Deep Learning for Vehicle Speed Estimation and Number Plate Recognition using ChatGPT-Based Computer Vision Code Generation.

By

Patrick Uwayo (patrick.uwayo@aims.ac.rw)

African Institute for Mathematical Sciences (AIMS), Rwanda

Supervised by: Dr. rer. nat. habil. Abebe Geletu W. Selassie
German Research Chair, AIMS Rwanda
Co-Supervised by: Dr. Eunice Gandote
AIMS Rwanda

June 2023

AN ESSAY PRESENTED TO AIMS RWANDA IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF MASTER OF SCIENCE IN MATHEMATICAL SCIENCES

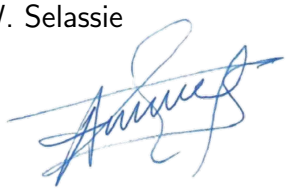
DECLARATION

This work was carried out at AIMS Rwanda in partial fulfilment of the requirements for a Master of Science Degree.

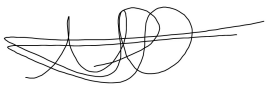
I hereby declare that except where due acknowledgement is made, this work has never been presented wholly or in part for the award of a degree at AIMS Rwanda or any other University.

Student: Patrick Uwayo

Supervisor: Dr. rer. nat. habil. Abebe Geletu W. Selassie



Co-Supervisor: Dr. Eunice Gandote



ACKNOWLEDGEMENTS

I am deeply thankful to Mr. Sam Yala, the Center President, and Prof. Blaise Tchapnda, the Academic Director of AIMS Rwanda and all AIMS tutors for their visionary leadership and commitment to creating a conducive learning environment have greatly influenced my educational journey. I am grateful to my supervisor, Dr. rer. nat. habil. Abebe Geletu W. Selassie, and co-supervisor, Dr. Eunice Gandote, for their exceptional guidance, support, and invaluable insights throughout the thesis process. Their expertise and advice have been instrumental in shaping the direction of my research.

I would like to express my appreciation to Dr. Joseph Ndenda, my personal tutor, for his valuable advice. His guidance has been pivotal in my academic and personal growth. I would also like to acknowledge the visiting lecturers who shared their knowledge and experience that enriched my learning experience at AIMS Rwanda.

Furthermore, I extend my gratitude to my fellow classmates for their fellowship and support. Our collaborative efforts and shared experiences have made my time at AIMS Rwanda memorable. Lastly, I want to express my thanks to all staff members and workers at AIMS Rwanda for their dedication and commitment to promote an environment that values and encourages high academic standards.

To everyone mentioned above and others who have supported me along this journey, I am sincerely grateful for your contributions.

Abstract

This project explores the application of deep learning and computer vision techniques for vehicle speed estimation and number plate recognition using ChatGPT-based computer vision code generation. The objective is to develop an efficient and accurate system for real-time speed estimation and number plate recognition in various scenarios. The object detection/classification and Optical Character Recognition (OCR) deep learning algorithms as well as the object speed estimation algorithm using optical flow were applied on a video stream recorded by the camera together with edge computing. The deep learning models and computer vision frameworks are built and accessed under the python infrastructures, numpy, opencv, pytorch, etec. We used Chinese City Parking Dataset (CCPD) and Licence plate digits classification dataset, and they showed good performance in training and testing the models. The integration of all the algorithms into a complete system, along with the utilization of edge computing, is demonstrated. This research contributes to the field of computer vision, deep learning and road traffic management system by presenting a framework for efficient and accurate vehicle speed estimation and automated number plate recognition. The results demonstrate promising applications in intelligent transportation systems, paving the way for further research and practical implementations.

Keywords: Deep learning, Computer vision, Object detection/classification, Optical flow, Vehicle speed estimation, Number plate recognition, ChatGPT, Edge computing.

Contents

List of Figures

1. Introduction

As the volume of vehicles in the roads has been increasing on high rate, the traffic management became a great challenge. Nowadays, road traffic accidents is a real public issue all over the world for all concerned agencies to deal with. The accurate estimation of vehicle speed and number plate recognition are considered as crucial tasks in traffic management and road accident prevention.

There are different technologies and devices used through out the history for estimating vehicle speed. For instance doppler effect was mostly used by using radar (radio detection and ranging) and/or lidar (light detection and ranging) devices. With the development of technology, the use of cameras for that task was employed. The classical method of using radar and/or lidar devices, showed inconvenience in terms of accuracy and time consuming due to environmental conditions, traffic density, the battery power of the device, and distance of a vehicle from the device (Kisingo, Hamisi, Iddi, and Maiseli, 2021). In addition to that, these devices are monitored by a well trained person every time. Also number plate recordings were done manually which might be time consuming and might result into a burden to traffic as the vehicle had to be stopped most of the times.

To address these, deep learning techniques are being employed currently, which have shown promising results in various computer vision tasks. As per (Lin, Jeng, and Lioa, 2021) a lot of computer vision algorithms like Gaussian mixture model (GMM), single-shot detection (SSD), Convolutional Neural Network (CNN), Region-based Convolutional Neural Network (R-CNN) and You Only Look Once (YOLO) are being employed for that purpose. However, developing and training deep learning models can be a challenging task that requires significant expertise in both machine learning and computer vision. In this essay deep learning model and computer vision techniques will be used to estimate vehicle speed and number plate recognition.

To simplify this process, ChatGPT-based computer vision code will be generated, as many studies including (Akbar and Khan, 2023) have mentioned the potential of ChatGPT in software engineering researchs. ChatGPT uses language processing to process the input texts and generate the code as response.

1.1 Motivation

Smart road traffic management systems rely on accurate and efficient vehicle speed estimation and number plate recognition. However, classical methods for these tasks are often limited. To overcome these limitations, modern thecniques based on deep learning, specifically CNN, have occurred as promising methods. This thesis proposes a deep learning-based framework for vehicle speed estimation and number plate recognition using ChatGPT-based computer vision code generation. The proposed framework utilizes the power of deep learning and natural language processing to generate optimized computer vision code for these tasks. This code can significantly improve accuracy and efficiency, and reduce the need for manual development.

The proposed framework is based on recent advances in deep learning and computer vision, including CNN. The framework also leverages recent advances in computer vision and object detection algorithms to detect and recognize vehicles and number plates in real-time as well as enhancing the smart or intelligent road traffic management systems.

1.2 Problem Statement

Despite the improvements in the vehicle speed estimation and number plate recognition systems, challenges such as extreme weather conditions, noisy systems and differences in light intensities persist (Pirgazi, Pourhashem Kallehbasti, and Ghanbari Sorkhi, 2022). The other challenges include the numerous shapes and designs of the number plates, the non existence of a single standard number plate format, angle variations in image capturing, and vehicles occlusion (Sultan, Khan, Shah, Shahzad, Khan, and Mahmood, 2023).

The need of automatic and instant implementations motivated the integration of vehicle speed estimation and number plate recognition deep learning models with edge computing systems. However, the integration of deep learning and computer vision models for vehicle speed estimation and number plate recognition are also a hard task, due to the computation requirements of those methods and the limitations of edge devices (Surianarayanan, Lawrence, Chelliah, Prakash, and Hewage, 2023).

This project is intended to cope with most of those challenges using the most optimized deep learning and computer vision system, with the assistance of ChatGPT. Optimisation techniques in deep learning models are employed and the most efficient and accurate object detection and recognition methods are used to improve the performance of the system.

1.3 Objectives of the Thesis

The objectives of this thesis are:

1. to study methods of vehicle speed estimation and number plate recognition.
2. to develop a deep learning model for vehicle speed estimation and number plate recognition using ChatGPT-based computer vision code generation.
3. to compare the performance of the developed model with classical methods and past computer vision versions.
4. to evaluate the accuracy and efficiency of the developed model.
5. to demonstrate the feasibility of using ChatGPT code generation for developing and optimizing deep learning models for computer vision tasks.
6. to integrate model implementation with edge computing.

1.4 Organization of the Thesis

This thesis is organized as follows. Chapter 1 provides background information, the motivation, and the problem statement of the project. In chapter 2 we review the historical accounts of traffic management specifically on vehicle speed estimation and number plate recognition. The literature review of the related work in areas of deep learning, computer vision, vehicle speed estimation and number plate recognition is presented in chapter 3. Chapter 4 follows by describing the methods and algorithms used including data collection, algorithms of models to be used and the use of edge device. Chapter 5 presents the implementation of the methods. This chapter focuses on the experimentation of the methodology and the results obtained it also explain the practical demonstration of this thesis. Chapter 6 discusses the implications of the findings and the potential for future research in the same field. It also provides a summary of the main contributions of the thesis and concludes the work.

2. Historical Accounts of Vehicle Speed Estimation and Number Plate Recognition

Vehicle speed estimation and number plate recognition are two important technologies used in modern road traffic management systems, like speed limit enforcement, detecting stolen vehicles, etc. These technologies have a long and interesting history of development. Usually, talking about speed estimation, we consider the distance travelled by an object per unit time. It is a hard task to estimate the speed of an object with only our eyes, but easy to recognise images like number plates in normal conditions.

For that purpose, with the development of technology, in traffic management systems, people adapted different methods for estimating vehicle speed and recognising number plates. Through out history, there are a lot of improvements made on these methods to increase effectiveness of the system. In this section we review some of these methods and their evolution up-to-date.

