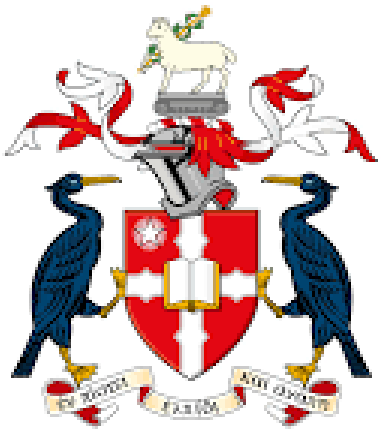
Using AI in motorway toll booths to apply appropriate rates based on vehicle type recognition.



LIVERPOOL HOPE UNIVERSITY

Adrian Urbańczyk

Department of Mathematics, Computer Science and Engineering

Liverpool Hope University

Supervisor: Dr Ogbonnaya Anicho

This report is submitted in partial fulfilment of the requirement for the degree of Bachelor of Science in Computer Science

May 2023

Plagiarism Statement

I confirm that this assignment is my own work, no part of it has copied from any other person's work (p published or unpublished), and any text that has been paraphrased has been cited in the body of the document with the full details of the original work included in the reference list.

I confirm that I have read the guidance provided at:

https://www.hope.ac.uk/gateway/students/studentadministration/understandingyourdegree/

I confirm that I understand the consequences of academic misconduct and that it may result in termination of my studies at Liverpool Hope University.

Acknowledgements

I would like to thank my supervisor Dr Ogbonnaya Anicho for his guidance and support while working on this dissertation.

Table of Contents

List of Figures

List of Tables

1 Introduction

Artificial intelligence has revolutionized the way businesses operate in today's world. Once confined to the tech industry, AI has now started being used in non-tech industries such as health care, the food industry, or travel. Many of them noted substantial growth and development due to technological improvement by more advanced software, algorithms, and machine learning. The development of Artificial Intelligence, Machine Learning and Deep Learning drives the growth of many industries and increase the value and quality of the product and services provided. One of the most rapidly growing fields in this realm is computer vision which empowers computers to recognize and comprehend objects in images. The industry has been growing rapidly through most recent innovations like self-driving cars, robots, surveillance systems and any software that utilizes AI for image processing.

1.1 Background

This dissertation research will focus on vehicle type classification in the images into the four most common vehicle classes which are, Truck, Bus, Car and Motorcycle in a motorway toll booth environment. The idea is to automate the process of charging drivers when going through the highway or any type of road where the charge is applied. The system should be able to detect the vehicle type based on the image provided and then eventually apply an appropriate charge.

1.1.1 AI in Transportation and Automotive Industry

AI in the transportation and automotive industry has been growing especially with the application of computer vision and machine learning being among the most notable developments. These are highly used in self-driving cars, which have gained significant attention as a potential use case for AI in transportation. However, it is important to note that despite the progress made so far, the technology is still in its developmental stage, and it still has not replaced traditional vehicles yet. Nonetheless, AI's impact on the transportation and automotive industry has been substantial, with the potential to revolutionize the sector in the future.

1.1.2 Self-driving cars

Object detection models are used in self-driving vehicles, as it has to be capable of detecting objects like signs, road, traffic lights, pedestrians, and other vehicles. The cars use sensors mounted on the vehicle to collect various types of data and react properly to the current situation in which the vehicle is. The way the car behaves is based on all of that collected information. That technology has a huge potential and could solve many modern world problems. Fully self-driving trucks for example, will greatly reduce operating costs and environmental impact. Nowadays the environmental impact and automation is a great motivation for companies to invest in such technology as it can lead to increasing company's revenue and lower the costs of the business [10]. The AI could increase safety on the roads and prevent accidents, by gathering data from various sensors in the car about the vehicle and the driver it could prevent for example from drunk driving or falling asleep while behind the wheel. M. Hashemi et al. created such a monitoring system with the usage of deep learning in their research [12].

1.1.3 Traffic signs detection

Traffic signs detection is crucial while driving, it ensures moving the vehicle according to the law and in an expectable way for other drives. The better the understanding of the signs and law the safer the roads are. Doing that manually for a long time can be difficult for a human, so traffic signs detection using deep neural networks was introduced [13]. Most of these applications are in self-driving cars, while the industry is growing it may find different usages for that, such as driving assistance. It has a significant impact on road safety, as each year over 1000 people are killed in the US [10], because of drivers running a red light. This can be reduced to zero if all cars were self-driven as it will simply not allow the car to do it. We cannot avoid people making mistakes while driving, but AI over time may become much less accident-prone than humans. There are already introduced deep learning models to detect many signs in a short time, using mounted cameras on the vehicle [13].

1.1.4 Pedestrian detection

Together with self-driving cars, pedestrian detection is an absolute must, even in manually driven cars it could have huge advantages. This problem is particularly tough for artificial intelligence models as people are simply hard to predict, however, there are already existing models, which can warn the driver of various objects or even a pedestrian willing to cross the road. The real problem is when a pedestrian is in a wheelchair or has a pet or some unknown object, like bags of groceries, such objects could trick the AI [14]. As it involves human beings it is especially important to be able to predict it accurately if not perfectly.

1.1.5 Traffic Flow Analysis

Analysing congestion and crowding in cities can bring a lot of advantages, as both of them have an impact on people's lives like traffic, noise, or air pollution. Studying traffic patterns may reduce traffic while road maintenance is performed or construction causes some roads to be inaccessible. The model can guide drivers to omit congested roads and find optimal routes to their destinations [14]. Computer vision models provide precise information like traffic density, and freeway traffic count which could result in improved traffic flow if connected to some global system and as the transportation and autonomous vehicle industry groves it will play an important role there [15].

1.1.6 Parking Management

As cities are getting more urbanized, it also means there are more vehicles on the road. Over time finding parking spots in the city centre or around the city centre became time-consuming and simply difficult. By connecting the navigation system with AI, it could locate a free parking spot and directs the car to specific places where the car will fit and forewarn about eventual traffic congestion [14]. This software combined with plate recognition may significantly improve the process of parking [15].

1.2 Motivation

The automation with Artificial Intelligence classification models such as vehicle type recognition could significantly increase the throughput of a tool booth or any toll gate which prices are based on the vehicle type. That could motivate companies and attract their interest into investments in such technologies which would increase the value of their product and the quality of the services

provided. Vehicle type classification could also be used in self-driving cars or other traffic analysis devices which could lead to greater safety on roads. Especially when well-trained AI would take control over some human's actions who by making mistakes could put someone's life in jeopardy, in that moment advanced and properly trained AI could eliminate the potential danger. Smart usage of AI by companies in the transport industry could lead to increasing the company's revenue and overall income. By applying various types of artificial intelligence, the company could assign more complex tasks to people and increase their salaries as simple tasks were automated by AI models.

1.3 Aims and Objectives

The primary objective of this research is to create an AI model for vehicle type classification presented on an image to apply an appropriate charge for a vehicle type passing through. The investigation includes determining if the model can classify the images with accuracy enough to make the process smoother and present visible improvement. Additionally, the research aims the model to be capable of working in a completed system thus its time complexity and hardware requirements must be considered as well. In general, software should be as efficient as possible while keeping its accuracy.
There are two objectives to be achieved in this research project.

- Create an efficient AI model to classify vehicle images into four types, Bus, Car, Motorbike, and Truck. The model shall have an overall accuracy of at least 60% on a test set with enough per-class accuracy to be able successfully to fulfil required tasks.
- Design and implement a user interface to run the software and examine the results. UI should go through all the images and display per-class predictions.

2 Literature Review

Deep learning and computer vision have been developing rapidly and it became an interesting topic, particularly for researchers. There are papers explaining foundations of image processing, convolutional neural network and other inventions used in computer vision. Furthermore, vehicle type classification and vehicle detection were researched in different ways and in different environments which gives a great overview of the problem and its challenges. The datasets were widely discussed in these papers as there is not a publicly available dataset which would cover all of the common vehicle types from different regions and modifications of the vehicles and the dataset should be fitted to the specification of the problem.

