dataset into two distinct data subsets for different purposes. The first is a subset used to fit the model and in this report is referred to as the training data subset, and the second is used to evaluate the model fit during the testing stage of development and is referred to as the testing subset throughout the remainder of this report. The benefit of splitting the original dataset into these two subsets allows for modelling the effectiveness of the machine learning model in a realistic scenario, by allow it to fit or train on data (training subset) with known results (the parameters from the train_solution_bounding_boxes.csv file) and then test on new examples (testing subset) with expected but not known results that will simulate the real world. The results from testing on the testing subset will then give an accurate depiction of the effectiveness of the model. The use of the Google dataset described in the Model Design section will be

One final learning model must be included in the Jupyter notebook before the model can be trained. VGG16 is implemented into the notebook using TensorFlow and Keras, which is a Convolutional Neural Network (CNN) architecture used in computer vision and is one of the most popular CNNs used for the purposes of object detection, achieving approximately a 93% accuracy score on average. The VGG16 (Visual Geometry Group) architecture has 16 weight layers, reference by the 16 in its title. 13 of these layers are convolutional layers, and the remaining 3 are the fully connected layers. The input is passed through these 16 layers, and output to the output layer as shown below:

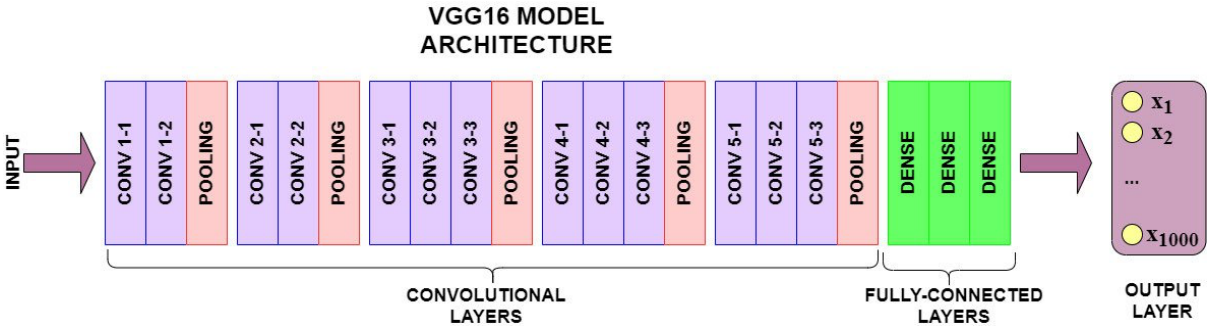


Figure 8: VGG16 model architecture

For this project, VGG16 takes image input shape as 244 x 244 with 3 RGB channels. This is set to the base_model, and the base_model summary is shown below:

```
In [23]: base_model=tf.keras.applications.VGG16(include_top=False,input_shape=(224,224,3),weights='imagenet')

In [24]: base_model.summary()
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

Figure 9: VGG16 architecture within Jupyter notebook

The figure above presents the output of the base_model summary displayed in the Jupyter notebook, which follows the expected VGG16 architecture shown in figure 8.

Following this implementation stage, the model was fully configured and ready to begin the training stage. Training will make use of all the libraries and scripts implemented into the notebook and model up to this point and will be done over several iterations.

4.3 Model Training

This section will describe the process of training the model to fit and reliably detect and recognise vehicles in new scenarios, implemented in the Jupyter notebook using the pre-labelled datasets of images.

The training dataset of images originally obtained from Kaggle was split in the Model Implementation stage into two subsets: one for the purposes of training and one for the purposes of testing. Before training can commence, the decision has been made to set the model to make use of the sequential API from Keras, which groups the model into linear layers to be created one at a time. The sequential API is simpler than its alternative, the functional API, but for the purposes of this project it will suffice as the model only has one given input and output for each image.

The model will be trained over 8 iterations of epochs. An epoch is a term in machine learning which describes the number of passes the machine learning model makes over the entire training dataset, which for this project is the training subset created when the train_test_split method was run. The figure below presents the training stage over the 8 epochs, showing the loss and accuracy and validation loss and validation accuracy for each.

```
In [30]: epoch=8
         hist=model.fit(x_train,y_train,epochs=epoch,validation_data=(x_val,y_val))

         Epoch 1/8
         222/222 [==============================] - 1266s 6s/step - loss: 0.7614 - accuracy: 0.7952 - val_loss: 0.1581 - val_a
         ccuracy: 0.9573
         Epoch 2/8
         222/222 [==============================] - 1227s 6s/step - loss: 0.2587 - accuracy: 0.9289 - val_loss: 0.1239 - val_a
         ccuracy: 0.9656
         Epoch 3/8
         222/222 [==============================] - 1226s 6s/step - loss: 0.2094 - accuracy: 0.9405 - val_loss: 0.1124 - val_a
         ccuracy: 0.9702
         Epoch 4/8
         222/222 [==============================] - 1231s 6s/step - loss: 0.1945 - accuracy: 0.9457 - val_loss: 0.1027 - val_a
         ccuracy: 0.9728
         Epoch 5/8
         222/222 [==============================] - 1226s 6s/step - loss: 0.1605 - accuracy: 0.9516 - val_loss: 0.0976 - val_a
         ccuracy: 0.9731
         Epoch 6/8
         222/222 [==============================] - 1224s 6s/step - loss: 0.1573 - accuracy: 0.9567 - val_loss: 0.0899 - val_a
         ccuracy: 0.9739
         Epoch 7/8
         222/222 [==============================] - 1223s 6s/step - loss: 0.1542 - accuracy: 0.9587 - val_loss: 0.0969 - val_a
         ccuracy: 0.9759
         Epoch 8/8
         222/222 [==============================] - 1233s 6s/step - loss: 0.1401 - accuracy: 0.9584 - val_loss: 0.0867 - val_a
         ccuracy: 0.9774
```

Figure 10: Model training process over 8 epochs and validation accuracy scores

As the figure above demonstrates, the model took a significant amount of time to train during each epoch. This is a drawback from changing the development software to implement this project using Jupyter notebook over Google Colab due to hardware issues, as there was no access to the external GPUs offered by Colab. However, this proved to only be a minor inconvenience at this stage of the project development and not a fundamental issue.

It can be seen from the training of the model that the validation loss decreases as the validation accuracy increases after each consecutive epoch. Although this was an expected result from the training stage, it points to the successful training of the machine learning model and shows that the model is getting increasingly accurate as it is shown how to label a larger number of images from the training subset.

The final validation accuracy achieved from the training stage was 97%, which is a successful accuracy score and better than originally expected for the model given the use of the VGG16 CNN, which had an approximate accuracy score of 93%.

With the machine learning model now fully implemented and trained over the training images subset, testing could now commence on both the testing subset and additional testing set with images from Google to fit the model to a scenario using unlabelled images.

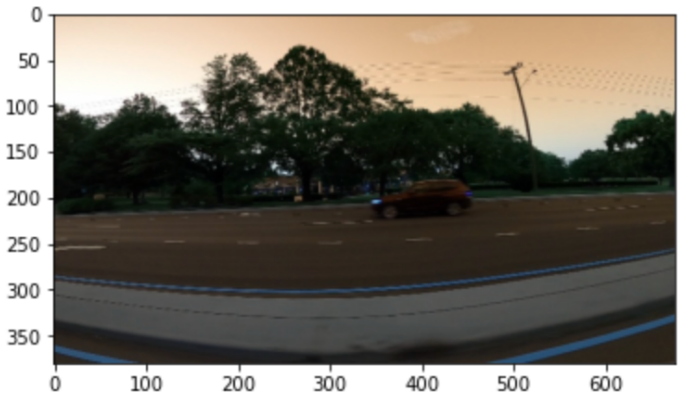
4.4 Model Testing

This section will describe the process of testing the model implemented in the Jupyter notebook on the two datasets of unlabelled images and the results acquired from testing.

Testing was conducted on the now trained machine learning model using the two testing datasets defined in the Model Design section. These two datasets are the testing subset acquired from using train_test_split on the original training dataset, and the secondary testing set formed by manually selecting unrelated images containing vehicles from Google. Each of these phases of testing will occur over 5 iterations within each of the datasets and will be used to evaluate the accuracy of the model as it encounters new previously unseen images and situations. At each iteration, the original image will be displayed with the total number of objects detected within the image presented and then narrowed down to include only objects with a sufficient class prediction rating. Once the vehicle has been detected and localized within the image, a solution bounding box will be drawn around it which will be presented in a separate image, and the accuracy score showing the confidence that the object found is indeed the vehicle displayed.

The first phase of testing will be done using the testing subset, with 5 images being selected at random and running the model over them:

```
Number of possible objects detected:  1801
Number of boxes with a class prediction of 1: 26
------------------------------------------------------------------------
```



Class number = 1 and bounding box shown has the highest probability.: %97.66657948493958

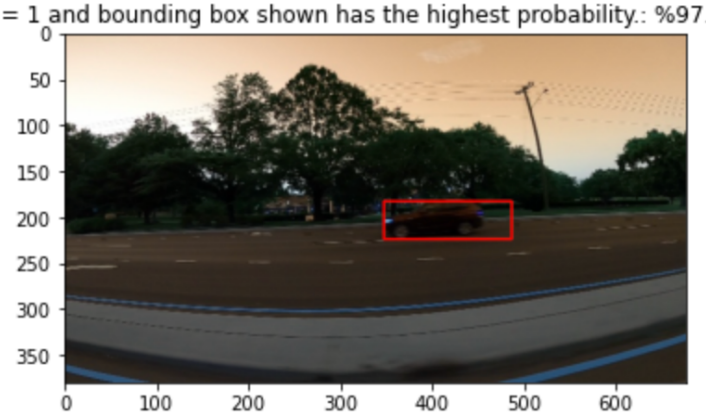


Figure 11: First iteration of testing subset phase

The figure above displays the first image used from the testing subset and presents the total number of possible objects detected and the object with the highest probability of being the vehicle within the image, surrounded by the bounding box with an accuracy prediction score of 97.66% for this image.

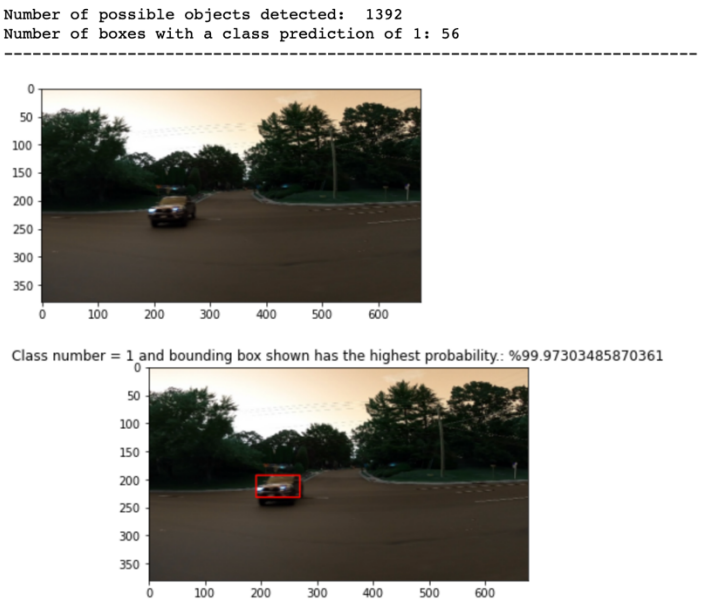


Figure 12: Testing subset iteration 2

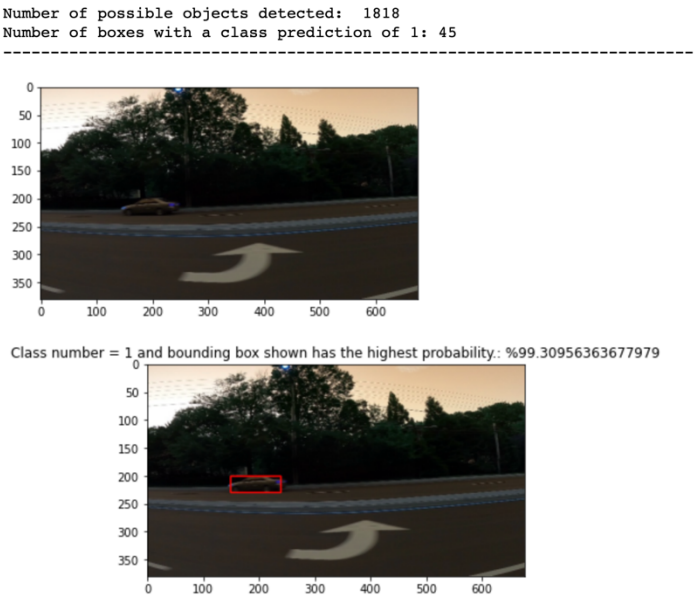


Figure 13: Testing subset iteration 3

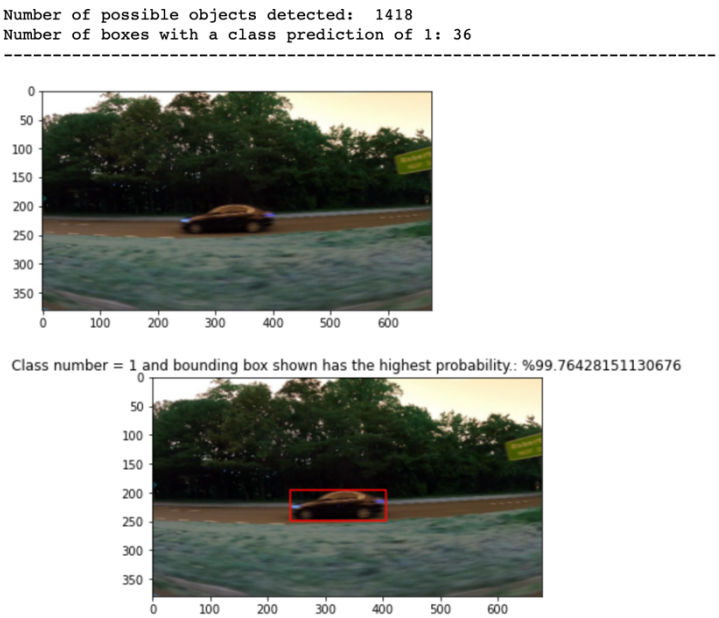


Figure 14: Testing subset iteration 4

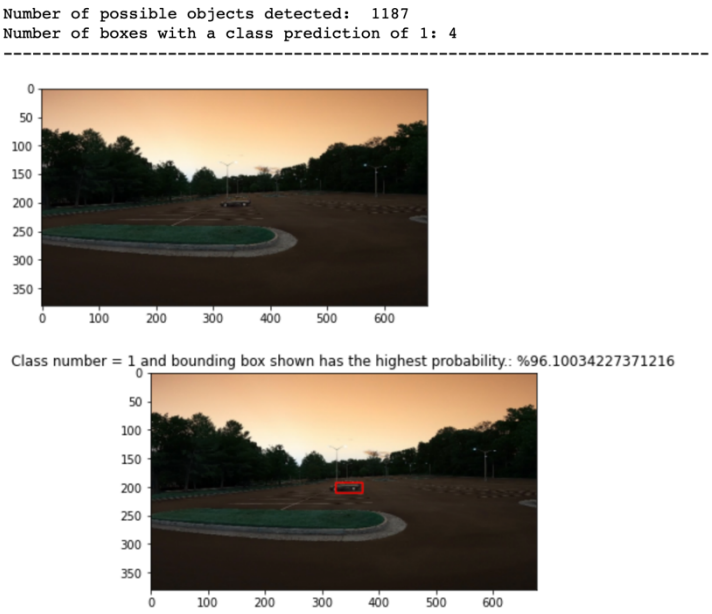
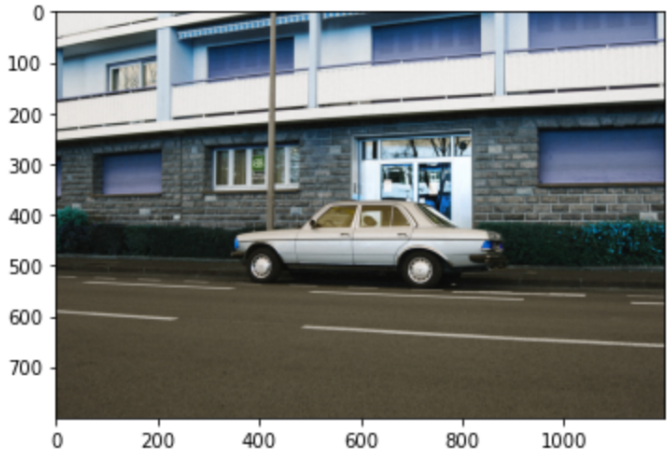


Figure 15: Testing subset iteration 5

The figures above present the remaining images from this phase of testing using the testing subset, with their bounding boxes and accuracy scores. As can be seen from the images, the model was able to successfully localize and identify each vehicle within each of the images.

The model was then tested on the dataset of 5 images found randomly on Google images, to ensure it would still perform its intended function on images it had no previous connection with:

```
Number of possible objects detected:  6896
Number of boxes with a class prediction of 1: 393
--------------------------------------------------------------------------
```



Class number = 1 and bounding box shown has the highest probability.: %100.0

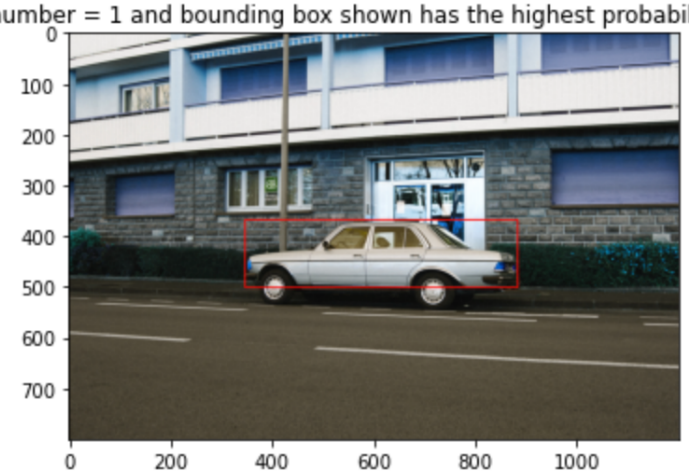


Figure 16: First iteration of Google dataset testing phase

The figure above displays the first of the five images from the Google dataset and presents the number of objects detected within the image and the object detected with the highest probability of being the vehicle within the image, surrounded by the bounding box with an accuracy score of 100% for this image.

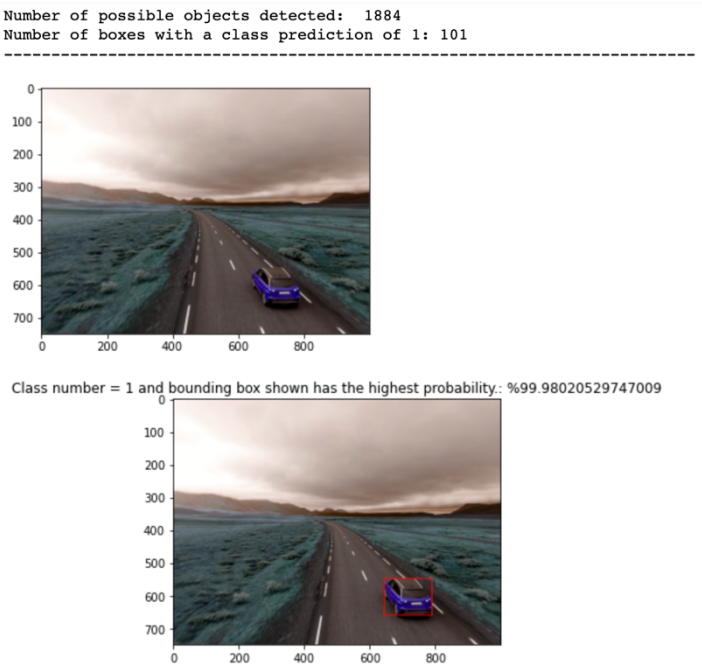


Figure 17: Google dataset iteration 2

Figure 18: Google dataset iteration 3

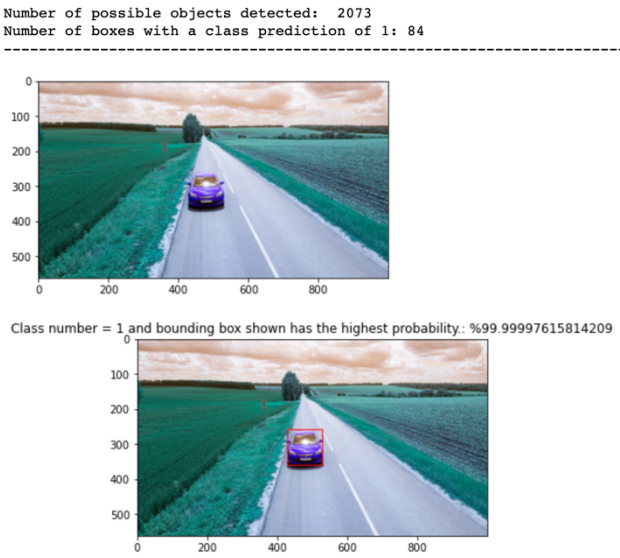


Figure 19: Google dataset iteration 4

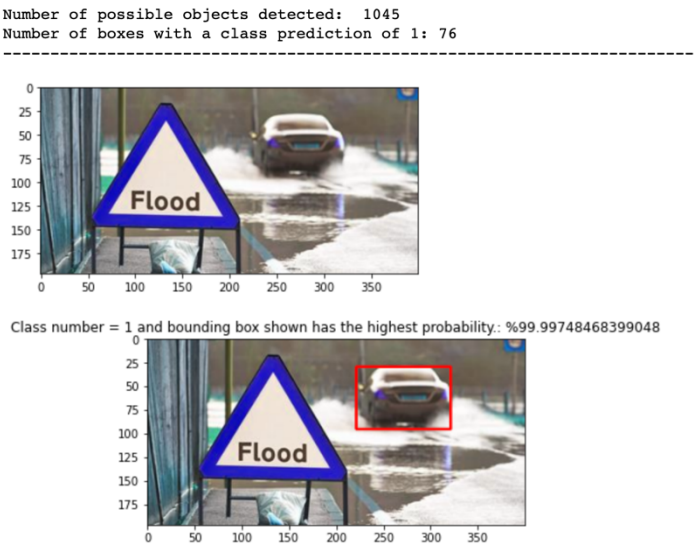


Figure 20: Google dataset iteration 5

The figures above present the remaining images from the Google dataset used to test the machine learning model, with their bounding boxes and accuracy scores. As can be seen from this phase of testing, the model is able to accurately localized and identify each vehicle from within each of the images to the same degree as with the testing subset, which is encouraging as it suggests the model is effective at performing its function on images entirely foreign to the original dataset.

From the results of testing the machine learning model over both phases using the two datasets, the following table was produced showing each testing image name and its accuracy score:

Dataset	**Image Name**	**Objected Detected Y/N**	**Accuracy Score**
Testing subset	vid_5_27620.jpg	Y	97.66%
“	vid_5_29820.jpg	Y	99.97%
“	vid_5_29020.jpg	Y	99.30%
“	vid_5_26620.jpg	Y	99.76%
“	vid_5_420.jpg’	Y	96.10%
		Average Accuracy:	98.55%
Google dataset	test1.jpeg	Y	100.00%
“	test2.jpeg	Y	99.98%
“	test3.jpeg	Y	99.24%
“	test4.jpeg	Y	99.99%
“	test5.jpeg	Y	99.97%
		Average Accuracy:	99.83%

Table 1: Testing accuracy scores over both testing phases

As can be seen from the above table, the vehicle within each image was able to be detected and localized by the model in both testing datasets. The first iteration of testing using the testing subset concluded with an average accuracy score of 98.55%, and the second iteration of testing using the Google dataset concluded with an average accuracy score of 99.83%. Although the second iteration of testing had a slightly higher average accuracy score, this could be down to a few factors:

- The images in the Google dataset are of a higher quality on average than the images in the testing subset
- The vehicles within the images in the testing subset are somewhat further away from the camera than the vehicles in the Google dataset
- General sways in accuracy scores for other minute factors which will be the case in any image or different situation

Both average accuracy scores are sufficiently high and the vehicles in all of the images have been detected, correctly localized, and recognised. The results from this testing stage will be fully evaluated in the next chapter with a conclusion drawn whether the problem statement has been satisfied and on the success of the machine learning model as a whole.

5 Evaluation and Conclusion

This chapter will present the evaluation of the success of developing the machine learning model based on the results from testing the model in the previous chapter. A conclusion will then be made on whether the original problem statement has been satisfied by its development, followed by a section detailing the potential further work that could be taken to advance the model in the future.

5.1 Evaluation of Testing Results

This section will analyse the results of testing from the Model Testing section and evaluate the meaning behind the results while also reference back to the introduction and project background.

Using the results of the data acquired during the testing phase which was presented in Table 1 in the previous chapter, it can be seen that the machine learning model was able to label each of the images from both phases of testing accurately. This accuracy scores higher than what was originally expected from the use of the VGG16 Convolutional Neural Network at 93%, which was discussed and implemented during the Model Implementation section, which is a result of training.

The model was trained over several stages of epochs using elements of semi-supervised learning, which enabled the accuracy score of the model labelling to increase with each training iteration as it learned from a larger pool of experience starting from the expected 93% of VGG16. From figure 10 in the Model Training section, we can see that the final validation accuracy score at the end of training was 97.74%, which is approximately where the expected accuracy scores should hover during testing. However, it should be noted than at two iterations during the first testing phase using the testing subset the accuracy score of the model dropped below this final validation accuracy score. This could be due to the quality of those individual testing images or just considered a discrepancy in the model, as even with these lower-than-expected accuracy scores the model still ends both phases of testing with an average accuracy score above the final validation accuracy.

It is clear that the machine learning model developed throughout the execution stage is able to make use of the different Python libraries and scripts and deep learning methods that were decided upon for implementation to perform its function of localizing the objects within the image, recognising it as a vehicle and drawing a solution bounding box around it for clarity.

The images from both phases used for testing display vehicles from different angles including the front, side and back and all at varying distances from the camera. This tests the effectiveness of the model in different scenarios and ensures the model can still perform its function in a variety of situations and not just when being tested on similar types of images.

The accuracy scores of each iteration of testing are not only sufficiently accurate for detecting and recognising the vehicles but are consistent and do not drastically decrease at any iteration without explanation.

5.2 Conclusion

This section will make the final conclusion from the evaluation of the testing results and determine whether the original problem statement has been satisfied.

This project was envisioned as a means of investigation the process of object detection and recognition within machine learning and computer vision. The benefits of a consistently accurate object detection model within the context of autonomous vehicles were outlined in the Project Justification section and include smaller development timeframes for vehicles and cut development costs while promoting safer driving performance from autonomous vehicles.

To conduct this investigation, a primary research method was conceived that would be developed to satisfy a problem statement:

Develop and train a machine learning model for the purposes of object detection and recognition in the context of autonomous driving using semi-supervised learning elements and evaluate its accuracy

To draw a conclusion on the development success of the machine learning model and the success of this project as a whole, the following criteria will be discussed –

- Is the machine learning model able to detect and recognise vehicles in unlabelled images consistently and accurately?
- Did the machine learning model become increasingly accurate at detecting and recognising vehicles throughout the training phase?
- Is the machine learning model able to be used effectively in industry in its current state within the context of a larger overall system in an autonomous vehicle?

From the results of both phases of the Model Testing stage, discussed in the Evaluation of Testing Results section, the conclusion can be made that the machine learning model is able to detect an object, localize it within an image and recognise it as a vehicle consistently and accurately.

As shown in the Model Training stage, the machine learning model did indeed become increasingly accurate at detecting and recognising the vehicles within the images throughout the training phase. The validation accuracy score began at 95.73% and increased consistently over the course of 8 consecutive epochs to an accuracy score of 97.74%. Although not a drastic increase in accuracy ratings, any increase in model accuracy gained through consecutive training epochs is useful and lends to the success of the model. This suggests the semi-supervised learning elements used for training the model were successful and confirms the findings from the literature review that the semi-supervised learning method is the most effective method for developing object detection and other software for autonomous vehicles. Although there is potential for the supervised machine learning method to be implemented for object detection and recognition software, using the information gathered from conducting the primary research in this project it is expected to be less effective than semi-supervised.

In its current state, it is unlikely that the machine learning model could provide any substantial benefit for object detection and recognition within a larger overall system in an autonomous vehicle. This is because the model can only work on still images, and not a live