Speed measurement technology has evolved from basic stopwatches to more sophisticated radio-based and laser-based systems. The first attempts to measure the speed of vehicles using stopwatches and two reference points date back to the 19th century. Later, radio-based device radar was adapted for use in measuring the speed of cars in the early 20th century. The first radar speed detection device (radar speed gun) was developed in the 1940s and was quickly widely used by police forces around the world (Wikipedia contributors, 2023a).

To increase the accuracy in vehicle speed measurements, in 1989, laser-based speed estimation devices, also known as lidar speed guns, were developed and started to be used by police forces (Wikipedia contributors, 2023b) in speed estimation. These devices use laser beam to measure the distance to a vehicle and calculate the speed based on the change in distance over time. They were found to be more accurate than radar speed guns because of the longer wave lengths and lower frequencies of laser beams.

The optical flow method, which studies the motion of an object in a visual scene was developed in the 1980s (Prazdny, 1981; Rieger and Lawton, 1983; Adiv, 1985), and in the 1990s it underwent further improvements (Beauchemin and Barron, 1995). In the 2000s, this method became famous in the field of robotics and autonomous vehicles in their way (i.e, road) navigation systems and obstacle detecting (Song, Seneviratne, Althoefer, and Song, 2007; Song, Song, Seneviratne, and Althoefer, 2008). Since then, optical flow is being employed in vehicle speed estimation for traffic management purposes. Optical flow is currently, integrated with the modern object detection algorithms to increase its relevancy. The most used optical flow version is Lucas-Kanade algorithm for instance (Qimin, Xu, Mingming, Bin, and Xianghui, 2014) and it is mostly integrated to CNN based object detection algorithms (Ren, He, Girshick, and Sun, 2015).

In traffic flow monitoring, it is an important task to recognise the vehicles which are violating the rules and orders, specifically the vehicles which over speed. For that purpose, the idea of unique identifier for vehicles, motivated the invention of the number plates. The use of number

plates to identify vehicles dates back to the 1890s century when car registration systems were first introduced (Wikipedia contributors, 2023c). Since then and before the revolution of technologies like computer vision, in traffic management systems, the number plates were recognised and recorded manually by traffic police officers in charge.

In 1976, an automated computer vision based system for reading number plates were developed at the police scientific development branch in UK (Qadri and Asif, 2009). In these systems the characters of number plates are detected and recognised, after detecting and extracting the number plate(s) from a whole image of a vehicle. The development of these systems was powered by the development of computer vision and deep learning, where the more accurate algorithms are being employed into these systems.

Today, automated number plate recognition systems are widely used in road traffic management systems to monitor traffic flow, detect violations such as speeding and red-light running, and enforce road user charges such as tolls and congestion charges. They are also used by law enforcement agencies to track down stolen vehicles and investigate crimes.

3. State of the Art

3.1 Deep Learning

Definition and Applications of Deep Learning

Since 2006, deep learning has emerged as a new machine learning research area (Bengio, 2007), so before reviewing deep learning we need first to understand machine learning and artificial neural network. According to (Chassagnon, Vakalopolou, Paragios, and Revel, 2020) the term "Machine Learning" was first mentioned by (Samuel, 1959). They defined machine learning as the field of study that gives computers the ability to learn without being explicitly programmed. The model is programmed but the behaviour and the learning of the program is changed by the data used to train the program/model.

In (Bishop, 1994), they described artificial neural network as a machine learning model which is designed to learn information based on biological neural network or brain information processing mechanism. Deep learning which is deep neural network, is an artificial neural network configuration where neurons are organized in multiple successive layers (Lecun, Bengio, and Hinton, 2015). (Deng, 2014) defined deep learning as a type of machine learning, with many layers of information-processing stages in hierarchical architectures. Deep Learning methods are kind of learning methods where non-linear transformations are organized into multiple layers to process information with the purpose of learning complex functions (Lecun, Bengio, and Hinton, 2015).

Although deep learning has various applications in many fields, in this project, much focus is on deep learning applications for image processing (Computer Vision). The most used deep learning model for Computer vision are CNN and R-CNN. These two methods have shown good results for object detection, CNN for single object detection and R-CNN for multiple object detection in a single image (Zhou, Gong, Fu, and Du, 2017). In Biological data processing, deep learning has shown great importance. (Mamoshina, Vieira, Putin, and Zhavoronkov, 2016) summarised different deep learning approaches applied on different kinds of biological data.

Many studies including (Deng, 2014), examined the applications of deep learning in signal processing for many purposes. Deep learning tehcniques have also being applied in the field of ecology. In particular, (Christin, Hervet, and Lecomte, 2019) examined some of them like behaviour studies, population monitoring, ecological modelling and ecosystem management and conservation. There are wide range of applications in this field, which include food science, climate science, cybersecurity, material science, network management, etc. Specifically in the sense of this essay, a lot of works have been done in deep learning for traffic management purposes. (Jia, Wu, and Du, 2016) studied traffic speed prediction using deep learning method and there are many related studies including (Lv, Duan, Kang, Li, and Wang, 2015) where they explored Traffic Flow Prediction With Big Data.

Deep Learning Structure for Computer Vision

In computer vision, CNN is the basic structure used in designing most of the algorithm. The introduction of CNN was motivated by the visual cortex of animals, for machines to learn the

images the way animal's brain and optical system process them. Despite the popularity of CNN, the basic deep learning framework (Deep feedforward neural networks) can also be used for simple computer vision tasks.

Deep feedforward neural networks (classical fully-connected multi-layer neural networks), the most basic deep neural network structure are built in a way that the information are processed in a parallel and in a forward direction (Khadivi, Tandon, and Ramakrishnan, 2016). Figure 3.1 shows an example of a Deep feedforward neural network structure. The discribed network consist of $v+1$ layers with multiple neurons. X is the vector representation of input data, X_n is the representation of data in the n^{th} hidden layer, $X_v = \hat{Y}$ is the representation of data in the output layer of the neural network which is the estimation of the desired output Y.

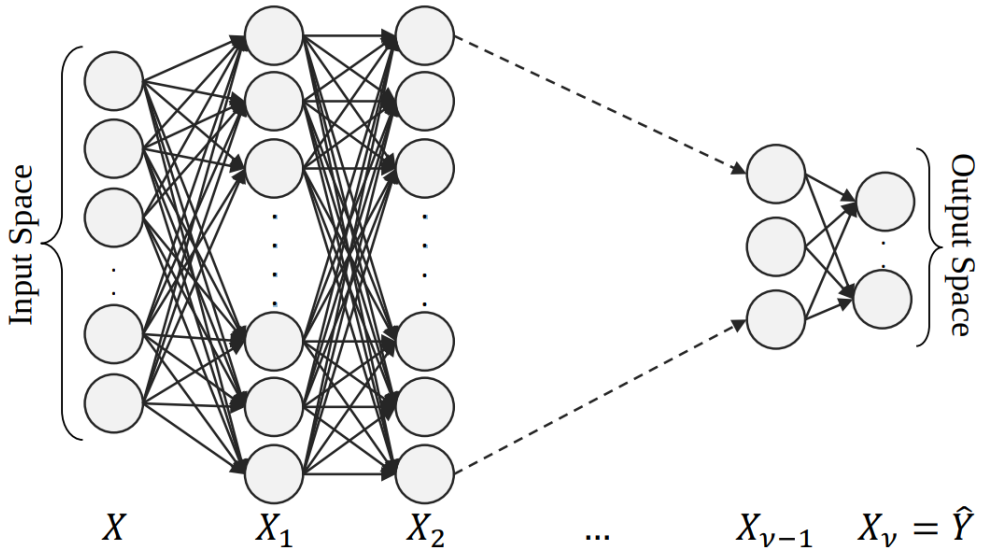


Figure 3.1: Deep Feedforward Neural Networks structure (Khadivi, Tandon, and Ramakrishnan, 2016)

The j^{th} neuron of the n^{th} layer process data using the formula in the equation (3.1.1) to get $x_{n,j}$, where m_{n-1} is the number of neurons in $(n-1)^{th}$ layer and $w^n_{i,j}$ is a weight between the output of the i^{th} neuron in $(n-1)^{th}$ layer and the input of the j^{th} neuron in n^{th} layer. b^n_j is the bias of the j^{th} neuron of n^{th} layer and $f^n(.)$ is the output function (activation function) of the neurons in that layer. The weights w^n_i and biases b^n_j are found by solving the optimization problem on the used loss function.

$$x_{n,j} = f^n\left(\sum_{i=1}^{m_{n-1}} w^n_{i,j} x_{n-1,i} + b^n_j\right) \tag{3.1.1}$$

(Aloysius and Madathilkulangara, 2017) explained the structure of CNNs which are composed of three layers convolutional layer, pooling layer and fully connected layer.

In the convolutional layer, there is use of filters or kernels on the input array of data. In this layer the nodes or neurons are considered as feature maps, where in each feature map the same filter parameters are shared. An example of a description of the operations on the input data in the convolutional layer is presented in figure 3.2.