2.1.1 Convolution Neural Network

Convolution Neural Networks are widely used in deep learning, replacing basic artificial neural networks in certain types of problems. They have proven their great performance in solving many machine learning and computer vision problems [1]. The maths in CNNs was nicely explained in Jianxin Wu's paper, it allows us to approach the neural network only from the mathematical side, which helps to understand how it works and processes data. A proper understanding of maths certainly helps in developing models which use CNNs. They are primarily used for pattern recognition, occurring in images or voices. They were created to solve problems with large data which normal artificial neural networks could not handle. Also, in classification models where the location of a classified object is not needed because with CNNs the only concern is to detect the object or its type and not its location [2] which fits perfectly with the potential solution to the

problem, where a live camera on a highway is monitoring what vehicle approaches the toll booth, only its type is needed not the location. The camera may be moved by some independent factors like weather or strong wind, these conditions should not affect the model performance. Convolution neural networks have proven their performance in image classification therefore they have been chosen to be the foundation of the model to solve the thesis problem. Ideally, that would be the model's job to classify an image taken from the camera, there will be additional software required to decide when to take a snapshot of a vehicle from the liv footage.

2.1.2 Convolution Neural Network over Artificial Neural Networks

Convolutional neural networks have many architectures and variants. To the CNN basic components, we can include convolution layers, pooling layers and fully connected layers. By using filters, kernels, strides, convolution layers and pooling they are more likely to classify the image correctly than normal neural networks. Convolution layers aim to find features of the input. Features are something special about the input that may be specific for certain classes or may help to differentiate them from other classes. For example, if a model classifies cat and dog images, the features would be dropped or pointed ears, the shape of eyes or mouth. However, these features which were found by CNN may differ from what people would find. The largest limitation of Artificial Neural Networks is that they struggle with computing image data due to their complexity. For example, if we take images with dimensions of *28 x 28 x 1* where one refers to black or white, a single neuron in the first hidden layer will have 784 weights which most artificial neural networks can handle. If we take a coloured image with dimensions of *64 x 64 x 3* (RGB) the weights on the first hidden layer on a single neuron would be 12,288 [4]. Reducing computational complexity with CNNs also helps reduce the risk of overfitting by decreasing the number of parameters required to train which helps to prevent the network from overfitting and improve the final results of the model [4]. ANNs fail to train effectively if the provided input is too big such as images, it's caused by the architecture of ANNs, specifically the fully connected manner of standard ANN neurons. To overcome this every neuron in the convolutional layer is connected to a small piece of the input volume [4]. If we again bring the example with *a 64 x 64 x 3* image in a convolutional layer, if we set the receptive field dimensions as *6 x 6* each neuron would have 108 weights whereas standard ANN neurons would have 12,288 as calculated above.
There are three extra hyperparameters which help CNNs reduce the complexity of the model, these are the depth, the stride and setting zero-padding [4].

2.1.3 Vehicle Detection

Vehicle detection is a widely known problem, it is a crucial component of traffic surveillance, automatic driving, speed prediction or object detection [5]. Each of these types of models works differently, it also depends on if the data is a live footage or just a static photo. It is worth considering that live object or vehicle detection differs from image detection not only by computational complexity but also models and algorithms used.

2.1.3.1 Existing vehicle detection models

Vehicle detection or vehicle-type detection is widely used in traffic surveillance systems like licence plate detection, speed prediction, intelligent traffic control, public safety and security [7]. It can be

divided into two categories, general detection where the model simply guesses the type of the vehicle determined by some major characteristics for example truck, car, motorbike, or bus. The second one is a fine-grain where the model recognizes two different models of the same brand. M. Atif Butt et al. tested 5 pre-trained models for vehicle type classification. All of them can be encountered in research papers expanding or improving their performance or just analysing or comparing them on different datasets for different purposes [6]. These models were tested on the same dataset, which was collected from road surveillance videos. Vehicles were extracted from the frames and put into 6 classes, car, and bus. Van, truck, motorbike and rickshaw. It is worth mentioning that the datasets come from many environments and different parts of the world which have an impact on the result and will be explained later. These models are:

1. AlexNet by Alex Krizhevsky and his collaborators
2. VGG by K. Simonyan and A. Zisserman
3. GoogleNet by Google
4. ResNet
5. Inception

All of these models got over 92% accuracy on the same dataset before applying fine tuning which was applied to ResNet as it had the best accuracy 95%, The fine-tuning is adding one new classification block. It resulted in accuracy going up to 99.6%. For the problem of this research the general vehicle type recognition will be more than enough thus there is no need to use sophisticated models and put more effort into gathering the data. However, there are a few things that had to be taken into account like environment, country and placement of the camera. M. Atif et al. described in their paper how significant taking environmental characteristics like for example weather which may distort the image and final guess. Also, a lot depends on the data that the model is trained on as many researchers like M. Atif [7] or Linkai Chen [5] found it difficult to find appropriate datasets for vehicle type classification. Another issue is what type of vehicles are in the dataset, some of them may be customised or modified by their owners significantly which will trick the model. Also if we train the model on vehicles from for example Asia, the model will be the most accurate there, as in Europe different cars or trucks may be more popular or simple some of the vehicles may look different as they are newer and have a modern design.

2.1.3.2 Datasets issues

The size and quality of the dataset have a significant impact on the model and its efficiency. If the model would be used in modern intelligent transportation systems the dataset that it is trained on had to have multiple vehicle classes, decent quality images and vehicles from multiple regions. Certainly, there is a lack of datasets for general vehicle type classification and fine-grain vehicle recognition. In many research papers, we can encounter that the authors use powerful and tested pre-trained models simply to gather more images or images which will suit their needs.

M. Atif Butt et al. in their research paper before training the model performed data augmentation to artificially expand the dataset and mitigate overfitting which is the most common way of doing it. It was done by applying Gaussian blur and noise to the image, rotating it or flipping horizontally [6].

X. Ke et al obtained images to their dataset by scraping major automotive websites which provide a general picture of their cars. However, these images could be classified into two categories: overall and detailed appearances. The detailed photos do not have much of a use therefore they had to be removed from the dataset. For this purpose classification strategy had been used to delete these photos from the dataset before training. Furthermore, Faster-RCNN for vehicle detection was used to automatically label gathered images to split them into appropriate categories [7].

Certainly, there is a problem with vehicle datasets for classification, in most research papers the authors have to somehow gather data or modify existing data to fit the purpose of the research and training requirements. There are few existing vehicle datasets which have appropriate types of images, which show the vehicle features and have decent picture quality. Stanford car dataset has over 16000 photos where there is always one car in the picture with the right perspective which demonstrates a car's features, however, the problem is that the dataset contains only cars and all of them are modern vehicles which may cause the model to struggle to recognize older vehicles [6]

2.1.3.3 Real-time applications

In real-time applications where frames are analysed, much more resources are used than in processing just one image. Therefore, time complexity and efficiency of the model have to be taken into account. There also can be a case when some extra software, not necessarily AI or machine learning, lets the model work only with single frames. The models that were tested by M. Atif Butt et al. demonstrated great accuracy on a test set, furthermore, ResNet after fine-tuning got close to 100% accuracy. However, they would not work well in real-time applications as there were limited classes, and the datasets were unbalanced datasets. The database was self-constructed which may cause problems as vehicles were collected from different regions of the world and a wide variety of vehicles modernity [6].

3 Methodology and Research Design

The approach to the problem and the solution design will be explained in this chapter. There are many steps in the approach as the solution has many layers of implementation. The components of the implementation process will be discussed, these are the dataset, model selection, and techniques applied to improve the model.

3.1 Simulation setup

The primary programming language is Python 3.9, and the used libraries are TensorFlow 2.4.1, scikit-learn, NumPy, and QT Framework in an Anaconda environment. The pre-trained models will be downloaded and imported from the Keras library [11]. The TensorFlow was set up to use GPU for training with the tensor library. The calculations and training will be performed on a local machine on GPU as it has greater calculation power than the CPU in the case of training artificial intelligence models.

Local Machine Hardware Specification	
Name	Specification
CPU	Intel Core i5-10300H (4 cores, 8 threads, 2.50-4.50 GHz, 8 MB cache)
GPU	Nvidia RTX 2060 6 GB
RAM Memory	16 GB (DDR4, 3200 MHz)
Operating System	Linux Ubuntu 20.04

Table 3.1: Local Machine Hardware Specification

Table 3.1 presents the hardware specification of the local machine on which all the models were trained. The GPU driver version was 520.61.05 and CUDA Version was 11.8.
With this setup it was possible to train simpler models with their initial input value, which is 224x224x3, however, it was not enough for models more complex than EfficientNetB2 [11] thus the input for the models trained will be 128x128x3.

3.2 Vehicle Type Classification

To find the best model for the classification problem, both existing and self-made neural networks will be considered and compared. The approach and decision made will be discussed in this chapter and the exact implementation in the following one.