video feed, which would be the scenario in a functioning autonomous vehicle driving on the road. This, however, was not the goal for this project and therefore does not impact the final conclusion on the success of this project as a whole. If the model were to be further developed in future iterations, it could be used effectively within the context of a driving system for an autonomous vehicle.

Following the discussion on the above criteria and reference back to the original problem statement and goals for this project, the conclusion can be made that the machine learning model developed during the Execution and Development stage does indeed satisfy the problem statement, and this project has been successful. The hypothesis that the model would become consistently more accurate during the Model Training stage due to the model drawing on increased experience has also been proven correct.

5.3 Further Work

This section will consider the further work that could be taken to develop the machine learning model and discuss the steps that could be taken.

Based upon the implementation of the machine learning model and the discussion on its use in a larger system for an object recognition, there is scope for future development of the model which would make the model more practical in a real-world scenario and more useful to be used conjunction with other autonomous vehicle software.

The machine learning model could be further developed to detect and recognise objects in videos and real-time camera feeds. This would allow the model to be significantly more effective in a real-world scenario and would provide a tangible benefit for detecting and recognising objects and vehicles in the context of a larger system for the purposes of an autonomous vehicle driving on roads and making decisions such as how to react to obstacles, whether to change lanes etc. This could be developed by first obtaining a dataset of videos for training and testing purposes and configuring the model to recognise objects in motion using more sophisticated learning methods and techniques.

Machine learning was explored within the context of autonomous vehicles during the literature review, and the production of the machine learning model for this project provided an in-depth perspective on the processes behind the method of object detection. The field of computer vision and object detection will continue to advance at an increasing rate as technologies improve, and the machine learning model developed for this project is only a narrow look at an ever-expanding topic. Currently, the technology powering autonomous vehicles is limited and is not yet sophisticated enough to operate a vehicle entirely independent of a human driver. However, with the rate of advancement of computer vision technology and specifically semi-supervised machine learning methods a future where vehicles are being operated and driven entirely by software without the need for input from a human driver is imminent.

6 Legal, Social, Ethical and Professional Issues

This chapter will present and discuss the different legal, social, ethical, and professional issues that may arise from the development of this project and the machine learning model and its use in the world.

The model developed for this project is only a small section of a very complex industry in computer vision technologies and autonomous driving for vehicles. It was discussed in the introduction chapter that this project could serve as part of a system for recognising objects on the road in the context of autonomous vehicles and as such this context will play a role in evaluating the different issues surrounding the use of the model developed for this project.

6.1 Legal Issues

As computer vision technologies for object detection and autonomous vehicles develop and become more advanced, the legislation surrounding their use will develop simultaneously.

Both the machine learning model developed throughout this project and autonomous vehicles as a whole work by intaking data from the world around them through cameras to analyse. This includes other motorists, their vehicles and individuals on and around roads which could be considered a breach of privacy and raises an issue around the legality of data gathering and how this data is stored in the context of autonomous vehicles. Different countries will need to assess their own legislation around this data gathering as the number of vehicles making use of this technology will increase exponentially over the coming years.

Another important legal issue to consider is driver liability in the case of an accident. Depending on the sophistication of the autonomous vehicle involved in an accident, the determination on the liability of the driver would need to be considered and whether they are to blame for the driving decisions the vehicle made, or the manufacturer of the vehicle and its software are to blame. Currently, most vehicles on the road that make use of some form of machine learning for the purposes of automated driving are still considered to be under the control of the driver and are not fully automated, however this will change as technologies advance and decisions on driver liability when their vehicle is driving automatically will need to be made.

6.2 Social Issues

Social issues are a consideration when envisioning the machine learning model developed being used within the context of autonomous vehicles. There is a discussion to be had on the safety of autonomous vehicles and whether they will be safe enough to be widely used on public roads; if vehicles making use of the model developed in this project and more complex driving systems were to be unsafe there could be an increase in traffic incidents on the road which could cause increased commuting times and even danger for the average motorist.

There is also a discussion to be had on whether non-autonomous vehicles or vehicles that rely entirely on human control should be banned from public roads as the technology powering autonomous vehicles enables them to have higher safety levels on average when driving than normal vehicles driven by people. The number of autonomous vehicles on the roads does not yet outweigh the number of human-controlled vehicles so this discussion is not yet prevalent,

but as the number of vehicles using this type of technology on the roads increases this will be an inevitable social issue in the future.

The impact of these societal issues will be decided by the efficiency of the machine learning models used in autonomous vehicles, and how effective these vehicles are on average at observing the road around them and navigating traffic. This effectiveness will increase as more sophisticated models are developed that are better at detecting and reacting to objects on the road.

6.3 Ethical Issues

Both the machine learning model developed throughout this project and autonomous vehicles already in the world make use of cameras to view the area around them to detect objects and react appropriately. This involves capturing other drivers and information about their vehicle (e.g., car registration numbers), pedestrians, cyclists etc. which is all done without consent. Of course, it is impractical to consider the consent of every person captured by this type of computer vision technology but ethical issues around the use of this data are still a concern.

The ethical issue around data capturing and its use is closely connected with the legislation around computer vision technology previously discussed, but one method of addressing this issue would be to ensure all data captured for this project and autonomous vehicles on the road is only used for the intended purpose of object detection and safe driving and is deleted entirely after it is no longer needed.

An interesting ethical issue that can arise from the use of machine learning models in these types of vehicles surrounds decision making in the event of an unavoidable crash. That is, if a vehicle being driven by software were to encounter a situation where a crash is inevitable, for example between two cars, how does it decide which direction to swerve and which car to impact with? If a model were to be constructed that could make this decision based on perceived safety of one impact over the other, an issue on how that model is ethical would arise, which could impact the behaviour of other road users.

6.4 Professional Issues

There is a concern that as autonomous vehicles become more advanced, the need for jobs such as taxi and bus drivers, lorry drivers etc. will be diminished as companies can avoid employing humans to do the jobs and save money on wages, insurance etc. This would eliminate several job roles for people and potentially increase unemployment rates. However, there has been a higher demand for lorry and freight drivers in recent years, particularly due to COVID-19, which could be alleviated if autonomous driving in vehicles advances to the point where delivery vehicles could perform their function without the need for a human driver. Although this level of autonomous driving is a long way off, it is a possibility to be considered in the future.

Other jobs that also involve a large amount of driving but still require a person, for example police officers, could experience a change in job dynamic and wages because of autonomous vehicles. As they become more prevalent on roads, crimes such as speeding, careless driving and illegal parking will diminish which could result in a reduced need for police officers or possibly lower wages.

Appendix

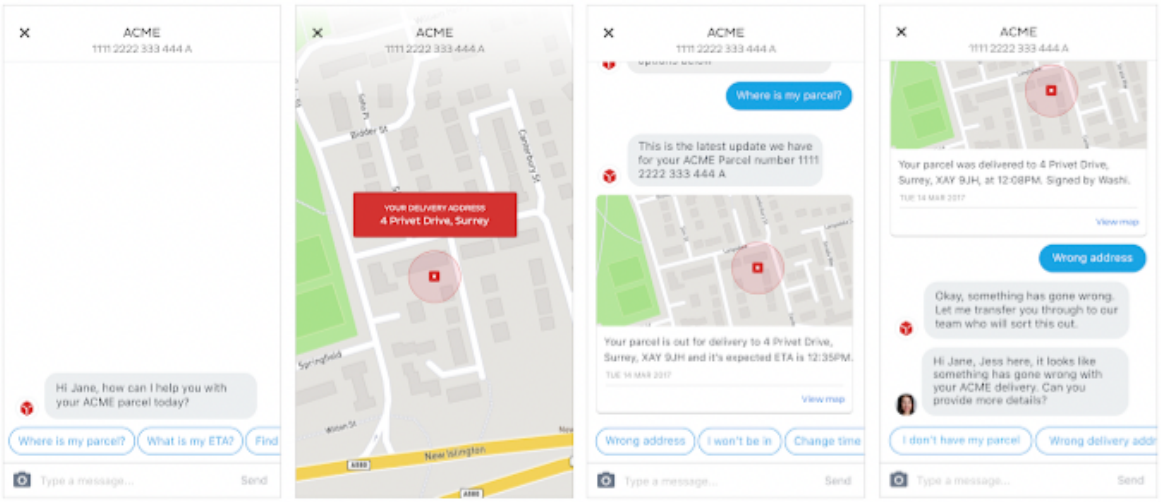


Figure 1: DPD SML Chatbot by Google [8]

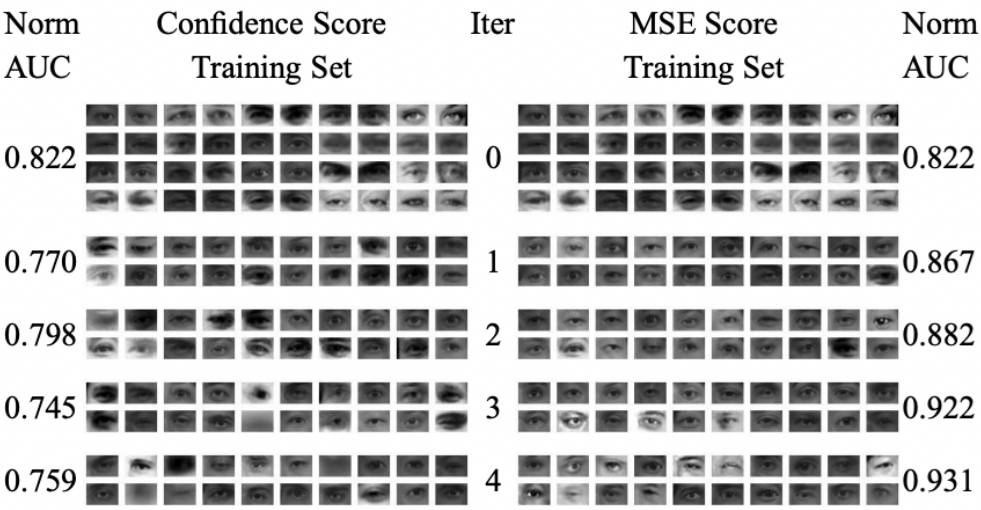


Figure 2: Accuracy of SSML Detector using different Selection Metrics after each iteration [18]

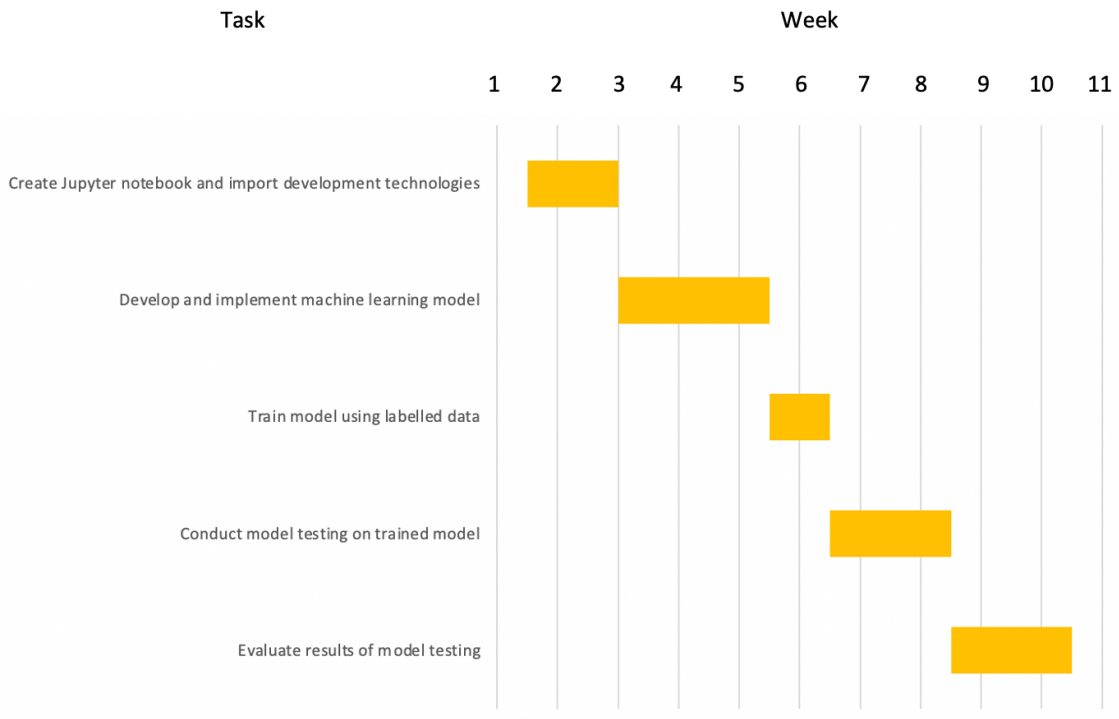


Figure 3: Project Lifecycle Gantt chart