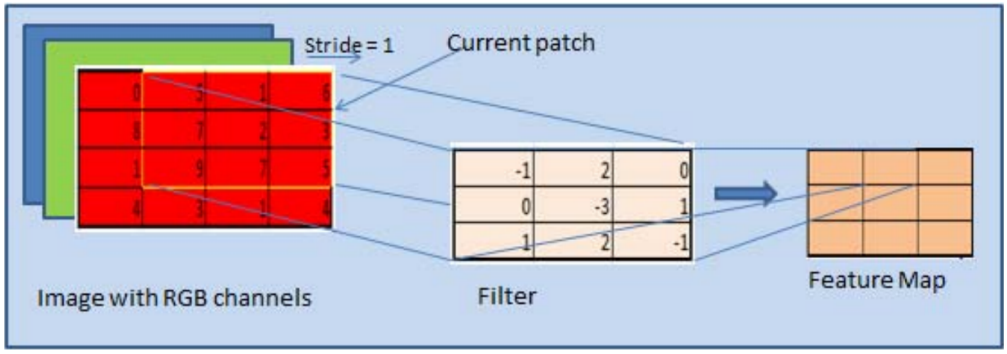


Figure 3.2: Convolutional Layer, demonstration (Aloysius and Madathilkulangara, 2017)

An example used by (Aloysius and Madathilkulangara, 2017), is image data in RGB form (with three channels), where the filter must apply on all channels, and the feature map should also have three channels. Each entry in the feature map is the sum of the hadamard product of filter and a portion of input array of the same size plus bias on each step relative to the stride value.

In the pooling layer, information is summarized by reducing the dimensionality of the convolutional layer output array. (Aloysius and Madathilkulangara, 2017) mentioned the common pooling operations, which are max pooling, average pooling, stochastic pooling, spectral pooling, spatial pyramid pooling and multiscale orderless pooling. Figure 3.3 demonstrates the pooling layer operations with max-pooling method. The desired output array's size is what dictates the slicing of the input array and in each slice the maximum is extracted. At the end of this process, an activation function is applied at each feature map to connect all the feature maps to the fully connected layer (i.e, the resulting array will be the input to the fully connected layer). The fully connected layer is of the same structure as the classical network (deep feedforward neural network) and the operations are the same, in this layer is where a loss function is applied.

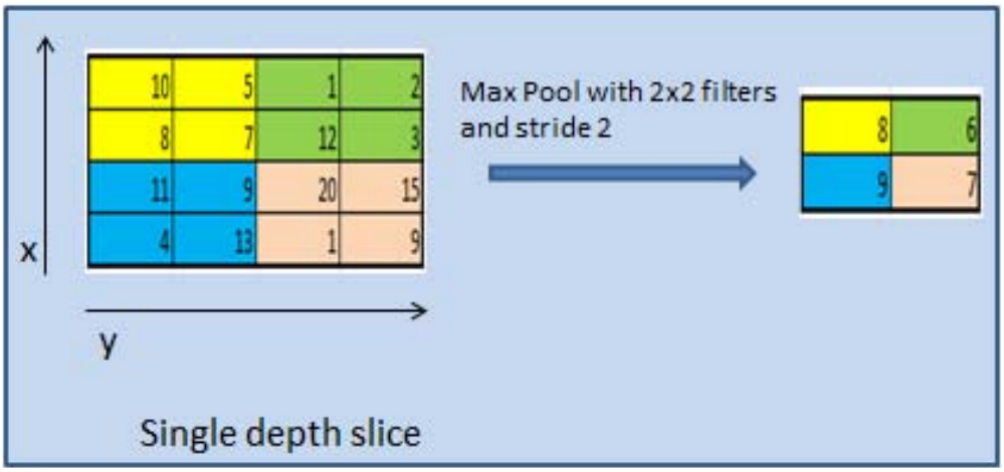


Figure 3.3: Pooling Layer, demonstration (Aloysius and Madathilkulangara, 2017)

The full structure of the whole deep CNN is demonstrated in figure 3.4. An example of image array with shape 32×32; the deep CNN with five 20×20 shaped feature maps on the first

convolutional and pooling layers and ten 8×8 shaped feature maps on the second convolutional and pooling layers, the fully conected layer with three layer 30 neurons each and the twenty neurons output layer for classification purpose.

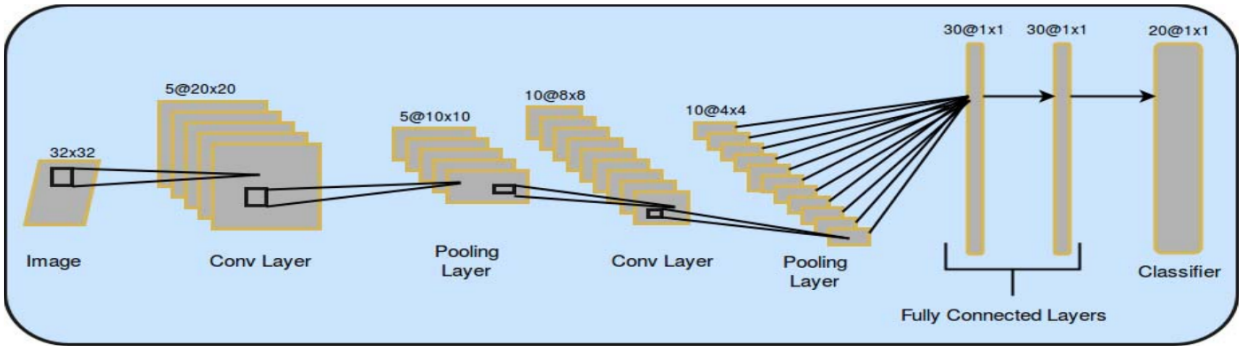


Figure 3.4: Deep Convolutional Neural Network, demonstration (Aloysius and Madathilkulangara, 2017)

3.2 Computer Vision Frameworks

Computer vision has a wide range of applications, in object detection and recognition, image/video analysis and captioning, image deblurring and generation. All of these are applied in many research and industrial areas including security systems, medical hospitals, self driving cars, etc. In traffic management, specifically vehicle speed estimation and number plate recognition, there are a lot of computer vision algorithms to cope with. These computer vision algorithms are employed because of their effectiveness, efficiency and accuracy in vehicle speed estimation and number plate recognition over classical methods.

Estimation of the speed of objects using computer vision requires videos and computer algorithms to process the video the way human eyes and brain process them and estimate the speed which is not trivial for humans. The video acquired is divided into consecutive frames of images which the computer vision algorithm will process. In this section, we explore both tradition computer vision algorithms and deep learning based algorithms specifically for vehicle speed estimation and number plate recognition.

Vehicle Speed Estimation

To estimate the speed of an object from video data, there must be the tracking of that object in each image frame of the video. There are a lot of computer vision algorithms for objects detection and they can be employed in vehicle speed estimation of the object knowing the pixel change and time difference between the consecutive frames. Some proposed methods include statistical models like GMM and hidden markov models. There are many other algorithms including optical flow, snake-based techniques, and edge-based techniques (Schoepflin, 2003).

In current vehicle speed estimation research, there is a preference for optical flow algorithms over other methods. In optical flow, the apparent motion of an objects in a visual scene is detected.

It is used to determine the object motion by calculating the spatiotemporal changes and the correlation of the pixel between two consecutive frames in the video (Qimin, Xu, Mingming, Bin, and Xianghui, 2014). Equations (3.2.1) was used by (Qimin, Xu, Mingming, Bin, and Xianghui, 2014) to estimate the speed of each matching vehicle corner detected in the optical flow framework applied on two consecutive frames k^{th} and $(k-1)^{th}$.

$$v_j(k) = \frac{P(x_j(k), y_j(k)) - P(x_j(k-1), y_j(k-1))}{\Delta t} \tag{3.2.1}$$

In this equation, $P(x_j(k), y_j(k))$ is the position of the vehicle corner j estimated using the pixel displacement and the camera calibration parameters in the k^{th} and $(k-1)^{th}$ frames, Δt is the time difference between the two frames, and $v_j(k)$ is the speed estimation of matching vehicle corner j between the two frames. Additionally, he used

$$v = \frac{\sum_{j=1}^{n_{max}} v_j(k)}{n_{max}} \tag{3.2.2}$$

to calculate the average vehicle speed, where v is the estimated average speed of the vehicle and n_{max} is the total number of detected matching vehicle corners between k^{th} and $(k-1)^{th}$ frames.

To improve the performance of optical flow algorithm and other traditional computer vision algorithms, they integrate them to deep learning algorithms. For instance, different CNN-based algorithms were integrated to the optical flow to estimate vehicle speed. These algorithms include, single-shot detection (SSD) (Liu, Anguelov, Erhan, Szegedy, Reed, Fu, and Berg, 2016), R-CNN (Ren, He, Girshick, and Sun, 2015) and YOLO (Redmon, Divvala, Girshick, and Farhadi, 2016). Fast R-CNN, faster R-CNN, YOLOv1 (Redmon, Divvala, Girshick, and Farhadi, 2016), YOLOv2 (Redmon and Farhadi, 2017), YOLOv3 (Zhang and Zhu, 2019) and YOLOv4 (Bochkovskiy, Wang, and Liao, 2020) were developed on the basis of improving R-CNN and YOLO.

Number Plate Recognition

Similarly to the vehicle speed estimation, number plate recognition can be done automatically using computer vision ans deep learning algorithms. In this project, number plates of the target vehicle are extracted from video frames and then recognised. For this reason, the ability for a computer vision algorithm to successfully extract the number plate from an image is an important aspect in determining the performance of the algorithm.