3.2.1 Datasets

In a deep learning classification model dataset plays a crucial role in training. It is important to train the model on accurate and high-quality data which relate to the problem. For now, there is no public vehicle dataset which contains commonly used vehicles. Additionally, to acquire maximum generalization and to make to model to be used everywhere the dataset should contain vehicles manufactured with modern and older technology as there are a lot of differences between modern and older vehicles [6]. Two datasets were analysed and compared, and a few first models were tested on both of them to deteriorate the most suitable one for the problem. However, the decision to compare two datasets was not planned, the choice came from the dataset analysis.

3.2.1.1 Datasets Overview

The first dataset was acquired from Kaggle.com, however, it was a combination of two separate collections of data. The second dataset was gathered from the Open Images repository, for this research first dataset will be called the combined dataset, and the second one is just the Open Images dataset.

Class	Combined Dataset	Open Images Dataset
Car	4489	6781
Truck	4423	2033
Bus	3638	2133
Motorbike	3655	2986

Table 3.2: Number of images of classes in both datasets

Table 3.2 presents the number of images of each class in both datasets. The combined dataset is more balanced, and the Open Images dataset has a significantly higher number of images of cars than other classes. Combined data was a collection of two datasets where one of them had bikes and cars and the other one had buses and trucks. After analysing the data few conclusions were drawn, light vehicle images (cars and motorcycles) and heavy vehicles image (buses and trucks) had significant differences. Most of the light vehicle images were taken in ideal environments, where the vehicle takes most of the space in the image and the background is plain. Additionally, the vehicles were not in their place of occurrence like roads or highways. The images were perfect for advertisements and vehicle models' catalogues.

Figure 3.1: An example of images from the bus and car dataset

Figure 3.2: Example of outliners in the heavy vehicle collection

On the other hand, heavy vehicle images were taken in cities, on the roads or stopping at bus stops, which means that the environment was dynamic and in some of these images the vehicle was not the main part of the image. Figure 3.1 presents a comparison of these two collections of data. Additionally, there were some outliers in the dataset like a vehicle being less than 15% of the image, images without any vehicles or images without any vehicle at all as shown in Figure 3.2. These outliners would have been deleted in pre-processing stage however that was not a primary issue of this dataset. The big advantage of the combined dataset is that number of images is relatively balanced, 3000 images in a training set is more than enough to achieve satisfactory accuracy of the model.

The open images dataset was taken from Open Images Dataset V7 made by Google. We can observe that the imbalance in the number of images is greater than in the combined dataset, thus it required more pre-processing and implementation of class weights calculation. This dataset is used for benchmarking various artificial intelligence models like object detection, classification, and segmentation it contains over 1000 classes, and it was used in ILSVRC 2010 and 2012 challenges [16]. Despite the different number of images for each class, this dataset has a lot of advantages. All of these images have been taken in real-life scenarios, without images on a plain background, the vehicles are seen from different angles and reduced the number of outliners. Many researchers have proven that is one of the best collections of data for classification algorithms as it has standardised types of images which were picked with precision. It also provides a diversity of images where the object's appearance changes quite often to provide a wide area of training [18].

Figure 3.3: Examples of images from the Open Images dataset

Figure 3.3 shows examples of images from car, truck, and motorbike classes. All of them are in their natural environment and the vehicles take up most of the space in the image, furthermore, another important factor is that the motorbike class contains both just motorcycles and motorcycles with people on them.

3.2.1.2 Dataset analysis

To properly analyse and pick the best dataset for the task, the research problem's specification and potential environment should be discussed. The efficiency is an important factor in the implementation of the toll booth system, as the classification model will not be fed with live footage but with snapshots of the vehicle from the image from the camera processed by other algorithms.

So object detection program detects when a vehicle has stopped then takes a snapshot of the vehicle and passes it to the model. Naturally, the best training set should be the most similar to a snapshot taken by the camera at a toll booth. We should take into account that the snapshot will be cut from the image thus the environment should be barely visible and the vehicle should take most pixels of the snapshot. To summarize the best training set would be with images of vehicles on the roads or in the cities in heavy or light traffic, the dataset shall not contain images with plain backgrounds unless the image is perfectly cut to the vehicle's borders.

3.2.1.3 Data preparation

Dataset's classes were split into training and test sets where the training set took 80% of the images and the test set remained 20%. Both datasets were imbalanced, therefore class weights were calculated and passed into model training. To be able to evaluate which dataset is a better choice to solve the classification problem, before training models, the same data pre-processing was performed on both datasets. Because all images in both datasets had different dimensions, downscaling was performed before training a model. Both training and test sets were downscaled to 224x224 with anti-aliasing enabled to avoid aliasing artifacts. After rescaling the data, some images appeared to be corrupted, so a script was created to remove them. If an image could not be opened and loaded by the PIL library that meant it was corrupted and thus removed from the dataset.

3.2.1.4 Choosing the best dataset

In this case of the research when there are two datasets which both have the potential to solve the problem with great results, before fine-tuning the models and finding the best one, a better dataset had to be chosen. The datasets will be compared based on the results of training 2 models on them, VGG16 and VGG19 [16] It is worth mentioning that all models will be tested on the same collection of images.

VGG16	Motorcycle	Car	Truck	Bus	Accuracy	Value accuracy
Combined dataset	97%	70%	0.1%	7%	75%	79%
Open Images dataset	79%	68%	22%	70%	74%	67%

VGG19	Motorcycle	Car	Truck	Bus	Accuracy	Value accuracy
Combined dataset	99%	80%	6%	3%	73%	75%
Open Images dataset	52%	85%	10%	57%	70%	65%

Table 3.3: VGG16 and VGG19 results on combined and open images datasets.

Based on the results, the dataset on which the trained models had higher overall accuracy and per-class accuracy will be picked for further development and fine-tuning. The training will take 20 epochs with an input size of 128x128 and batch size of 32, the best model will be picked based on value accuracy and per-class accuracy.

Based on the data shown in Table 3.3 it can be concluded that all of the models trained on the combined dataset are prone to overfitting towards cars and motorcycles as there is a huge difference in accuracy between cars and motorcycles to trucks and buses. The differences between the light vehicles images and heavy vehicles images had an impact on the models' ability to generalize images.

The Open Images dataset provided a better foundation for training the same models as the dataset consists of high-quality images, with variety in the vehicle position, camera angle, environment and background [18]. Even though there are some imbalances in terms of per-class accuracy, the models are not that exposed to overfitting like in the combined dataset case. These imbalances can be easily minimised with fine-tuning, oversampling, under sampling, adding dropout layers or increasing the number of epochs.

Summarizing the results of both datasets, the Open Images dataset was chosen for further development and fine-tuning. However, later in the fine-tuning process of the model, good samples picked from the combined dataset may be used to improve the model's accuracy.

3.2.2 Model design

Tested models will be analysed and evaluated. The best model will be chosen based on the value accuracy and per-class accuracy, the models were trained with the same input size, but different batch sizes and various layers added in case of transfer learning. At the beginning of the research, a convolutional neural networks model was created from scratch, then with an application of transfer learning various pre-trained models on the Open Images dataset were trained, evaluated, and compared to each other.

3.2.2.1 Input, data augmentation and Pre-processing

As discussed previously, firstly the images were scaled to 224x224 as this is the input for most pre-trained models, however, due to hardware limitations the actual input size will be smaller. Data augmentation is activated in the training process by creating Image Data Generator from TensorFlow.

Figure 3.4: Examples of images after performing augmentation.

Figure 3.4 presents data augmentation performed before training the model, the data is rescaled, flipped vertically or horizontally, and rotated, and shear, zoom and channel range are applied. Data augmentation can reduce overfitting and make the model learn more valuable features [6] [17].

3.2.2.2 Models' architecture

Components will be discussed which are used to build the model from scratch and to expand pre-trained models. A brief overview of these layers used will be provided, and in implementation, the chapter on the whole model architecture and its performance will be explained in depth.

3.2.2.3 Convolutional Layers

Convolutions are widely used in computer vision, especially in image processing. Convolutional neural networks use filters to detect features on an image, like edges, shapes and anything that may differentiate this image from another. Filters convolve with the image and create an activation map, The filter goes through the image and the dot product is calculated between every element of the image and an input [1]. The activation maps are generated for every element of the input image. As an output, there are stacked activation maps of every filter along the image depth. Finally, each neuron is connected to a small local piece of an image whose size is equal to the kernel size. When the convolutional layers are stacked one after the other, the top ones extract low-level features and the later ones high -level features [1,11].