```
In [5]: Data.head()
```

Out[5]:

	image	xmin	ymin	xmax	ymax
0	vid_4_1000.jpg	281.259045	187.035071	327.727931	223.225547
1	vid_4_10000.jpg	15.163531	187.035071	120.329957	236.430180
2	vid_4_10040.jpg	239.192475	176.764801	361.968162	236.430180
3	vid_4_10020.jpg	496.483358	172.363256	630.020260	231.539575
4	vid_4_10060.jpg	16.630970	186.546010	132.558611	238.386422

Figure 4: Initial entries of train_solution_bounding_boxes.csv

```
Photo shape: (380, 676, 3)
Name,xmin,ymin,xmax,ymax: ['vid_4_1000.jpg' 281.2590449 187.0350708 327.7279305 223.225547]
```

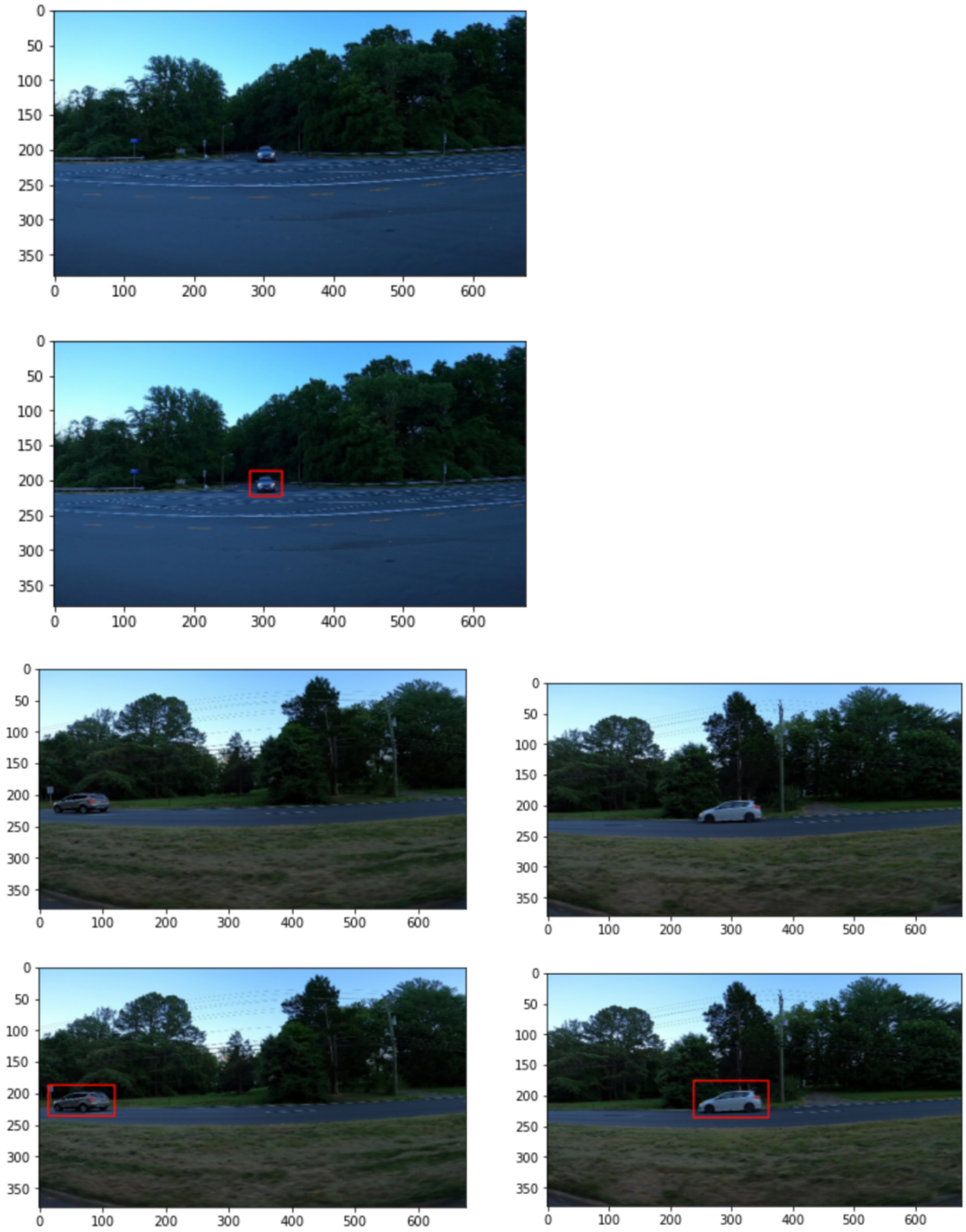


Figure 5: Examples of images before and after bounding box

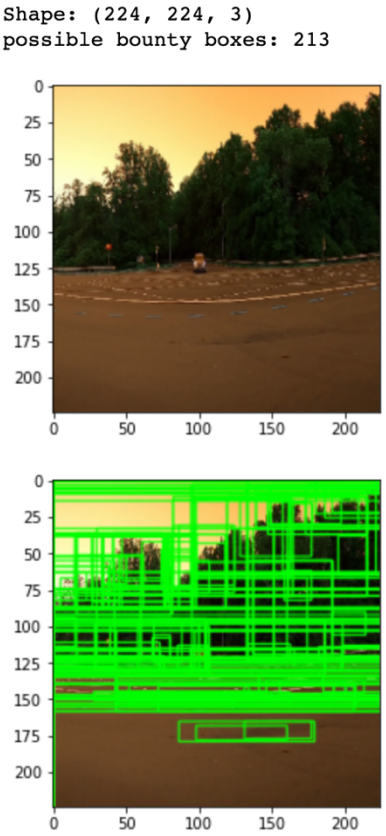


Figure 6: Example of bounding boxes surrounding all detected objects prior to segmentation