Most of number plate extraction algorithms use the shape, color and characters of number plates. Those with shape detection framework include, Vertical Edge Detection Algorithm (VEDA) (Dev, 2015), Connected Component Analysis (CCA) (Slimani, Zaarane, Hamdoun, and Atouf, 2019) which detect the rectangle shapes on the target object (i.e, vehicle) by identifying connected pixels in the image, generalized symmetry transform (GST) (Kim and Chien, 2001) These use the symmetry of the number plate corners to extract it.

On the other hand, those algorithms which are build on the basis of detecting color of the number plate to extract it, are also used. One of these was developed by (Shi, Zhao, and Shen, 2005), where they applied color detection based number plate extraction algorithm in china using Hue,

Lightness, Saturation (HLS) color space. Another one was developed by (Chang, Chen, Chung, and Chen, 2004) utilizing the color edge detection algorithm.

The number plate extraction using character features of number plates are also employed. (Zhang, Jia, He, and Wu, 2006) constructed a number plate extraction algorithm based on detecting text/characters using cascade classifier (Viola and Jones, 2004). (Matas and Zimmermann, 2005) developed an algorithm to detect text-like or number-like characters. After detecting texts or numbers, they label the longest linear spatial configuration of these region as number plate and extract it. To improve the accuracy and efficiency of those algorithms, (Laroca, Severo, Zanlorensi, Oliveira, Gonçalves, Schwartz, and Menotti, 2018) designed a YOLO-based plate detection algorithm. Also, (Montazzolli and Jung, 2017) CNN-based framework provided better results in detecting in their study of detecting Brazilian number plates.

The process of number plate extraction is followed by number plate (characters) recognition. But before recognition, the characters in the number plates need to be detected and segmented. Most of number plate segmentation methods use texts/characters detection algorithms to do segmentation of the characters. (Haider and Khurshid, 2017) used CCA method to detect and segment the characters on the number plates. CNN/YOLO-based models are also used in number plate character segmentation.

The last crucial step in number plate recognition, is the text/character recognition known as Optical Character Recognition (OCR). The segmented number plate characters are recognised alone one by one. (Slimani, Zaarane, Hamdoun, and Atouf, 2019) used the cross correlation method to recognise segmented characters by identifying the character pattern that segmented characters follow. This method was named template matching and it is said to be the easiest one (Mufti and Shah, 2021). However, it is not applicable in solving complex problems in real time. The support vector machine (SVM) based are also used in charactor recognition. (Al-Shemarry and Li, 2020) used SVM to classify the characters. Deep learning methods like CNN and feed forward networks, are also used in character recognition (Liu, Huang, Cao, and Huang, 2018b).

3.3 ChatGPT Code Generation

ChatGPT is a conversational designed model, which operates on basis of natural language processing model and other machine learning models. It is a product of OpenAi, one of the leading artificial intelligence research organisation. The fundamental structure of ChatGPT's design is a transformer, a type of neural network architecture that empowers the model to examine sequences of information (Hassani and Silva, 2023). This model has been trained on a massive amount of text data sourced primarily from the internet, including web pages, articles, books, and social media platforms, in multiple languages, which enables it to process various texts and generate the outputs.

In this project, the deep learning-computer vision code is generated from ChatGPT, by querying it with some text asking for codes needed. In that context, we review some of the studies made on ability and the performance of ChatGPT in terms of generating computer programming codes.

(Biswas, 2023) examined the role of ChatGPT in computer programming. He tested the role of ChatGPT in code generation and correction, predicting and suggesting code snippets. He also tested automatic syntax errors and common mistakes fixing abilities as well as the ability to provide suggestions for code optimization and restructuring, to generate missing code based on the context of the project and answering technical questions about programming. A study on debugging capabilities of ChatGPT (Shafiq Surameery and Shakor, 2023), concluded that it can play a role in solving programming bugs by providing debugging assistance, bug prediction, and bug explanation. The overall advantage of ChatGPT is a promising artificial intelligence technology, and it can be time saving, efficient, accurate and easy to use, etc.

3.4 The use of Edge Device

The purpose of machine learning is to solve real life problems. Some of those problems need real time handling and they exist with big volume of data incoming on high speed. With the development of Internet of Things (IoT), machine learning models are being incorporated into IoT devices to deal with those issues, and this is known as edge computing. IoT devices which process data at the point and moment of acquisition are called edge devices.

(Murshed, Murphy, Hou, Khan, Ananthanarayanan, and Hussain, 2021), defined edge device as an end device which process data or the end server which store data. The use of edge devices reduced the need of storing or transferring too much data in the main servers. For instance (Côté-Allard, Fall, Campeau-Lecours, Gosselin, Laviolette, and Gosselin, 2017) built a video analytics system for autonomous drones where an edge devices were used to save bandwidth. The need for transferring video data to the cloud was reduces up to 10 times.

There are a different examples of edge devices that are being employed. (Johnson, 2019) mentioned some examples of leading tech companies which are investing in the edge computing area specifically for machine learning, deep learning, artificial intelligence. He reported about the efforts made by Intel, Microsoft, NVIDIA, Google, and IBM on improving the processing power of edge devices by shifting from 8-bit processors to 16-bit processors. An architecture called Agile Condor has been created to perform real time computer vision tasks using machine learning algorithms (Isereau, Capraro, Cote, Barnell, and Raymond, 2017). The architecture can process 7.5 teraflops, which allows it to run the latest deep learning-based object detection architectures like Faster RCNN and YOLO in real time.

Despite those improvements, there are still inconveniences in using edge devices. The high computation power required by deep learning models is the first challenge. In addition, edge devices have limited processing power, memory, communication resources and network bandwidth. Edge devices are also low-power devices and operate on batteries (Surianarayanan, Lawrence, Chelliah, Prakash, and Hewage, 2023). To cope with these issues, researchers are involved in building optimization methods for artificial intelligence/deep learning-based models and edge devices systems.

There are hardware optimisation methods proposed, for instance (Buber and Banu, 2018) demonstrated the advantage of implementing deep learning models on graphical processing units (GPU)

computers. (Lee, Tsung, and Wu, 2018) reviewed different trends being made in improving the use of edge devices in artificial intelligence, where they mentioned the ongoing developments and improvements of edge devices processing power using AI processing Units (APU). A technique of network quantization was utilised by (Novac, Boukli Hacene, Pegatoquet, Miramond, and Gripon, 2021), where the number of bits used to encode each weight of the model are reduced, to ensure that the total memory footprint is reduced by the same factor. Operations are also performed using integer instead of floating-point data type as integer operations and these require less computational power.

There are also optimisation methods (Surianarayanan, Lawrence, Chelliah, Prakash, and Hewage, 2023), the pruning method was proposed for the model complexity reduction, stochastic gradient descent and use of hyperparameter tuning methods were also suggested. Other methods are matrix decomposition, where the dimensions of original matrices to be used are reduced to lower dimensions and gradient scaling where the 32 bit gradient values are scaled to short integers.

4. Methods and Algorithms

One of the most important things in any task is defining well the methods to be used for better performance. In this chapter we demonstrate all the methods and algorithms which are used in the experimentation part of this thesis. We start with the process of acquiring the data, algorithms description follows and the ways of implementation follows. The description of the materials used, is also provided.

4.1 Data Acquisition and Preparation

The successful application of deep learning-based computer vision models for vehicle speed estimation and number plate recognition depend largely on the availability and quality of relevant datasets. In this section, we describe the datasets used in this study for fine-tuning the object detection model, as well as the dataset that will be generated/used during the implementation phase using a Megapixel 4k USB camera module.

Firstly, the 2020 repository of CCPD (Chinese City Parking Dataset) (Xu, Yang, Meng, Lu, and Huang, 2018) was used to fine-tune the object detection model for number plate recognition. The CCPD2020 dataset comprises vehicle images in different scenarios annotated with number plate bounding boxes, positions, and characters. It is divided into three sets: train, validation, and test sets.The dataset was selected because of its easy availability, large size as well as the annotations provided, making it suitable for training deep learning-based object detection models. The other repository used was Licence plate digits classification dataset provided from European Union (EU) licence plates characters. This is a an open source huge dataset which is composed with images of characters in gray scale. The dataset includes different scenarios like characters captured at different angles at different intensities. The dataset includes normal images blurred images and bolded images which make it fit in deep learning model training basically for the task of text recognition.

Secondly, to generate the dataset for vehicle speed estimation and number plate recognition during the implementation phase, a Megapixel 4k USB camera module (see figure 4.1) is used to capture street video streams. The camera module is equipped with a Sony IMX415 4K pixel color CMOS image sensor, with a maximum resolution of 3840 (H) x 2160 (V) and CCD full frame image acquisition technology. The camera supports 4K video at 30fps, which makes the video more stable and smooth, without any noise or broken images. It has an 85° non-distortion lens and auto-focus capabilities, including auto-exposure control and facial-enhancement technology, which optimizes the image automatically to make the picture or video streaming look more professional. The camera module is connected to the Raspberry Pi 4 to process the video streams and feed them into the models for speed estimation and number plate recognition. The generated dataset is used to evaluate the performance of the trained models under real-world conditions.