3.2.2.4 Max Pooling

After each convolutional layer, a pooling layer is needed. Pooling layers consolidate the features learned by convolutional neural networks and shrink their spatial dimension to minimize the number of parameters and computations in the network. These layers are crucial as they always extract the important information from the feature map, even when the features are shifted or ambiguous. If an image is modified by some small value pooling layer is there, to still be able to read or extract the feature contained in the feature map. There are two types of pooling, max pooling and average pooling [1, 11]

Max pooling selects the most important feature which is the maximum value of the defined region whereas average pooling simply calculates the average values of the feature map. The output of a max pooling will be a sharpened image when the average pooling smoothes the image. In most cases in this research project, max pooling will be used as it is crucial to extract the most distinctive features of the different vehicle types such as the shape and size of the wheels, and the overall shape of the body [1, 11].

3.2.2.5 Dense

A dense layer is a fully connected layer where all of the neurons are connected to the input values and to all neurons of the other layer. In order to create a dense layer units have to be specific which is the number of neurons and also the size of the output from that layer. There is also an activation function which applies non-linearity to the network, so the network can learn the relationship between input and output values. There are many activation functions however ReLu will be used for adding additional dense layers to transfer learning and building models from scratch and Softmax as the last layer of 4 units as there are 4 classes to classify images into [1, 11].

3.2.2.6 Dropout

This layer refers to the dropping nodes of an input and hidden layer. It randomly sets input units to 0 with a frequency specified in the rate argument at each step during the training process [11]. It is a great tool to prevent overfitting of the model on the training data, often used after full connection layers [11, 1].

3.2.3 Convolutional Neural Network Architecture

One of the models was created from scratch without any pre-trained base applied to it. The model will consist of convolutional layers and a 2D max pooling layer after each of them. The activation function for all of the layers will be ReLu and kernel size 3. The model has at least 4 convolutional layers and 1 dense layer, however these number change due to fine-tuning and finding the best model.

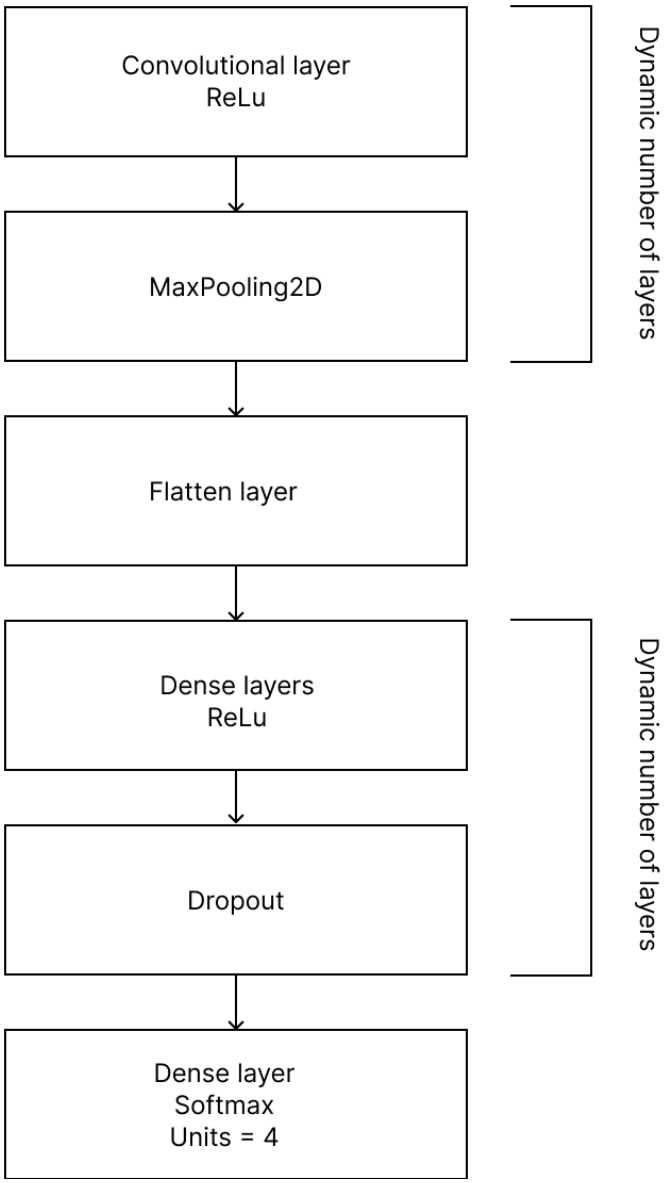


Figure 3.5: Base CNN model architecture

Finding the best model has started from the base architecture as presented in Figure 3.5, where a number of convolutional layers with max pooling and dense layers are yet to be determined based on the model's results. If the model shows a promising outcome the dropout layers will be added to maximize model performance and minimise overfitting.

3.2.3.1 Transfer Learning and Pre-trained Models

To find the best classification model in an efficient way, pre-trained models will be used and compared to the original convolutional model. Based on existing literature and research papers four models have been widely used in image classification and proven their effectiveness after fitting them to the problem. These are:

- VGG16 [16]
- VGG19 [16]
- ResNet50 [18]

3.2.3.2 VGG16 and VGG19

VGG architecture is a successor of AlexNet but was created by the Visual Geometry Group at Oxford University. Both of these architectures are similar to each other the main difference is that VGG16 has 16 layers and VGG19 has 19 layers [11, 16]. VGG19 consists of 16 convolutional layers and 3 fully connected layers, kernel size is 3x3 with a stride size of 1 pixel enabling the model to cover the whole image. Max pooling is performed over a 2x2 pixel window with a stride size of 2. These layers are followed by the ReLu activation function, at the end there are 3 fully connected layers where two of them from the top having 4096 units and the third one with 1000 units. The VGG16 architecture uses a similar setup but it has 13 convolutional layers and 3 full connection layers [11,16].

3.2.3.3 ResNet50

ResNet50 is a variant of the ResNet model, it has 48 convolution layers with 1 max pool and average pool layer and it uses an interesting technique called residual learning. Residual learning was a solution to a common problem called vanishing gradient problems [18]. Residual connection lets the model “skip” layers of the network, this enables the model to learn the residual mapping which is the difference between the desired output and the input to a block of layers. By using this technique ResNet architecture can learn deeper and more complex features without suffering from vanishing gradient problems. However, this model has a complex architecture and has been trained on 12 million images from 1000 classes, thus the model may be prone to overfitting as the classification problem of this research is much simpler than the initial one.

3.2.3.4 Training, fine-tuning and Picking the Best Model

All of the theses 4 models will be trained with hyperparameter tuning to achieve the best accuracy and most importantly per-class accuracy. As the problem requires similar accuracy across all classes that will be the most desirable result. Good accuracy across all classes will be chosen over any dominance in the accuracy of one class over another. As the objective is to create a functional piece of software that is the crucial value of the final solution.

4 Implementation

4.1 Tools and Libraries

The IDE used for this project is PyCharm Professional with a scientific view enabled. The project environment is set up based on an Anaconda environment with extra libraries installed like OpenCV to process images. PyQt5 version of QT Framework was used as it was pre-installed with the Anaconda environment, it was not updated for PyQt6 as it is more than enough for the graphical user interface implemented in this project.

4.2 Data Pre-processing

In the project implementation, there is a DataProcessing class designated strictly for data processing, modification, and preparation. Data augmentation is performed in the model class as it uses the TensorFlow library, and it is done just before training. As the data preparation is usually done once before the training, the file where that class is placed is designed to be executed as a single script. When the script is executed in the terminal the user will be asked for the names of the folders of the datasets and the size to which the images should be rescaled.