```
vid_4_9620.jpg 541 1644
vid_4_9720.jpg 542 1623
vid_4_9760.jpg 543 1537
vid_4_9760.jpg 544 1537
vid_4_9760.jpg 545 1537
vid_4_9740.jpg 546 1599
vid_4_9700.jpg 547 1630
vid_4_9780.jpg 548 1477
vid_4_9780.jpg 549 1477
vid_4_980.jpg 550 1099
vid_4_9800.jpg 551 1513
vid_4_9800.jpg 552 1513
vid_4_9820.jpg 553 1386
vid_4_9840.jpg 554 1565
vid_4_9860.jpg 555 1497
vid_4_9880.jpg 556 1450
vid_4_9900.jpg 557 1499
vid_4_9960.jpg 558 1380
vid_4_9980.jpg 559 1379
```

Figure 7: Training images being loaded and matched to relevant bounding box parameters

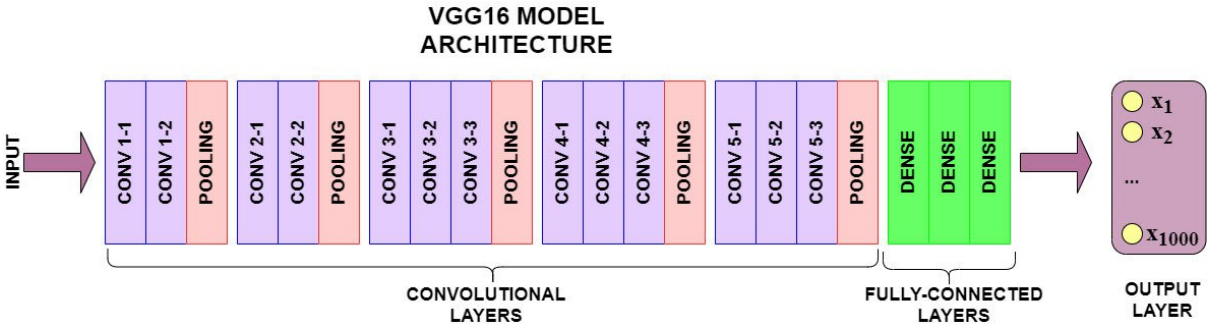


Figure 8: VGG16 model architecture

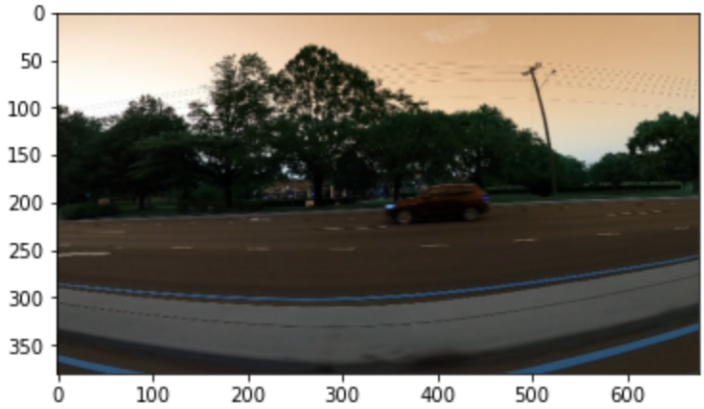
```
In [23]: base_model=tf.keras.applications.VGG16(include_top=False,input_shape=(224,224,3),weights='imagenet')

In [24]: base_model.summary()
Model: "vgg16"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 input_1 (InputLayer)         [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0

=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

Figure 9: VGG16 architecture within Jupyter notebook

```
In [30]: epoch=8
         hist=model.fit(x_train,y_train,epochs=epoch,validation_data=(x_val,y_val))
Epoch 1/8
222/222 [==============================] - 1266s 6s/step - loss: 0.7614 - accuracy: 0.7952 - val_loss: 0.1581 - val_a
ccuracy: 0.9573
Epoch 2/8
222/222 [==============================] - 1227s 6s/step - loss: 0.2587 - accuracy: 0.9289 - val_loss: 0.1239 - val_a
ccuracy: 0.9656
Epoch 3/8
222/222 [==============================] - 1226s 6s/step - loss: 0.2094 - accuracy: 0.9405 - val_loss: 0.1124 - val_a
ccuracy: 0.9702
Epoch 4/8
222/222 [==============================] - 1231s 6s/step - loss: 0.1945 - accuracy: 0.9457 - val_loss: 0.1027 - val_a
ccuracy: 0.9728
Epoch 5/8
222/222 [==============================] - 1226s 6s/step - loss: 0.1605 - accuracy: 0.9516 - val_loss: 0.0976 - val_a
ccuracy: 0.9731
Epoch 6/8
222/222 [==============================] - 1224s 6s/step - loss: 0.1573 - accuracy: 0.9567 - val_loss: 0.0899 - val_a
ccuracy: 0.9739
Epoch 7/8
222/222 [==============================] - 1223s 6s/step - loss: 0.1542 - accuracy: 0.9587 - val_loss: 0.0969 - val_a
ccuracy: 0.9759
Epoch 8/8
222/222 [==============================] - 1233s 6s/step - loss: 0.1401 - accuracy: 0.9584 - val_loss: 0.0867 - val_a
ccuracy: 0.9774
```

Figure 10: Model training process over 8 epochs and validation accuracy scores

```
Number of possible objects detected:  1801
Number of boxes with a class prediction of 1: 26
--------------------------------------------------------------------------------
```



Class number = 1 and bounding box shown has the highest probability.: %97.66657948493958

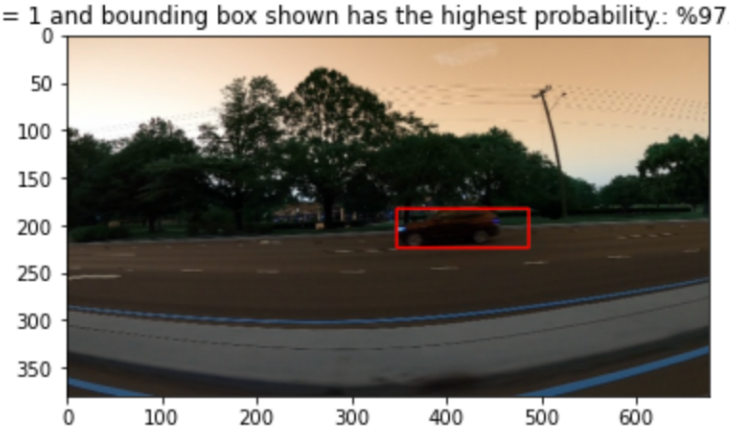
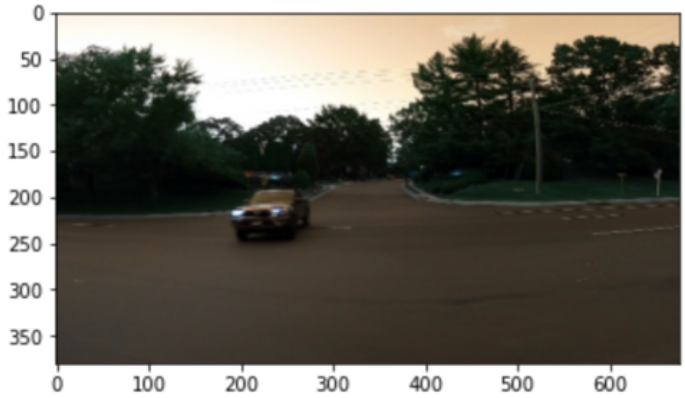


Figure 11: First iteration of testing subset phase

```
Number of possible objects detected:  1392
Number of boxes with a class prediction of 1: 56
--------------------------------------------------------------------------------
```



Class number = 1 and bounding box shown has the highest probability.: %99.97303485870361

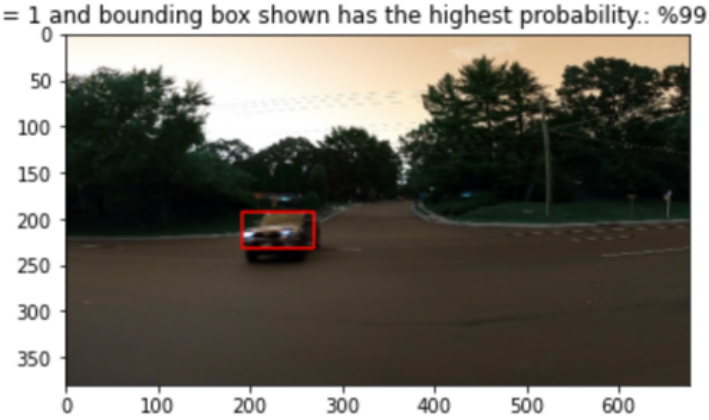
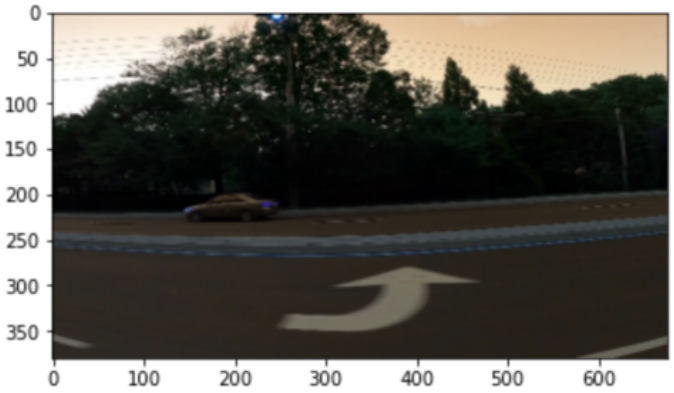