Figure 4.1: Image: Megapixel 4k USB camera

4.2 Object Detection Algorithm

Object detection is a fundamental problem in computer vision, with variety of applications. Over the past decade, significant progress has been made in this field, with the development of deep learning-based object detection methods that achieve good performance on standard benchmarks. Wide range of object detection algorithms are being employed in many areas for different applications. The most used deep learning algorithms include CNN based like YOLO variants, EfficientDet variants, ResNet and Transformers like Swin Transformer.

In this project, for the purpose of Object detection, we use one of the latest versions of YOLO algorithm, YOLOv5, which was released in June 2020 by Ultralytics. This algorithm is a real time object detection algorithm that uses a single neural network and anchor boxes to predict bounding boxes and class probabilities for multiple objects in an image. It has different scaled versions, designed for enhancement of lightness, efficiency, and accuracy relative to a specific purpose. Those versions are YOLOv5s (small), YOLOv5m (medium), YOLOv5l (large), and YOLOv5x (extra-large). YOLOv5s is specifically made to be deployed to devices with limited memory and computing resources by reducing its size which maintains the efficiency on those devices (Pang, Li, Zhang, Meng, and Zhang, 2022).

YOLOv5 Architecture

The YOLOv5 algorithm relies on the structure of all YOLO variants, consisting of a backbone, neck and head. The input end of YOLOv5 uses mosaic data enhancement method and adaptive anchor box calculation functionality and the input image to the backbone must be of 640. The figure 4.2 shows the overall structure of YOLOv5.

To improve the lightness and efficiency of the YOLOv5, the algorithm is down-scaled to the

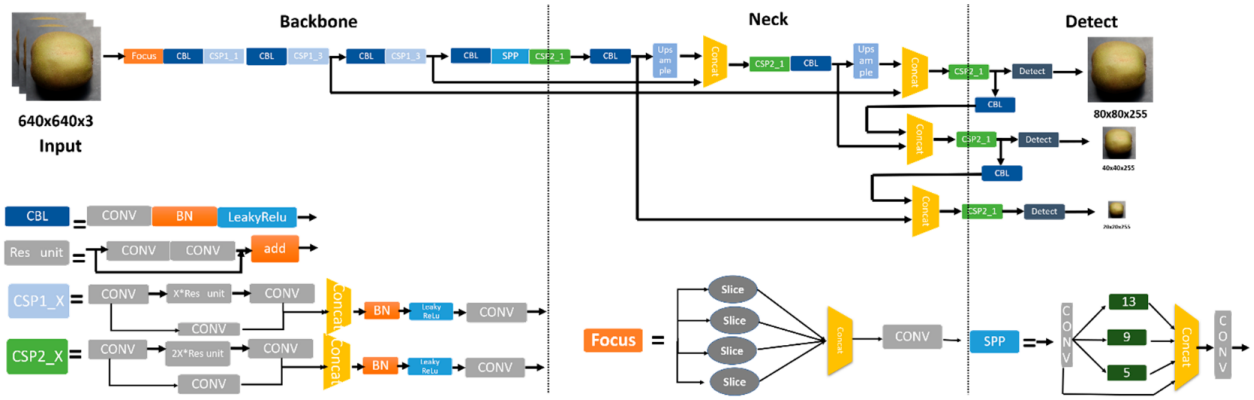


Figure 4.2: YOLOv5 Archtecture (Yao, Qi, Zhang, Shao, Yang, and Li, 2021)

small version, YOLOv5s. The Focus framework is replaced by CBL framework. The activation function is changed to the SiLU function $SiLU(x) = x \times Sigmoid(x)$. The SPP is replaced by a structure which serialize the input through multiple MaxPool layers of 5×5; i.e, the 9×9 convolution is replaced by two 5×5 convolutions, and 13×13 convolution is replaced by three 5×5 convolutions.

Anchor Boxes

Anchor boxes are predefined bounding boxes of different sizes and shapes that are placed throughout an image to help an object detection algorithm predict the location and size of objects. They allow the algorithm to detect objects of varying sizes and shapes, and help to reduce false detections by encoding prior knowledge about the likely location and size of objects. They are calculated by running a clustering algorithm like k-means on the training data set during the pre-processing stage of the YOLOv5.

Anchor boxes are used in detecting objects in the head structure of YOLOv5 by using a loss function to penalise the errors in the predicted bounding box coordinates.

Upsampling

In YOLOv5, the upsampling operation is used to increase the spatial resolution of the feature maps output by the backbone architecture. The upsampling operation can be defined mathematically as follows:

Suppose we have an input feature map x with dimensions $H \times W \times C$, where H, W, and C represent the height, width, and number of channels, respectively. We want to upsample the feature map by a factor of s, resulting in an output feature map with dimensions $sH \times sW \times C$.

One way to perform the upsampling operation is to use bilinear interpolation, which involves computing the weighted average of the values in the input feature map at each pixel location in the output feature map. Specifically, let $x_{i,j} \in \mathbb{R}^C$ denotes the feature vector at pixel location (i, j) in the input feature map, and let $y_{k,l} \in \mathbb{R}^C$ denotes the feature vector at pixel location (k, l) in the output feature map, where $i \in \{1, 2, \ldots, H\}$, $j \in \{1, 2, \ldots, W\}$, $k \in \{1, 2, \ldots, sH\}$, $l \in \{1, 2, \ldots, sW\}$ and s is the upsampling factor. Then the upsampling operation can be defined as:

$$y_{k,l,c} = \sum_{i,j} w^h_{k,i} w^w_{l,j} x_{i,j,c}, \tag{4.2.1}$$

where $w^h_{k,i}$ and $w^w_{l,j}$ represent the vertical and horizontal interpolation weights, respectively, and are given by:

$$w^h_{k,i} = \max\left(0, 1 - \frac{|k - is|}{s}\right) \tag{4.2.2}$$

$$w^w_{l,j} = \max\left(0, 1 - \frac{|l - js|}{s}\right). \tag{4.2.3}$$

In this formulation, the weights $w^h_{k,i}$ and $w^w_{l,j}$ are computed based on the distance between the pixel locations in the input and output feature maps, where closer pixels receive higher weights. The weights are normalized such that they sum to 1, ensuring that the output feature map has the same overall intensity as the input feature map.

By increasing the spatial resolution of the feature maps, the upsampling operation helps to improve the accuracy of object detection by enabling the network to detect small objects.

Batch Normalization (BN)

In YOLOv5, batch normalization is used to normalize the feature maps and improve the stability and performance of the algorithm.

The batch normalization layer is usually inserted after a convolutional layer and before the activation function. By normalizing the activations, batch normalization reduces the internal covariate shift, which is the change in the distribution of activations across layers during training. This improves the stability and performance of deep neural networks by allowing them to learn more robust features.

Let $x \in \mathbb{R}^m$ be the input to the batch normalization layer, where m is the number of activations in the mini-batch. The batch normalization operation can be broken down into the following steps:

1. Compute the mini-batch mean:

$$\mu_{\mathcal{B}} = \frac{1}{m} \sum_{i=1}^{m} x_i$$

2. Compute the mini-batch variance:

$$\sigma^2_{\mathcal{B}} = \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2$$

3. Normalize the input by subtracting the mini-batch mean and dividing by the square root of the mini-batch variance plus a small constant ϵ to avoid division by zero:

$$\hat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$

4. Scale and shift the normalized input by learnable parameters γ and β respectively:

$$y_i = \gamma \hat{x}_i + \beta$$

The parameters γ and β are learned during training by backpropagation to optimize the overall loss function.

YOLOv5 Loss Function

The loss function used in YOLOv5 is a combination of several loss terms. The overall loss is demonstrated as follows:

Let n be the number of grid cells in the image, m be the number of objects in the image, B be the number of bounding boxes per grid cell (in YOLOv5, $B = 3$) and C be the number of classes, then the overall loss function is given by

$$\mathcal{L} = \lambda_{\text{coord}}\mathcal{L}_{\text{coord}} + \lambda_{\text{obj}}\mathcal{L}_{\text{obj}} + \lambda_{\text{noobj}}\mathcal{L}_{\text{noobj}} + \lambda_{\text{class}}\mathcal{L}_{\text{class}} \tag{4.2.4}$$

where λ_{coord}, λ_{obj}, λ_{noobj}, and λ_{class} are hyperparameters that control the relative importance of the different loss terms.

The first term, $\mathcal{L}_{\text{coord}}$ defined as

$$\begin{aligned}\mathcal{L}_{\text{coord}} = &\sum_{i=1}^{n}\sum_{j=1}^{B} \mathbb{1}_i^j \left[\left(x_i^j - \hat{x}_i^j\right)^2 + \left(y_i^j - \hat{y}_i^j\right)^2\right] \\ &+ \sum_{i=1}^{n}\sum_{j=1}^{B} \mathbb{1}_i^j \left[\left(\sqrt{w_i^j} - \sqrt{\hat{w}_i^j}\right)^2 + \left(\sqrt{h_i^j} - \sqrt{\hat{h}_i^j}\right)^2\right]\end{aligned} \tag{4.2.5}$$

measures the error in predicting the center coordinates and dimensions of the bounding boxes where $\mathbb{1}_i^j$ is an indicator variable that equals 1 if the jth bounding box in the ith grid cell is responsible for detecting an object, and 0 otherwise. The variables x_i^j, y_i^j, w_i^j, and h_i^j are the predicted center coordinates and dimensions of the bounding box, while $\hat{x}_i^j$, $\hat{y}_i^j$, $\hat{w}_i^j$, and $\hat{h}_i^j$ are the corresponding ground-truth values.