4.2.1 Scaling images

It contains methods for scaling datasets, scaling specific folders and finding corrupted images in the folder. When the DataProcessing class is created, the dataset folder path is passed to the constructor as an argument, all the methods then refer to that folder to perform operations on the files.

```
def scale_dataset(self):
    for dataset in [self.training_set_folder_name, self.test_set_folder_name]:
        scaled_dataset = self.dataset_folder + sep + dataset + "_scaled"
        mkdir(scaled_dataset)
        for folder in listdir(self.dataset_folder + sep + dataset):
            self.__scale_images_in_folder(self.dataset_folder + sep + folder, scaled_dataset + sep + folder)
```

Figure 4.1: Data pre-processing class method to scale the dataset to desired image size.

```
    def __scale_images_in_folder(self, folder_path, new_folder_path):
        mkdir(new_folder_path)
        file_counter = 0
        for file in listdir(folder_path):
            try:
                img = imread(folder_path + sep + file)

                res = resize(img, (self.img_width, self.img_height), anti_aliasing=True)
                file = file.replace(" ", "-")
                print(file)
                imsave(new_folder_path + sep + file, img_as_float(res))
                file_counter += 0
                print(str(file_counter) + ' images processed')
            except IOError:
                print("Cannot read the file, image will not be scaled")
                print("Continue...")
                continue
```

Figure 4.2: Private method of the Data pre-processing class to scale images in the folder

Figure 4.1 presents a public method to scale all images in the dataset with the path passed when creating an instance of the class, then Figure 4.2 shows an actual scaling operation. When any input or output error occurs, the user is informed, and the script keeps going.

4.2.2 Data Augmentation

Augmentation of the input data is performed with the help of the TensorFlow functions which create data gen with specified modifications applied to the images and with other functions load images from the folder.

```
self.train_data_gen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.3,
    zoom_range=0.3,
    horizontal_flip=True,
    vertical_flip=True,
    channel_shift_range=0.3,
    rotation_range=30,
)
```

Figure 4.3: Data Augmentation with Image Data Generator

Figure 4.3 presents the creation of a training set generator which applies various augmentation techniques to the training data. An image can be modified in six different ways, zooming, shifting channel, applying shear range, flipping both horizontally and vertically and rotating. The test data generator only must rescale the image provided as the images in the test set should not be modified.

```
self.training_set = self.train_data_gen.flow_from_directory(
    'dataset/training_set',
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical'
)

self.test_set_data_gen = ImageDataGenerator(rescale=1. / 255)

self.test_set = self.test_set_data_gen.flow_from_directory(
    'dataset/test_set',
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical'
)
```

Figure 4.4: Filling the data generator using the flow from the directory function

As presented in Figure 4.4 dataset variables are created with flow from the directory method of the image data generator [11]. The folder of the dataset is passed as an argument and class mode which in this case is categorical as there are more than 2 classes.

4.3 Implementation of Vehicle Type Classification Model

Implementation of all of the classification models is done by creating Model and BaseModel classes. The base model class contains a pure implementation of model layers with input specified when creating an instance of the class. In the case of pre-trained models, it also takes care of freezing certain layers. The Model class has methods for training the model, loading the dataset, and methods required for GUI to use to make predictions on dynamic sets of images. The Model class also takes care of saving model versions per epoch, calculating class weights, and adding the last SoftMax layer with 4 units to all of the models taken from BaseModel class. The common hyperparameters are specified and held by the BaseModel class, however, as each model was separately trained and fine-tuned, they had to be in separate methods. The Model class requires the base model to provide a get function to return a sequential model with all of the layers except the final softmax layers with the same number of units as classes which is 4. The Model class provides a boilerplate for the model while the BaseModel provides the layers of the model.

4.3.1 Convolutional Neural Network from Scratch

The convolutional neural network implementation is held in BaseModel just like transfer learning models. The model has a separate method where all the layers are created, as the model boilerplate is in the Model class so that the last softmax layer can be omitted. The method returns the Sequential model of the convolutional neural network [11].

```
def get_cnn(self):
    model = tf.keras.models.Sequential()
    # Convolutional Neural Network built from scratch
    model.add(tf.keras.layers.Input(shape=self.input_shape))
    model.add(tf.keras.layers.Conv2D(filters=128, kernel_size=3, activation='relu'))
    model.add(tf.keras.layers.MaxPool2D(pool_size=1, strides=1))

    model.add(tf.keras.layers.Conv2D(filters=128, kernel_size=3, activation='relu'))
    model.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

    model.add(tf.keras.layers.Conv2D(filters=256, kernel_size=3, activation='relu'))
    model.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

    model.add(tf.keras.layers.Conv2D(filters=256, kernel_size=3, activation='relu'))
    model.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

    model.add(tf.keras.layers.Conv2D(filters=512, kernel_size=3, activation='relu'))
    model.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

    model.add(tf.keras.layers.Flatten())

    model.add(tf.keras.layers.Dense(units=1024, activation='relu'))
    model.add(tf.keras.layers.Dense(units=512, activation='relu'))

    return model, "CNN"
```

Figure 4.5: An implementation of Convolutional Neural Network from scratch.

Convolutional layers with kernel size 3x3 are added to the sequential object first as presented in Figure 4.5. The first convolutional layer has a pool and stride size of 1x1 to cover the whole image. Between the convolutional and full connection layers, there is a flattened layer to convert the multidimensional output into one-dimensional input to the full connection layer [11]. There are full connection layers in this case of the model. After fine-tuning and layer modification the best layer will be chosen based on the per-class accuracy and value accuracy.

4.3.2 Transfer Learning

The transfer learning models trained in this research are much different from each other, thus each of them will have different hyperparameters and the fine-tuning will look different. The models have separate methods of BaseModel class where they are initialised and developed.

```
def get_vgg16(self):
    model = tf.keras.models.Sequential()

    # Transfer learning model
    base_model = tf.keras.applications.VGG16(input_tensor=self.image_input, input_shape=self.input_shape,
                                             weights='imagenet', include_top=False)
    # Freeze last 3 layers
    for layer in base_model.layers[-3:]:
        layer.trainable = False

    model.add(base_model)

    # Extra layers added at the end of the model
    model.add(tf.keras.layers.GlobalMaxPool2D())
    model.add(tf.keras.layers.Dense(units=512, activation='relu'))
    model.add(tf.keras.layers.Dropout(0.2))

    return model, "VGG16"
```

Figure 4.6: An example of getting the VGG16 model from BaseModel

Figure 4.6 shows how the transfer learning models are implemented, in this case, the VGG16 model is imported from the Keras library. After the base model is included, the model is expanded by adding more layers and fine-tuning the hyperparameters. The loop goes through specific layers and freezes the weights, the number of frozen layers depends on the model. Finding the best model will focus on adding and modifying extra layers at the end of the model and evaluating it based on the per-class accuracy and value accuracy.

4.3.3 Data Balancing

The dataset chosen to be the best of the 2 at the beginning has data imbalance. The car class has significantly more images than other classes, the bus and truck class have only ⅓ and the motorcycle class has ½ the number of images of the car class. Thus, preventing imbalanced training of the model and overfitting, the weights of the classes will be calculated.

```
# Class weights
class_weights = class_weight.compute_class_weight('balanced', classes=np.unique(self.training_set.classes),
                                                  y=self.training_set.classes)
class_weights = {k: v for k, v in enumerate(class_weights)}
```

Figure 4.7: Class weight calculation

Figure 4.7 presents the calculation of class weights with the use of compute class weight function from the sklearn library [20]. As the class weights should be balanced the calculations are performed based on this equation:

$$n\_samples / (n\_classes * np.bincount(y))$$

The bincount function from the NumPy simply counts occurrences of each value in an array, in this case, y [11, 20].

Class name	*Weight*
Bus	*1.63*
Car	*0.51*
Motorcycle	*1.16*
Truck	*1.71*

Table 4.1: Weights of classes

Table 4.1 shows the calculated weights for each class. After the classes are calculated and put into an object with corresponding indexes, the weights are passed into the fit method as a class weight argument.

4.3.4 Saving Model Results

To keep track of the progress of the model development all the epochs of the training session are saved into a folder with a unique name. The name of the model trained at the epoch has all the necessary information like accuracy, loss, value loss and value accuracy. The version of the model is saved with a callback function from the Keras library called ModelCheckpoint [11].

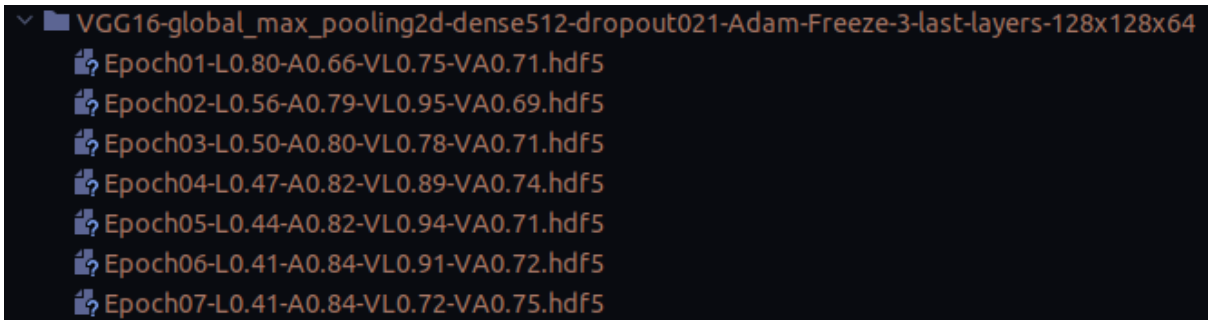


Figure 4.8: Folder and file structure of a VGG16 model with extra layers

As presented in Figure 4.8, models per epoch are saved in one folder corresponding to the model trained. The reason for that is it is impossible to evaluate per-class accuracy without running a script doing so as sometimes values accuracy could be over 75% when per-class accuracy results would suggest overfitting to some of the classes.