Figure 12: Testing subset iteration 2

```
Number of possible objects detected:  1818
Number of boxes with a class prediction of 1: 45
--------------------------------------------------------------------------
```



Class number = 1 and bounding box shown has the highest probability.: %99.30956363677979

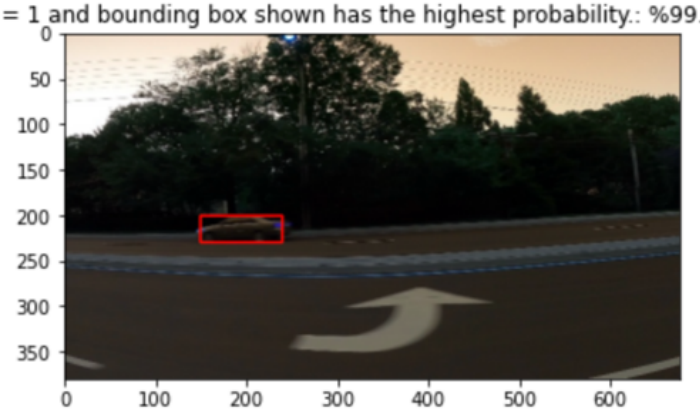
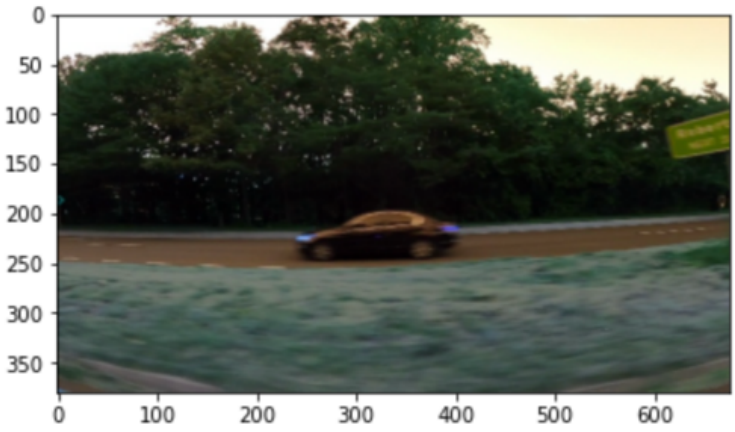


Figure 13: Testing subset iteration 3

```
Number of possible objects detected:  1418
Number of boxes with a class prediction of 1: 36
--------------------------------------------------------------------------------
```



Class number = 1 and bounding box shown has the highest probability.: %99.76428151130676

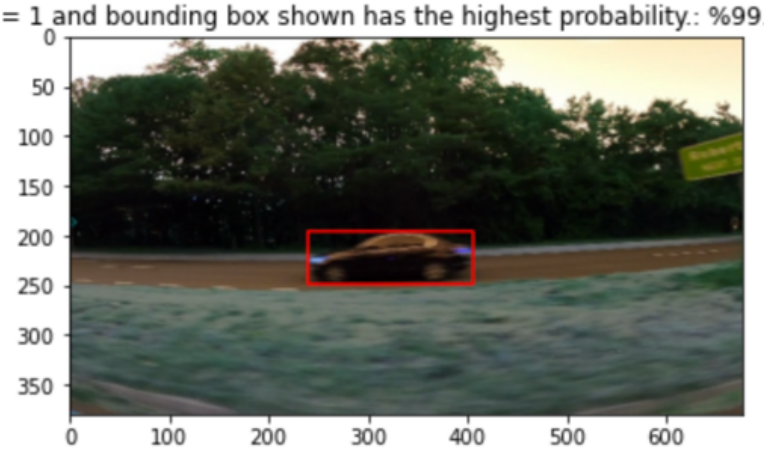
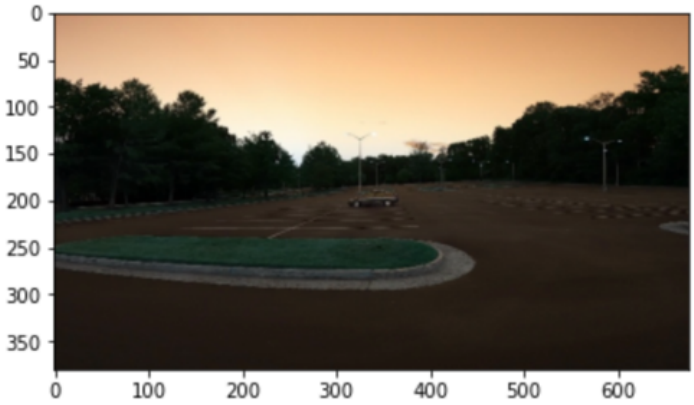


Figure 14: Testing subset iteration 4

```
Number of possible objects detected:  1187
Number of boxes with a class prediction of 1: 4
--------------------------------------------------------------------------------
```



Class number = 1 and bounding box shown has the highest probability.: %96.10034227371216

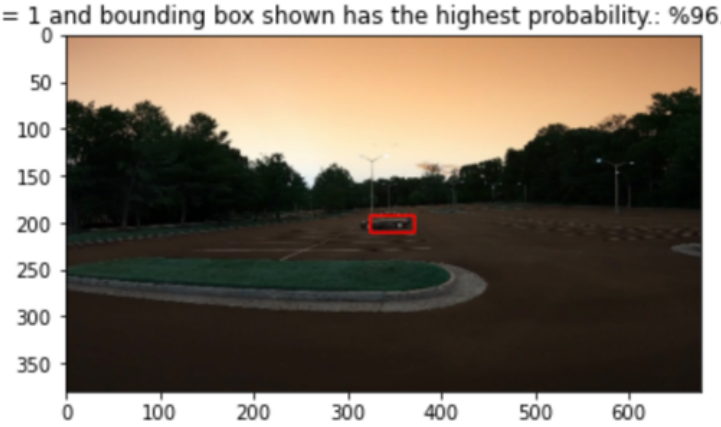
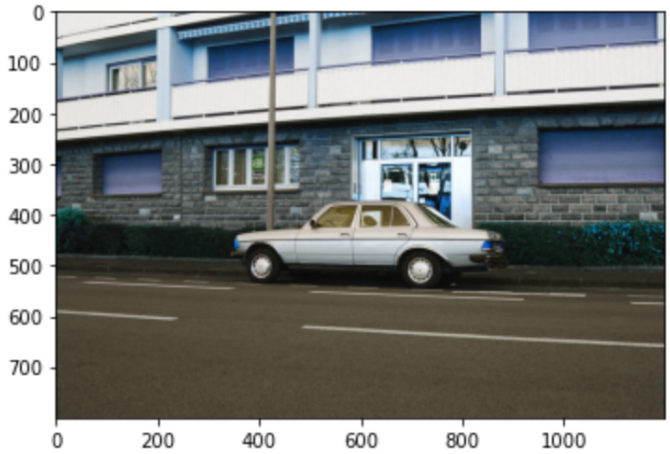


Figure 15: Testing subset iteration 5

```
Number of possible objects detected:  6896
Number of boxes with a class prediction of 1: 393
--------------------------------------------------------------------------------
```



Class number = 1 and bounding box shown has the highest probability.: %100.0

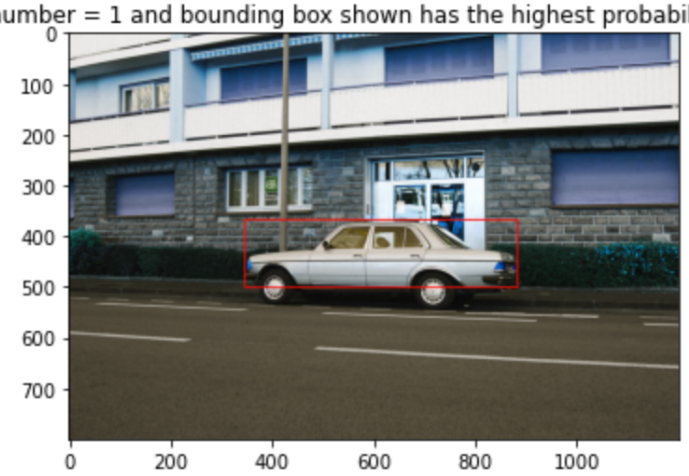
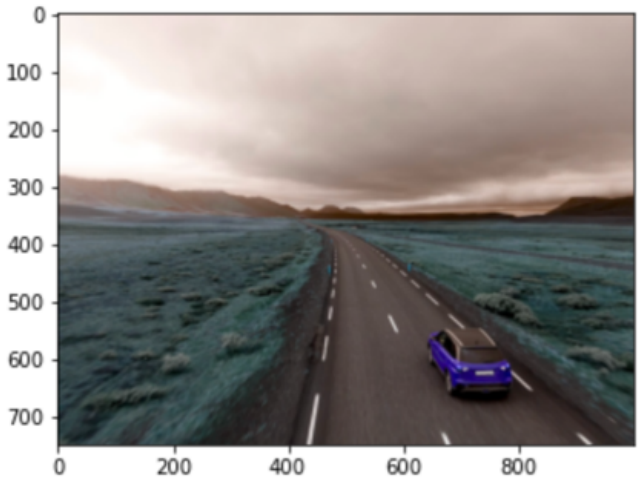