The second term, $\mathcal{L}_{\text{obj}}$

$$\mathcal{L}_{\text{obj}} = \sum_{i=1}^{n} \sum_{j=1}^{B} \mathbb{1}_i^j \left(C_i^j - \hat{C}_i^j \right)^2 \tag{4.2.6}$$

measures the error in predicting the presence of an object in each grid cell, where C_i^j is a binary variable that equals 1 if the jth bounding box in the ith grid cell contains an object, and 0 otherwise, and $\hat{C}_i^j$ is the corresponding predicted value.

The third term, $\mathcal{L}_{\text{noobj}}$

$$\mathcal{L}_{\text{noobj}} = \sum_{i=1}^{n} \sum_{j=1}^{B} \left(1 - \mathbb{1}_i^j\right) \lambda_{\text{noobj}} \left(C_i^j - \hat{C}_i^j \right)^2 \tag{4.2.7}$$

measures the error in predicting the absence of an object in each grid cell for the bounding boxes that do not contain objects, where λ_{noobj} is a hyperparameter that scales the importance of this term relative to the other terms.

The fourth term, $\mathcal{L}_{\text{class}}$

$$\mathcal{L}_{\text{class}} = \sum_{i=1}^{n} \sum_{j=1}^{B} \mathbb{1}_i^j \sum_{c=1}^{C} \left(p_i^j(c) - \hat{p}_i^j(c) \right)^2 \tag{4.2.8}$$

measures the error in predicting the class probabilities for each bounding box that contains an object, where $p_i^j(c)$ is the predicted probability of the object in the jth bounding box in the ith grid cell belonging to the cth class, and $\hat{p}_i^j(c)$ is the corresponding ground-truth value.

In YOLOv5, the values of these hyperparameters are set to $\lambda_{\text{coord}} = 2.0, \lambda_{\text{obj}} = 1.0, \lambda_{\text{noobj}} = 0.5, \lambda_{\text{class}} = 2.0$. These values were found through experimentation and are used to balance the contribution of each loss term to the overall loss function.

To optimize the parameters of the YOLOv5 model, the gradient of the loss function with respect to the model parameters is computed using backpropagation. The parameters are updated using an optimization algorithm such as stochastic gradient descent (SGD) variants like Adam (adaptive moment estimation).

Loss Optimization

YOLOv5s loss function is optimized by Adam algorithm to get optimal model parameters. Here is a step-by-step mathematical demonstration of the Adam algorithm. The Adam algorithm 1 was proposed by (Kingma and Ba, 2014) to improve the basic SGD algorithm.

Algorithm 1 Adam, g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With $\beta_{t,1}$ and $\beta_{t,2}$, β_1 and β_2 are denoted to the power t.

Input: α: Stepsize (learning rate)
$\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
$f(\theta)$: Stochastic objective function (loss) with parameters θ (weights)
θ_0: Initial parameter vector (weights)
Output: Resulting parameters θ_t

$m_0 \leftarrow 0$ // Initialize 1st moment vector
$v_0 \leftarrow 0$ // Initialize 2nd moment vector
$t \leftarrow 0$ // Initialize timestep
while θ_t *not converged* **do**
  $t \leftarrow t + 1$ // Update timestep
  $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ // Get gradients on stochastic data sample at timestep t
  $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ // Update biased first moment estimate
  $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ // Update biased second raw moment estimate
  $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ // Compute bias-corrected first moment estimate
  $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ // Compute bias-corrected second raw moment estimate
  $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ // Update parameters
return θ_t

4.3 Speed Estimation Algorithm

Optical flow is an important algorithm in computer vision, with many applications in motion analysis. The goal of optical flow is to estimate the motion of objects in a video sequence. In this project, we used one of common algorithms of optical flow, the Lucas-Kanade algorithm.

The Lucas-Kanade algorithm is a local method that assumes that the motion of objects in an image is smooth and can be approximated by a first-order Taylor series expansion (Sharmin and Brad, 2012). The algorithm estimates the optical flow by solving a set of linear equations that relate the spatial derivatives of the image to the time derivatives of the image.

Lucas-Kanade Algorithm

The Lucas-Kanade algorithm works by assuming that the motion of objects in an image is constant over a small neighborhood. Let $I(x, y, t) = I(x + dx, y + dy, t + dt)$ be the intensity of an image at position (x, y), time t and $(x + dx, y + dy)$, time $t + dt$ respectively. The algorithm estimates the optical flow vector $\mathbf{v} = (u, v)$ at a pixel (x, y) by solving the following set of linear partial differential equations:

$$\begin{bmatrix} \frac{\partial I}{\partial x}(x_1, y_1, t) & \frac{\partial I}{\partial y}(x_1, y_1, t) \\ \frac{\partial I}{\partial x}(x_2, y_2, t) & \frac{\partial I}{\partial y}(x_2, y_2, t) \\ \vdots & \vdots \\ \frac{\partial I}{\partial x}(x_n, y_n, t) & \frac{\partial I}{\partial y}(x_n, y_n, t) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \frac{\partial I}{\partial t}(x_1, y_1, t) \\ \frac{\partial I}{\partial t}(x_2, y_2, t) \\ \vdots \\ \frac{\partial I}{\partial t}(x_n, y_n, t) \end{bmatrix} \tag{4.3.1}$$

where (x_i, y_i) are the pixel coordinates in a neighborhood around (x, y), and n is the number of pixels in the neighborhood.

Actual Speed calculation

After calculating the optical flow speed vector ($\mathbf{v}$), we need to find the relationship between the camera calibration and optics parameters with the real world length measurements to adjust speed factor and estimate the actual speed of the object. We can also easily estimate the actual speed by adjusting speed factor basing on the optical flow speed results and the vehicle actual speed on sampled videos. The equation (4.3.2) is the used formula in calculating the actual speed where A_S is the actual speed, s_f is the speed factor and $\mathbf{v}$ is the optical flow speed.

$$A_S = s_f \times \mathbf{v} \tag{4.3.2}$$

4.4 Optical Character Recognition Algorithm

Optical Character Recognition (OCR) is a technique used to extract text from images and convert it into machine-readable formats. OCR has many applications in different domains, such as document digitization, text translation, and number plate recognition in traffic management systems. In this project, we used two algorithms, CCA for number plate segmentation and a deep learning model (ResNet-18) for recognizing each character of the number plate.

CCA Algorithm

Before applying the CCA we do some image transformations. The input image is transformed into gray scale before being transformed into binary image by

$$\text{Grayscale} = 0.3R + 0.6G + 0.1B \tag{4.4.1}$$

where R,G,B represent red, green and blue channel values of the image respectively. The gray scaled image is then transformed into binary image using a certain threshold pixel value. The values less than the threshold takes black pixel value (0) and those above the threshold takes the white pixel value (255). The CCA algorithm is applied on the binary image. This algorithm detects the pixel neighborhoods with the same pixel values, after that the bounding boxes of connected pixels are used to track each character's location in the image.

ResNet-18 Algorithm

After extracting each character from number plate image, we used the designed deep learning architecture (ResNet-18). It is an advanced deep CNN architecture that is used for classification tasks in the field of computer vision. It was proposed by (He, Zhang, Ren, and Sun, 2016), introducing the residual learning based deep learning to cope with the challenge of training very deep neural networks.

The core idea behind residual learning is to learn residual mappings, capturing the difference between the desired output and the current prediction. By explicitly modeling the residuals, ResNet-18 enables the training of significantly deeper networks by mitigating the vanishing gradient problem.

The fundamental building block of ResNet-18 is the residual block. Each residual block takes an input x and applies a series of convolutional operations to transform it. The output of the block is the sum of the input and the transformed output, where the transformation is denoted by the function $F(x)$. The mathematical Formulation of the model is stated below.

Let x be the input to the residual block, and $H(x)$ be the transformation applied to x. The output y is computed as

$$y = H(x) = F(x) + x. \tag{4.4.2}$$

In this formulation, $F(x)$ represents the residual connection, allowing the network to learn the deviation from the input x, while $H(x)$ captures the primary transformation performed by the residual block.

The ResNet-18 algorithm is structured in the following way:

1. Input: input image is resized to 224x224x3.
2. Initial Layer: Convolutional layer with a kernel size of 7×7 and stride 2, followed by batch normalization and ReLU activation (ReLU$(x) = \max(0, x)$).
3. Max Pooling: Max pooling layer with a kernel size of 3×3 and stride 2.
4. Residual Blocks: Four sets of residual blocks with different numbers of layers.
    - The input x is first passed through the convolutional layers, resulting in $F(x)$. This captures the transformation applied to the input by the layers. The purpose of these layers is to learn the residual mapping, which represents the "difference" or the additional information to be learned.
    - The input x is then added to the output of the convolutional layers, denoted as $F(x) + x$. This shortcut connection allows the gradient to flow directly through the network without passing through a series of non-linear activations, addressing the issue of vanishing gradients. If the dimensions of x and $F(x)$ do not match, an additional 1×1 convolutional layer is used to adjust the dimensions.
5. Global Average Pooling: After the last residual block, a global average pooling layer reduces the spatial dimensions to 1×1.
6. Fully Connected Layer: The output of global average pooling is fed into a fully connected layer with softmax activation for classification. The mathematical expression for the softmax function is

$$p_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{4.4.3}$$

where $z = (z_1, z_2, ..., z_K)$ represents the input vector to the fully connected layer and p_i represents the probability of class i. The predicted class is obtained by

$$\hat{y} = \arg\max_i(p_i). \tag{4.4.4}$$

The class $\hat{y}$ with the highest probability is considered the predicted class for the given input.