4.3.5 Model Presentation in Graphical User Interface

To present the results of the model and its accuracy a graphical user interface has been created. The user can put images into the folder which will be displayed in the GUI later with the model's guesses. In the project, there is a manual_test folder from which images for the presentation will be loaded. However, the name of the folder is not static it is passed as an argument to the GUI class constructor. The GUI class handles all logic related to the user interface. The model object is passed to the GUI constructor and used later in making predictions. The application goes one by one through all images in the folder displaying them in the application window and predictions for each class, which means that the model will pick the highest probability as the final prediction. Each image is displayed for 2.5 seconds, the interval is set with QTimer. Every certain amount of time the set image method is executed which loads another image passes it to the model and gets the values of the prediction, at the end the image is set in the UI window and progress bars values are set based on the result from the model single prediction.

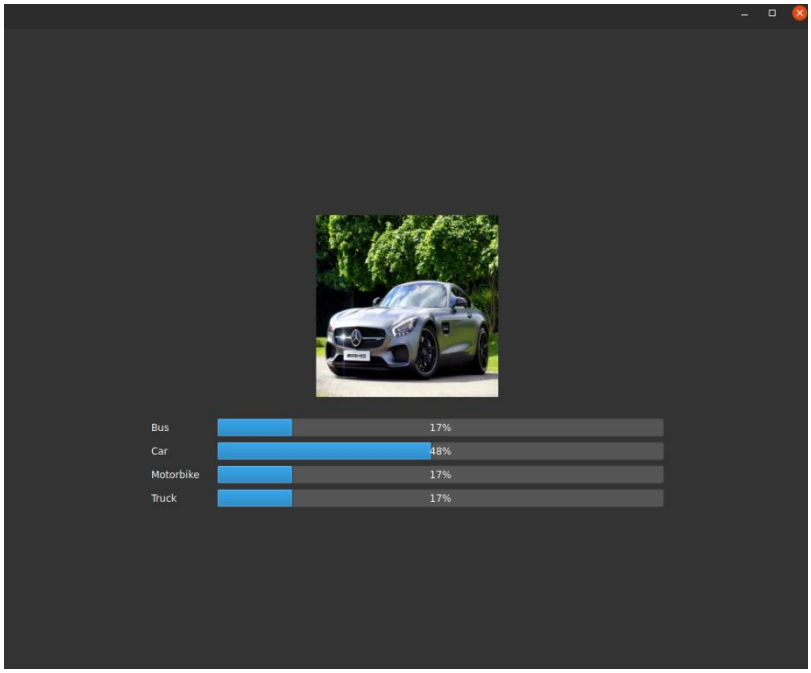


Figure 4.9: Graphical User Interface

As presented in Figure 4.9 the image is displayed in the middle of the window with progress bars symbolizing probabilities for each class.

```python
def change_image(self):
    # Exit program if finished
    if self.image_counter >= len(self.images):
        self.image_change_timer.stop()
        sys.exit()
    # Get predictions
    predictions = model.predict_single(self.images[self.image_counter])
    # Display results
    self.set_progress_bars(predictions)
    pixmap = QPixmap(self.test_image_path + self.images[self.image_counter])
    pixmap = pixmap.scaledToWidth(self.image_height)
    self.image_label.setPixmap(pixmap)
    self.image_counter += 1
```

Figure 4.10: Change the image method of the GUI class.

Change image method as presented in Figure 4.10 swaps image in the GUI window and gets predictions from the model. The library used for the application is PyQt5 as this version is part of the anaconda environment and the graphical user interface is not complex.

5 Results

This chapter presents and discusses the results of the research. All the models will be compared and evaluated, and a conclusion will be drawn why some of them would be a good choice for further research and development. The final training was performed with an input size of 128x128x3 with a batch size of 32, because of the hardware limitations, model complexity and problem specification. Anything over that would not be possible on the machine used in this research [11]. Furthermore, if we analyse the possible use cases of this model in the real world like a highway toll booth, it is unlikely that the snapshots of the vehicles from the image would be bigger than that size. However, it is highly dependent on the camera's resolution and quality.

5.1 Vehicle Type Classification with CNN from Scratch

A convolutional neural network was created from scratch to compare it with existing pre-trained models. As it is known the pre-trained models were trained and developed by companies and many AI specialists, thus it is interesting how close can a model built from scratch by one person get to the results of the well-known transfer learning models.

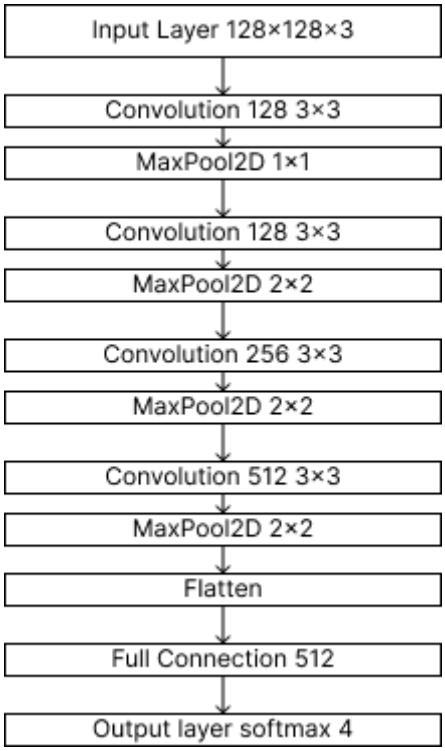


Figure 5.1: CNN Architecture

Figure 5.1 presents the architecture of the final convolutional neural network model, there are 4 convolutional layers and one full connection layer with 512 units. The first max pool layer has stride 1x1 to cover the whole image, then the convolutional filters increase to gather deeper features of the image. The kernel size stays 3x3 for all the convolutional layers.

Best CNN Model			
Accuracy	**Loss**	**Value Accuracy**	**Value Loss**
76%	0.6	63%	1.09
Bus	**Car**	**Motorcycle**	**Truck**
55%	62%	80%	52%

Table 5.1: Convolutional Neural Network Results

The CNN model which results are presented in the Table 5.1 was trained for 17 epochs. Based on the accuracy and loss the model was a decent fit and the training was going in the right direction, however looking at the value accuracy and the value loss it can be concluded that the model performed better on some classes than the on the others, which proves the per-class accuracy statistics and the disbalance between Motorcycles accuracy and the rest of the classes. Overall, the architecture of the convolutional neural network created from scratch in this research proved to be a good fit and it can certainly be developed and fine-tuned further to achieve better results. For

now, the model is quite shallow thus adding more layers both convolutional and full connection with appropriate parameters may increase the accuracy significantly.

5.2 Vehicle Type Classification with Transfer Learning

Three pre-trained architectures VGG16, VGG19 and ResNet50 were trained, fine-tuned, and compared to find the best model for the presented problem, the major factor for picking the best model is per-class accuracy. The weights were downloaded from the Keras library and were obtained on trained on an ImageNet dataset; however, the models will be fine-tuned, extended by adding extra layers and some layers from the base architecture might be frozen.

5.2.1 VGG16 and VGG19

Both VGG16 [16] and VGG19 [16] models have similar architectures however VGG19 have three extra convolutional layers. Their results were quite close to each other, and they presented the best accuracy of all the models trained. The achieved per-class accuracy was enough for it to be implemented in an actual system.