Figure 16: First iteration of Google dataset testing phase

```
Number of possible objects detected:  1884
Number of boxes with a class prediction of 1: 101
--------------------------------------------------------------------------------
```



Class number = 1 and bounding box shown has the highest probability.: %99.98020529747009

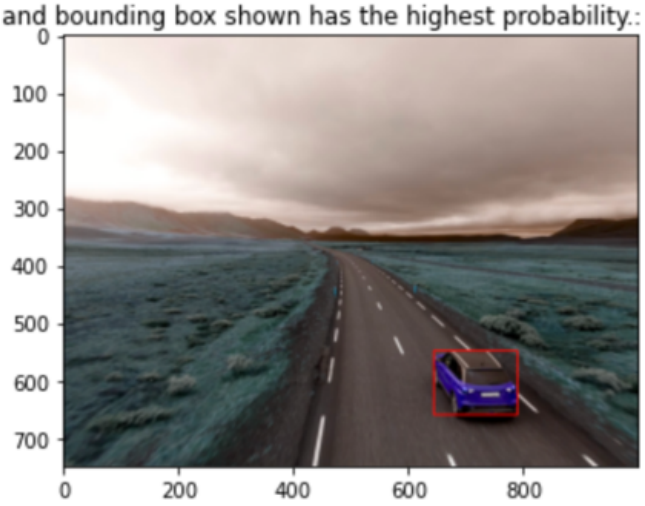
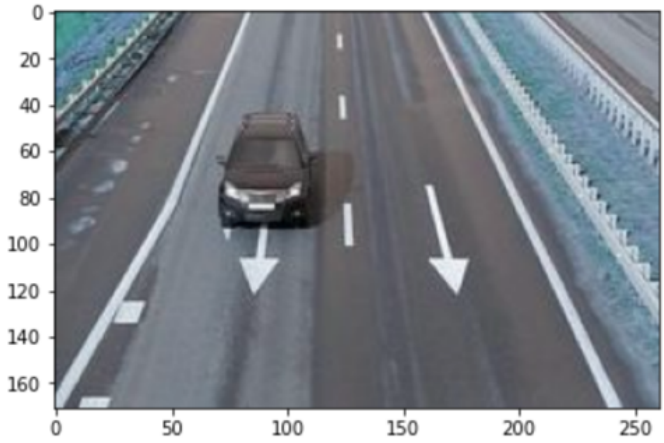


Figure 17: Google dataset iteration 2

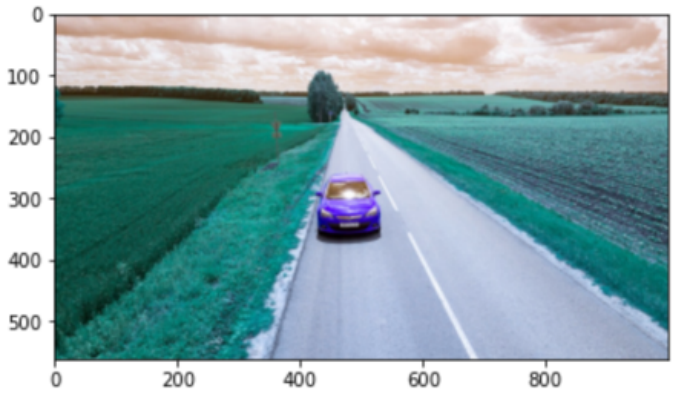
```
Number of possible objects detected:  331
Number of boxes with a class prediction of 1: 19
--------------------------------------------------------------------------------
```



Class number = 1 and bounding box shown has the highest probability.: %99.24404621124268

Figure 18: Google dataset iteration 3

```
Number of possible objects detected:  2073
Number of boxes with a class prediction of 1: 84
--------------------------------------------------------------------------------
```



Class number = 1 and bounding box shown has the highest probability.: %99.99997615814209

Figure 19: Google dataset iteration 4

```
Number of possible objects detected:  1045
Number of boxes with a class prediction of 1: 76
--------------------------------------------------------------------------------
```

Class number = 1 and bounding box shown has the highest probability.: %99.99748468399048

Figure 20: Google dataset iteration 5

Tables

Dataset	Image Name	Objected Detected Y/N	Accuracy Score
Testing subset	vid_5_27620.jpg	Y	97.66%
“	vid_5_29820.jpg	Y	99.97%
“	vid_5_29020.jpg	Y	99.30%
“	vid_5_26620.jpg	Y	99.76%
“	vid_5_420.jpg’	Y	96.10%
		Average Accuracy:	98.55%
Google dataset	test1.jpeg	Y	100.00%
“	test2.jpeg	Y	99.98%
“	test3.jpeg	Y	99.24%
“	test4.jpeg	Y	99.99%
“	test5.jpeg	Y	99.97%
		Average Accuracy:	99.83%

Table 1: Testing accuracy scores over both testing phases

References

[1] Van Engelen and Hoos, 2019, A Survey on Semi-Supervised Learning, Springer

[2] Singh, Nowak and Zhu, 2008, Unlabelled Data: Now It Helps, Now It Doesn't, University of Wisconsin

[3] Plasek, 2016, On the Cruelty of Writing a History of Machine Learning, IEEE

[4] Turing, 1950, Computing Machinery and Intelligence, Mind 49

[5] Paluzek and Thomas, 2017, An Overview of Machine Learning, O'Reilly

[6] IBM, 2020, Supervised Machine Learning, IBM

[7] Google, 2020, DialogFlow Chatbot, Google

[8] DPD, 2021, Contact Us Chatbox, DPD

[9] Burscher, Vliegenthart and De Vreese, 2015, Using Supervised Machine Learning to Code Policy Issues: Can Classifiers Generalise across Contexts?, SAGE Journals

[10] IBM, 2020, Unsupervised Learning, IBM

[11] Google, 2021, Machine Learning Glossary section U, Google

[12] Raschka, 2021, Machine Learning FAQ Semi-Supervised Machine Learning, Sebastian Raschka

[13] Balcan and Blum, 2005, A PAC-Style Model for Learning from Labelled and Unlabelled Data, Carnegie Mellon University

[14] Lafferty and Wasserman, 2007, Statistical Analysis of Semi-Supervised Regression, Carnegie Mellon University

[15] Ben-David, Pál and Lu, 2008, Does Unlabelled Data Provably Help? Worst-case Analysis of the Sample Complexity of Semi-Supervised Learning, Research Gate

[16] Liu, Yang, Huang, Yeo, Lin, 2016, Driver Distraction Detection Using Semi-Supervised Machine Learning, IEEE

[17] Rosenberg, Herbert and Schneidermann, 2005, Semi-Supervised Self-Training of Object Detection Models, Google Inc. and Carnegie Mellon University

[18] Rosenberg, Herbert and Schneidermann, 2005, Semi-Supervised Self-Training of Object Detection Models Figure 2, Google Inc. and Carnegie Mellon University

[19] Mohseni, Pitale, Singh and Wang, 2019, Practical Solutions for Machine Learning Safety in Autonomous Vehicles, Texas A&M University and NVIDIA

[20] Tuncali, Fainekos, Ito and Kapinski, 2018, Simulation-Based Adverserial Test Generation for Autonomous Vehicles with Machine Learning Components, IEEE

[21] Navarro, Fernandez, Borraz and Alonso, 2016, A Machine Learning Approach to Pedestrian Detection for Autonomous Vehicles Using High-Definition 3D Range Data, MDPI

[22] Shi, Wong, Chai and Zhi-Feng Li, 2020, An Automated Machine Learning (AutoML) Method of Risk Prediction for Decision-Making of Autonomous Vehicles, IEEE

[23] Soares, Angelov, Costa and Castro, 2019, Actively Semi-Supervised Deep Rule-Based Classifier Applied to Adverse Driving Scenarios, IEEE

[24] Li, Hong, Wang and Liu, 2019, Fatigue Driving Detection Model Based on Multi-Feature Fusion and Semi-Supervised Active Learning, The Institution of Engineering and Technology

[25] Ding, Meng, Zhang, Zhenzhen, Jiang, Xinyan, Cao and Yunfeng, 2020, Vision-Based Distance Measurement in Advanced Driving Assisstance Systems, ProQuest

[26] Mohktari, 2021, Which is Better for Your Machine Learning Task, OpenCV or TensorFlow?, Towards Data Science

[27] OpenCV, 2022, OpenCV

[28] Keras, 2022, Keras

[29] Python, 1997, Comparing Python to Other Languages, Python

[30] Kaggle, model training and testing datasets for object detection, Kaggle

[31] GitHub, 2021, GitHub

[32] Oates, 2005, Researching Information Systems and Computing, 8.1 Research and Development Methodology, SAGE Publications

[33] Eby, 2017, What’s the Difference? Agile VS Scrum VS Waterfall VS Kanban, SmartSheet

[34] J.R.R Uijilings et al. Selective Search for Object Recognition, International Journal of Computer Vision