The loss function used in ResNet-18 is cross-entropy loss. Mathematically, the cross-entropy loss is defined as

$$\text{Loss} = -\sum y_{\text{true}} \log(y_{\text{pred}}) \tag{4.4.5}$$

where y_{true} represents the true class labels (one-hot encoded) and y_{pred} represents the predicted class probabilities from the softmax activation.

ResNet-18 Loss Optimization

In the training, the model parameters are optimized by the SGD algorithm. Below is a demonstration of the SGD algorithm.

Algorithm 2 Stochastic Gradient Descent (SGD) Algorithm.

Input: α_0: Initial step-length (learning rate)
$f(\theta)$: Loss function with parameters θ (weights)
θ_0: Initial weights
Output: Resulting weights θ_t
$t \leftarrow 0$ `// Initialize timestep`
while θ_t *not converged* **do**
  $t \leftarrow t + 1$ `// Update timestep`
  $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ `// Get gradients on stochastic data sample at timestep` t
  $\theta_t \leftarrow \theta_{t-1} - \alpha_0 \cdot g_t$ `// Update weights`
return θ_t

4.5 Integrating All Algorithms

Python programming language is used for the implementation of all algorithms and integration of the models. Python was chosen due to its simplicity, ease of use, and the availability of numerous libraries for machine learning and computer vision, and the stated algorithms are available in python. The development process is carried out on a computer with the required specifications, and the algorithms is tested and validated following the flowchart 5.1 specified in chapter 5.

To deploy the integrated models, an edge device, specifically a Raspberry Pi 4, will be used. The Raspberry Pi 4 is a low-cost, energy-efficient, and compact computer that can perform various tasks. By deploying the models on an edge device, it will be possible to reduce latency, decrease bandwidth usage, etc. The integration of the models will be done using Python and the required libraries will be installed on the Raspberry Pi 4.

4.6 Edge Computing

Edge computing has become an increasingly popular approach to data processing and computation, as it involves carrying out these tasks closer to the source of the data rather than in a centralized location. This approach has several advantages, including reduced latency, improved performance, and enhanced data security. One popular platform for edge computing is the Raspberry Pi 4 (see figure 4.3), a low-cost, single-board computer.

Figure 4.3: Image: Raspberry Pi 4 (Raspberry Pi Trading Ltd, 2019)

The Raspberry Pi 4 can be used to deploy models and carry out big tasks in real-time. Its compact size, low power consumption, and low cost make it an attractive option for developers who need a small-scale computing environment that can handle complex machine learning algorithms. The Raspberry Pi 4 has 8GB memory, which is sufficient for running most machine learning algorithms. The platform comes equipped with 2.4 GHz and 5.0 GHz IEEE 802.11B/g/n/ac Wireless LAN, Bluetooth 5.0, and double-true Gigabit Ethernet capabilities, which provides flexible connectivity options. It has 2 $\times$ USB 3.0 ports and 2 x USB 2.0 Ports for peripherals, along with 2 $\times$ micro HDMI ports that support up to 4Kp60 video resolution. A micro secure digital (SD) card slot is available for loading the operating system and data storage. In our implementation we used a 64GB SD card. The Raspberry pi 4 requires a 5.1V, 3a power via USB Type C. Its hardware specifications, combined with its low cost and small size, make it a versatile platform for edge computing applications.

5. Implementation

In this chapter, we discuss the implementation of all the methods and algorithms described above. We also discuss the results of the experiments made. The pipeline of the implementation process is demonstrated in the figure 5.1. It is a simple example of implementation process which can be adopted in real life applications specifically traffic management activities. The whole process is implemented with help of both hard and software materials. On software part, the built models, the use of python language scripts and libraries were used. The 4k USB camera and the raspberry pi 4 are used as the hardware facilities.

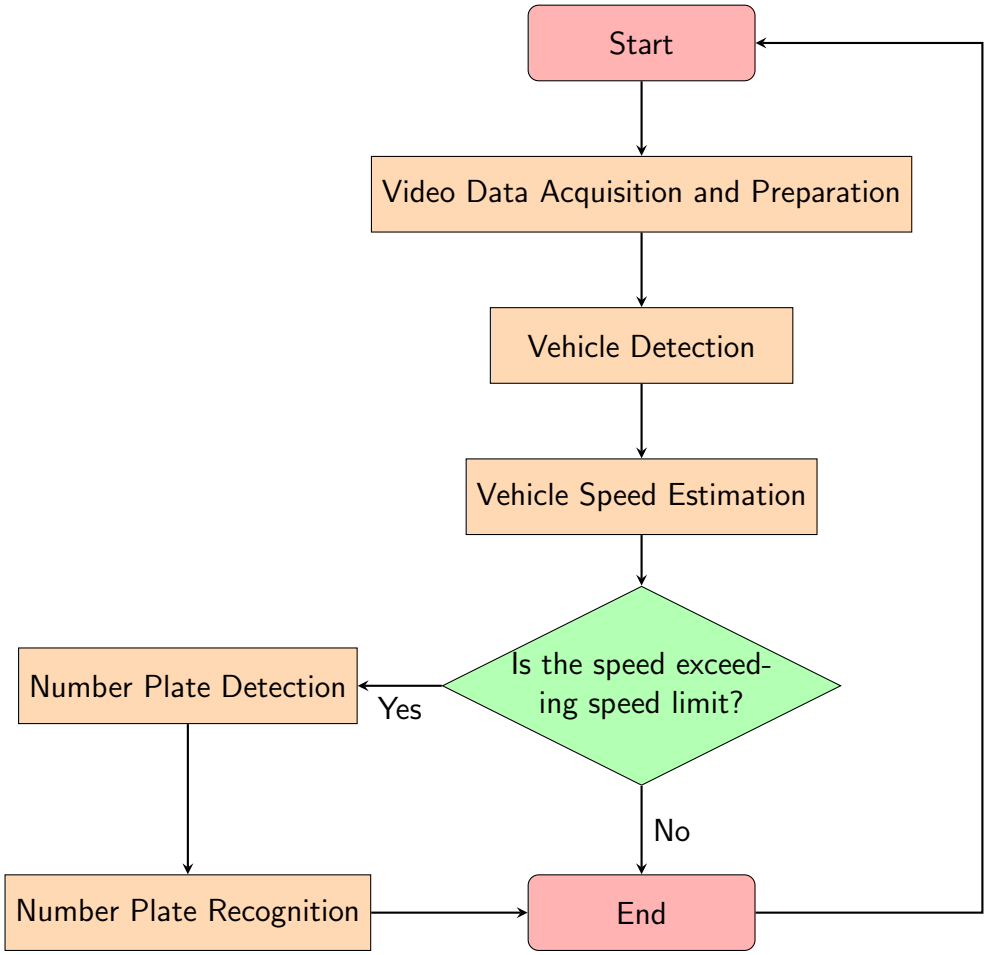


Figure 5.1: Implementation flow chart.

Below, all the stages of implementation process are covered. The first stage is the image data acquisition and preparation. The second is the vehicle detection model generation, the third is the number plate model development. Optical character recognition model generation is the fourth stage in the implementation. The next stage is camera calibration and vehicle speed estimation model development. The last stage is integrating all of the models in one system to follow the proposed implementation flow chart (see figure 5.1).

Data is the basic thing to start with in the process of building any kind of machine learning model.

The issue of getting good quality of data is still a challenge in many areas of machine learning. Specifically, in the area of deep learning where the data to be used are not exactly numerical, data collection and preparation become more difficult. In the case of computer vision models, one of the biggest challenges is data annotation yet it is the most important part for a model to be built. The other critical issue is the availability of variety of cases in the datasets.

In that regard, we used the data from the 2020 repository of the CCPD (Xu, Yang, Meng, Lu, and Huang, 2018) and Licence plate digits classification dataset. CCPD2020 is a dataset of 11775 vehicle images split into training, validation and test datasets. The data set is annotated with the bounding boxes of the number plates and the four coordinates of the number plate in the image as well as the brightness and blurriness of the image. This data set was used to train (fine-tune) the number plate detection model (YOLOv5s). This dataset was prepared before being used in the fine-tuning process. The only task required in this process was to provide the data annotation format that is acceptable by the YOLOv5s model. We used ChatGPT to generate a python code which changes the annotation format to the desired YOLOv5s annotation format. The Licence plate digits classification dataset from EU number plates characters was also used to train an OCR algorithm to recognize the number plate characters. This dataset is composed by 35500 number plate character images which are well labeled. The dataset was used to train the ResNet-18 model.

The vehicle detection model used was generated from a pre-trained object detection model, YOLOv5s which has shown abilities to detect many objects including vehicles from images in real time. The model was generated with help of ChatGPT. It is built under a python packages like pytorch which is known for dealing with big arrays and opencv which is known in handling visuals. This model is highly accurate for many object detection purposes. It is open source and it is implementable. Figure 5.2 is one example of YOLOv5s applied on image to detect vehicles.