VGG16 Model			
Accuracy	**Loss**	**Value Accuracy**	**Value Loss**
82%	0.45	76%	0.64
Bus	**Car**	**Motorcycle**	**Truck**
76%	74%	90%	60%

Table 5.2: VGG16 Results

The results of the best VGG16 model found are presented in Table 5.2. All of the classes have over 60% accuracy which was the minimum goal in this research. With about 75% accuracy in Bus and Car and over 90% in Motorcycles. The model is better than CNN, especially if we analyse the balance of per-class accuracy. The base of the VGG16 model, was extended by the global max pooling 2D layer and dense layer with 512 units, additionally a dropout layer was added with 0.2 dropout rate to prevent overfitting and make the model more robust. Furthermore, the weights of the last 3 layers of the model layers were frozen. The model presented a good generalization of the images, and the worst accuracy was achieved for the Car and the Truck class, however theses accuracies are still good enough to make the software operational.

.

VGG19 Model			
Accuracy	**Loss**	**Value Accuracy**	**Value Loss**
84%	0.41	74%	0.83
Bus	**Car**	**Motorcycle**	**Truck**
60%	80%	85%	70%

Table 5.3: VGG19 Results

VGG19 has three more convolutional layers than VGG16 which may make recognizing deeper features easier for the model, thus, in this case, the mistake between Bus and Truck should be lower. Table 5.3 shows the results of the VGG19 model which were achieved by adding global max pooling 2D and dropout layer with the rate of 0.2 at the end of the model. Same as in the VGG16, weights of the last 3 layers were frozen. Despite the 2% lower overall accuracy compared to the VGG16 model, the present model holds greater appeal as a problem-solving approach due to its higher per-class accuracy in the most frequently occurring vehicles which are cars. In particular, the VGG19 model is better suited for applications in toll gates on highways or other locations where charges are levied. The per-class accuracy for cars is 6% higher than VGG16, although there is a noteworthy decline in bus and motorcycle accuracy, the accuracies of trucks and cars are slightly higher. Even though the average per-class accuracy is lower, this model would provide more satisfying results when implemented in a real system, as it has better balance across per-class accuracies and performs better in predicting cars accurately.

5.2.2 ResNet50

ResNet50 [19] is one of the best-performing models on the ImageNet benchmark, it achieves over 90% accuracy on 1000 classes of images. It has 50 layers and uses residual learning to improve its accuracy and be more resistant to overfitting.

ResNet50 Results			
Accuracy	**Loss**	**Value Accuracy**	**Value Loss**
79%	0.79	64%	1.02
Bus	**Car**	**Motorcycle**	**Truck**
62%	72%	88%	55%

Table 5.4: ResNet50 Results

Table 5.4 presents the results achieved by fine-tuning the ResNet50 model, in this model last 3 layers were frozen, max pooling was applied and Adam optimizer was used. In the process of finding a better-performing model, different layers were set to trainable or frozen, more dense layers were added with dropout layers, and the optimizer was changed to SGD with a 0.001 learning rate. The results show that the model achieve good accuracy on Motorcycles and Cars, however the Truck accuracy is less than 60% and Bus just over that value. The lowest accuracy achieved by VGG16 and VGG19 was 60% thus these two are much more desirable than this model. The average accuracy is also lower than both VGGs' architectures.

5.3 Models Summary and Picking the Best Model

One of the four models trained and fine-tuned will be chosen as the final implementation. The most important factor is per-class accuracy which will allow the system to work properly.

#	VGG16	VGG19	ResNet50	CNN
Accuracy	82%	84%	79%	76%
Value Accuracy	76%	74%	65%	63%

Table 5.5: VGG16, VGG19, ResNet50 and CNN final comparison

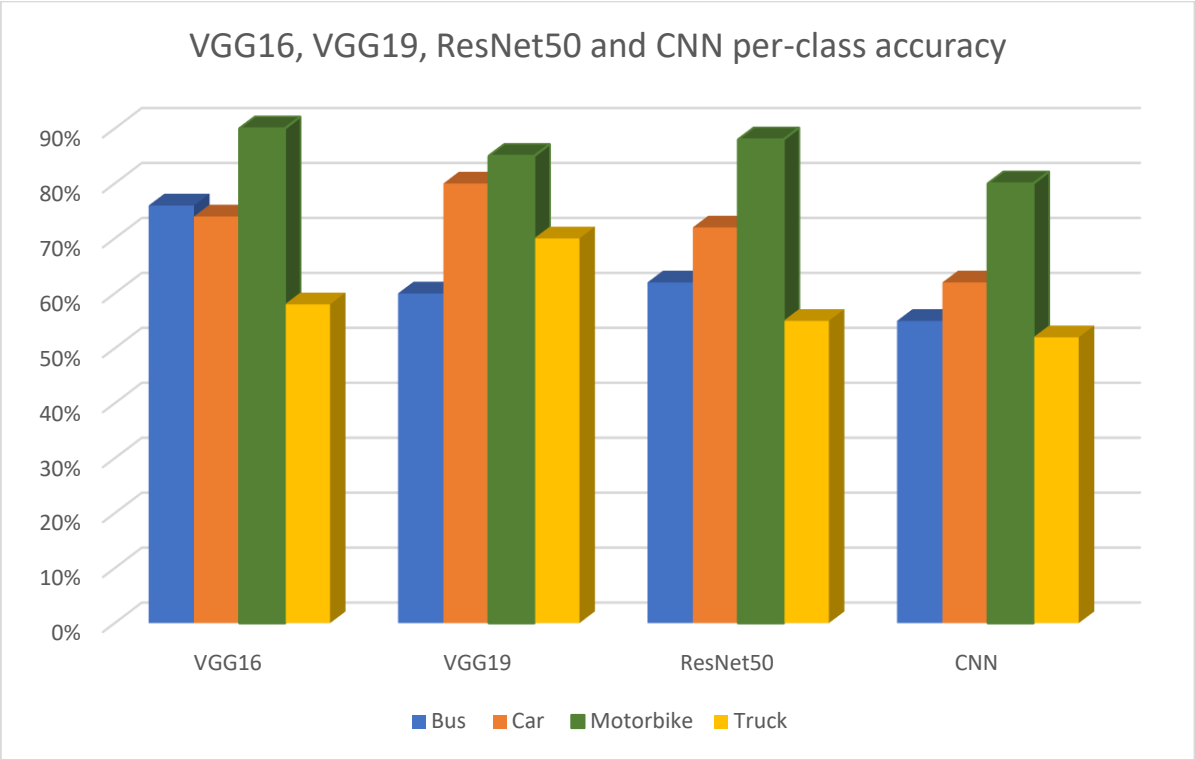


Figure 5.2: VGG16, VGG19, ResNet50 and CNN per-class accuracy

Based on data shown in Table 5.5 picking the best model would be a difficult choice, however in the case of this study the most important statistics is per-class accuracy of which graph is shown in Figure 5.2. The accuracy achieved by VGG16 and VGG19 models are similar and both satisfying, they both achieved over 60% on all classes and presented outstanding performance in predicting motorcycles and cars. However, the results of the VGG19 model are more balanced, and the car accuracy is

slightly higher than in the VGG16. The car accuracy is important as these are the most common occurring vehicle on the roads, thus the accuracy should be the highest possible when keeping balance in the accuracy of other classes. ResNet50 did not manage to achieve over 60% in the Truck class, additionally the Bus is 62% which is just slightly over 60 and just 72% in the Car class. ResNet50 has performed relatively well in identifying motorbikes, with an accuracy of 88%. However, it has lower accuracies compared to VGG16 and VGG19 for identifying buses, cars, and trucks. This may be because ResNet50 uses residual connections to allow for easier training of deep neural networks. These connections help to mitigate the vanishing gradient problem, but they may also lead to overfitting if not properly regularized. The ResNet50 average per class accuracy is lower than both VGG16 and VGG19, however with a different approach to the fine-tuning and experimenting with freezing different residual block or adding more dense layers the results could be improved. Furthermore, the ResNet50 is the deepest models of all trained thus this it might be more prone to overfitting as in this problem there are only 4 classes. The ResNet50 might simply learns too deep, and it may struggle with generalisation. The convolutional neural networks created from scratch in this project presents the worst results, however it is not far from the other models, especially from the ResNet50. Considering that the CNN model only has 4 convolutional layers and 1 full connection layers it is expected to achieve lower accuracy, however it has huge potential to be good at solving this problem, but it is a time-consuming work nevertheless it may pay off later. The best model for this research problem is VGG19, it has great per-class accuracy balance with all the scoring over 60% and a high score in predicting cars and motorcycles.

5.4 Comparison To Existing Literature

It is difficult to find literature where pre-trained models are used for simpler classification problems, where generalisation plays the biggest role. Most of the existing literature focus on very similar classes like flowers, animals, car models and objects from different general categories or just classification problems with a lot of classes [7]. The pre-trained models that were used in this research or slightly modified versions of them can be seen in many papers where they achieved accuracy easily over 90%, like in a fine-grained classification of vehicles [7], where exact models are found which requires a lot of small feature precision. It is worth mentioning that a lot of researchers use the Stanford Car or CompCars datasets [5, 6, 7] which are not publicly available and are one of the best vehicle datasets available overall. As said “Dataset. In deep learning-based classification systems, dataset is a key input that helps the algorithms learn the features to perform predictions based on the learned information.” [6] thus a dataset may change the results of the models significantly. In a research article about CNN-based vehicle classification [7], there are only 6 classes of vehicles, and models like VGG, Inception and ResNet50 are used, however, they are trained on initial input of the pre-trained models which is 224x224 and combined datasets mentioned above. Finally, the author adds another classification block of layers at the end to improve the models. Models trained in this project can be fine-tuned to achieve similar results as in the other literature, but more powerful hardware must be used, and it would be required to gather more high-quality data. It can be seen how important the dataset is based on Chapter 3 where two datasets are compared and the one with fewer images turns out to be a better fit. It is not said that deeper models cannot solve simpler problems, but the process of fine-tuning would be different, and the dataset would have to fit the problem that we are trying to solve.

6 Conclusion

It can be concluded that vehicle type detection, vehicle detection and vehicle recognition problems can be solved with quite good results, the actual problem is solving a specific classification problem in a certain environment with publicly available data. M. Atif Butt et al. were able to compare existing pre-trained models and apply fine-tuning to the best one, finally getting an unbelievable accuracy of 99.68% based on the dataset they gathered [6]. There are many more ways of achieving satisfactory accuracy on similar problems, like Z. Xaze et al. who created a model solving the same problem using reinforcement learning and error and error-prone samples, achieving 98.33% accuracy after applying their improvements to the model. We can encounter many ways and models to solve this problem in research papers or articles like improving the YOLOv2 model by J. Sang et .al [9] or using FasterR-CNN with ResNet [7] and they all have high accuracy. Thus, we can state that vehicle detection or vehicle type detection is not a problem anymore as it was solved, tested, and proven in many papers. However, even though the problem is easily solvable with existing methods and resources there is an issue which we can find in every single paper related to vehicle detection which is data. X. Ke, Y. Zhang et al. [7] were forced to gather data in an automated manner using existing models which can do so, but it did not fully resolve the problem as the data comes from different datasets [7]. J. Sang et al. [9] and M. Atif Butt [6] collected data from traffic surveillance cameras on which the models performed well but if the model were migrated to a real-time application the result could be different as the images were not unbalanced as vehicles were from different regions and constructed with modernity, design, and technology. We can certainly say that there is not a single dataset which would provide at least 4 basic vehicle types like a car, bus, truck, and motorbike with a wide variety of vehicles from different parts of the world. Therefore, if there is a lack of data in this area of research the development of models will take longer as researchers have to spend a good amount of time figuring out collecting the data and how to gather it in a way to have no impact on the model and its accuracy. Additionally, previous work is focused on many classes with more deep feature detection where classes are like each other, thus fitting the pre-trained models to the simpler problems is not as easy as it seems. The more complex models easily overfit when trained on fewer classes, however, a lot depends on a specific problem and its environment, datasets used and other values that are specific to the problem that the model is trained for.

6.1 Summary

The main objective of this research was to find a model which classified vehicle images into four classes, Truck, Car, Motorbike, and Bus. Two approaches were taken, creating a convolutional neural network from scratch, and using three pre-trained models with transfer learning, ResNet50, VGG16 and VGG19. The first challenge was to pick a better dataset, where one was a combined dataset of two others and the other one was taken from the Open Images collection. Based on the results of existing pre-trained models without fine-tuning the Open Images dataset was chosen for further development. Data pre-processing and augmentation were the same for all the models to identify the best one for that specific problem and environment with available resources. Four models were trained and fine-tuned, convolutional neural network from scratch, ResNet50, VGG16 and VGG19. These models were chosen as they were used in many research papers in computer vision and the depth of them vary, which is a significant factor when trying to solve a classification problem. As it

turned out with a classification problem with 4 classes where generalization plays the main role, VGG16 and VGG19 which are less complex than ResNet50 achieved satisfying results even though they are shallower models. Generic convolutional neural network created from scratch had the worst score, however taking into account that it only had 5 convolutional and 1 dense and final softmax layer, the results are satisfying as all of the per-class accuracies were over 50% and the value accuracy raised to 63% when being only 13% away from the accuracy value. There is a potential for this model to be developed further by adding more layers. ResNet50 which was the deepest model, achieved slightly worse results than both VGGs' architectures which is an interesting outcome as ResNet50 was performing better than VGG architectures in most of research paper with complicated classification problems. Out of all 4 models VGG16 and VGG19 obtained the best results, with 76% and 74% accuracy and VGG19 model with over 60% per-class accuracy on all classes, which was the desired result. It was tested by finding 20 random vehicle images and using the implemented way of testing images which are put into a certain folder and the presentation with probabilities is displayed using the graphical user interface coded into the project. On average over 17 predictions were correct which proved that the model could solve this classification problem. Finally, the VGG19 was chosen as the best one considering its per-class accuracy, balance and average accuracy, the results presented by this model proved that it could be used in a real system, fulfilling all requirements. With over 60% accuracy on each class, it could certainly help in toll booth automation if integrated into a fully working system.

6.2 Future Research

With the development of transportation, self-driving cars and general applications of modern software development and artificial intelligence on the roads in many systems the need for creating new models for specific problems will grow. The primary objective of future research would be to find a better model than the one achieved in this project, ideally, the model which has over 80% per-class accuracy. It could be done by further fine-tuning the models created or exploring new models and neural networks. It would be a good idea to obtain better datasets like the ones not available publicly like Stanford Cars and CompCars which are used in many research projects [5, 6, 7]. Other types of neural networks could be explored like Region-Based Convolutional Neural Networks which varieties are used in object detection and classification or adding reinforcement learning to the existing models [7, 8]. Furthermore, to train more complex models and increase the input size cloud computing could be used to omit the hardware limitations and only focus on achieving the best accuracy and finding the best model. Also researching how models with different depth performs on different type of classification problems could produce interesting results. The ability of the models to find small differences between images or being as precise as possible could be compared alongside with ability to classify images based on high level features.

7 References

1. Jianxin Wu, May 2017, Introduction to Convolutional Neural Networks, Nanjing University, China
2. S. Albawi, T. A. Mohammed, S. Al-Zawi et al. August 2017, Understanding of a convolutional neural network
3. J. Gu, Z. Wang, J. Kuen et al. May 2017, Recent advances in convolutional neural networks
4. K. O'Shea, Ryan Nash, December 2015, An Introduction to Convolutional Neural Networks
5. L. Chen, F. Ye, Y. Ruan et al. October 2018, An algorithm for highway vehicle detection based on convolutional neural network
6. M. Atif Butt, A. Masood Khattak et al., December 2020 Convolutional Neural Network Based Vehicle Classification in Adverse Illuminous Conditions for Intelligent Transportation Systems
7. X. Ke, Y. Zhang Fine-grained vehicle type detection and recognition based on dense attention network
8. Z. Xaze et .al 2018 A CNN Vehicle Recognition Algorithm Based on Reinforcement Learning Error and Error-prone Samples
9. J. Sang, Z. Wu et al. An Improved YOLOv2 for Vehicle Detection
10. Albero Rizzoli March 2023, 9 Revolutionary AI Applications In Transportation
11. Chollet, F. & others, 2015. Keras documentation
12. M. Hashemi, B. Farahani, F. Firouzi et al. September 2020 Towards Safer Roads: A Deep Learning-Based Multimodal Fatigue Monitoring System
13. D. Tabernik, D. Skocaj et .al eep Learning for Large-Scale Traffic-Sign Detection and Recognition
14. A. Coskun Computer Vision Applications In Transportation
15. P. Chabra April 2022 How Computer Vision is Transforming the Global Transportation Industry?
16. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton ImageNet Classification with Deep Convolutional Neural Networks
17. Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in Proceedings of the AAAI, pp. 13001–13008, 2020.
18. J.Deng, W.Dong, R.Socher, et al. ImageNet: A large-scale hierarchical image database
19. K. He, X.Zhang, S.Ren, J.Sun, "Deep residual learning for image recognition" 2016
20. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011
21. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," 2016.
22. M.Tan, Q. Le "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks" May 2019