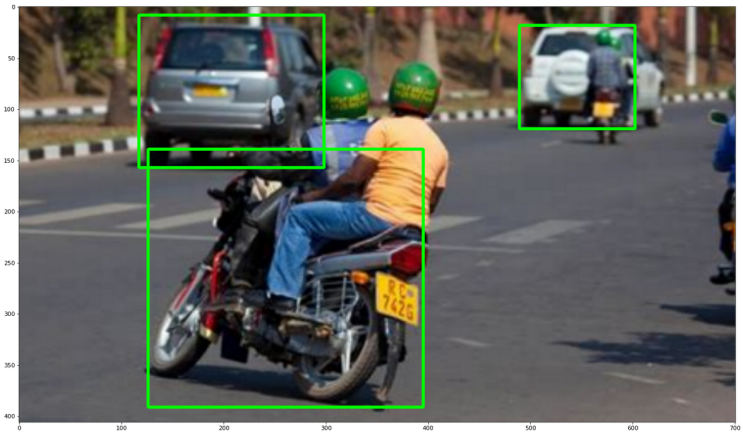


Figure 5.2: Vehicle detection with pre-trained YOLOv5s

The number plate detection model also relies on the object detection model, YOLOv5s. The original pre-trained YOLOv5s model is not able to detect the number plates. So, we used the structure of the pre-trained YOLOv5s model, fine-tuned it with the CCPD2020 dataset re-annotated with the number plates. The figure 5.3 shows an example of a detected number plate from a vehicle image.

Figure 5.3: Number Plate detection with fine-tuned YOLOv5s

One of the two most important aspects of this thesis is the number plate recognition. After detection of the number plate, the next thing is to detect and classify the characters on the number plate. To do that we used a character recognition system based on CCA (built under opencv) and the trained ResNet-18 model (built under pytorch). The CCA was used for number plate images segmentation and then the ResNet-18 trained on the number plate characters dataset was applied. In figure 5.4, a sample image where the number plate characters were recognized by CCA and ResNet-18 system is presented.

Figure 5.4: Number Plate recognition with CCA and ResNet-18

The other important aspect is the estimation of vehicle speeds. This is more crucial in the implementation of the thesis, and it can be adapted in the traffic management systems for vehicle speed enforcement. To build this model we integrated the YOLOv5s vehicle detection model with the optical flow algorithm known as Lucas Kanade (built under opencv). The implementation of this model was done with the help of the 4k USB camera by feeding the model with the

video streams via the raspberry pi 4. The Lucas Kanade algorithm is applied on two consecutive frames of a video in a continuous process. In order to accurately estimate the speed of vehicles through videos captured, the relationship between length measurements in the images/videos and the actual length measurements must be clear. For that reason we had to calibrate the camera before building and implementing the vehicle estimation model.

The rest was to integrate all the models in one system to easy the implementation. In the following sections, we discus more on the vehicle speed estimation and number plate recognition. We focus on the results of this project by evaluating the models built with respect to the desired results and assessing their effectiveness on being implemented in real life systems specifically traffic management system.

5.1 Vehicle Speed Estimation

The process of vehicle speed estimation is applied on the video recorded by a 4k USB camera which is connected to the raspberry pi 4 where the speed estimation model is deployed. The performance of the vehicle speed estimation depends on the YOLOv5s vehicle detection model and on the calibration of the camera. The algorithm of vehicle speed estimation has three parts. The first one is the YOLOv5s which detect the vehicles in each frame of the video. The second one is the Lucas Kanade optical flow algorithm which estimates the flow of pixel representing a vehicle in the frame and calculates optical flow speed. The third parts is the use of camera calibration parameters and other optical properties of the camera to estimate the actual speed of the vehicle.

5.2 Number Plate Recognition

The number plate recognition is divided into three parts which are vehicles detection, number plates detection and number plate characters recognition. This process lies on the YOLOv5s object detection models which are applied on one image from video frames recorded. The vehicle detection is done using the original pre-trained YOLOv5s, after that a vehicle is extracted from the whole image. The fine-tuned YOLOv5s number plate detection model is used to detect the number plate on the extracted vehicle image. Lastly, we apply the CCA and ResNet-18 system to detect and classify the number plate characters.

The fine-tuned YOLOv5s model was able to achieve good results where it detect the number plates with the accuracy of 98% on the CCPD2020 test dataset. The trained ResNet-18 model also achieved the accuracy of 99.7% on the Licence plate digits classification dataset.

5.3 The whole system and Edge Computing

The flow chart 5.1 dictated the implementation process and its setting on the raspberry pi 4. The videos generated through a 4K USB camera connected to raspberry pi 4 are fed into the integrated system of all built models. In this chapter we present the performance of the built models in the whole system after being applied on real data samples. We also discuss the challenges met while implementing the system in a real world scene.

Typically, the models demonstrate satisfactory performance on the training datasets. However, the effectiveness of the models in addressing real-world challenges fluctuates, based on the alignment between the knowledge acquired by the trained model and the characteristics of the actual data. The pre-trained YOLOv5s model was able to detect vehicles only from high quality images with no or less blurriness. The fine-tuned YOLOv5s performed well in detecting number plates, but it faced difficulties in handling images with low light conditions, specifically those taken at night with limited illumination. The OCR system also performs well, however, because of the training knowledge, the ResNet-18 model confuses **0** or **O** and **D** in some cases. It also has confusions on **I** and **T** in some cases. The challenges on the hardware (raspberry pi 4) arises with its capacity in handling large and heavy tasks. It is a big challenge to install all necessary packages on the raspberry pi 4 especially large packages like pytorch.

6. Conclusions and Future Work

This thesis presents a comprehensive study on the application of deep learning techniques for vehicle speed estimation and number plate recognition. The thesis explores different approaches in the field. The implementation of various algorithms and the integration of the entire system, along with edge computing, are also discussed.

The thesis has successfully addressed the problem of vehicle speed estimation and number plate recognition using deep learning techniques. The proposed algorithms, including the YOLOv5s, Lucas Kanade, CCA and ResNet-18, have demonstrated promising results in estimating the speed of vehicles and recognizing number plates. The pre-trained Yolov5s and Lucas Kanade algorithms are able to solve a problem of speed estimation where the vehicles are more than one in the camera sight. The fine-tuned YOLOv5s, CCA and ResNet-18 have shown potentials which can be adapted in intelligent road traffic management for automated applications like detection of stolen vehicles using cameras.

The integration of these algorithms into a cohesive system, along with the utilization of edge computing, has shown the potential for real-time application in intelligent road traffic management unlike the classical methods. The potentials of ChatGPT was also explored in this thesis. It was used for generating helpful accurate python codes with all needed information to easily understand and in short time. The findings of this thesis contribute to the advancement of computer vision-based approaches for vehicle monitoring and surveillance, however, it remains an open area of study for further improvements.

There are some future recommendations of this thesis to improve the accuracy and robustness in surveillance of vehicles in complex environments with edge computing. The first is the utilization of high performance edge computing devices to increase the efficiency in the implementation of real time deep learning models and continuous data flow management. Some of those high performance devices are Field-Programmable Gate Array (FPGA), NVIDIA-Jetson Nano. Having a specific data collection system relative to the problem to be solved and the area of implementation, specifically working with Rwanda national police is the second recommendation. The third is the use of high quality and easily calibrated cameras in the vehicle speed estimation systems. The integration of additional functionalities, such as vehicle make and model recognition, to provide a more comprehensive vehicle monitoring system is also recommended.

By addressing these areas, future research can contribute to the continued development and refinement of the proposed system, enhancing its practical implementation in real-world applications.

References

Gilad Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7(4):384–401, 1985. doi: 10.1109/TPAMI.1985.4767678.

Muhammad Azeem Akbar and Arif Ali Khan. Ethical aspects of chatgpt in software engineering research. 2023.

Meysam Sarfjoo Al-Shemarry and Yan Li. Developing learning-based preprocessing methods for detecting complicated vehicle licence plates. *IEEE Access*, 8:170951–170966, 2020.

Neena Aloysius and Geetha Madathilkulangara. A review on deep convolutional neural networks. pages 0588–0592, 04 2017. doi: 10.1109/ICCSP.2017.8286426.

S. S. Beauchemin and J. L. Barron. The computation of optical flow. *ACM Comput. Surv.*, 27 (3):433–466, sep 1995. ISSN 0360-0300. doi: 10.1145/212094.212141. URL https://doi.org/10.1145/212094.212141.

Yoshua Bengio. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2:1–127, 2007.

Christopher M. Bishop. Neural networks and their applications. *Review of Scientific Instruments*, 65:1803–1832, 1994.

S Biswas. Role of chatgpt in computer programming. *Mesopotamian Journal of Computer Science*, 2023:8–16, 2023.

Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

Etay Buber and Dragos-Ioan Banu. Performance analysis and cpu vs gpu comparison for deep learning. In *2018 6th International Conference on Control Engineering & Information Technology (CEIT)*, pages 1–6. IEEE, 2018.

Shyang-Lih Chang, Li-Shien Chen, Yun-Chung Chung, and Sei-Wan Chen. Automatic license plate recognition. *IEEE transactions on intelligent transportation systems*, 5(1):42–53, 2004.

G Chassagnon, M Vakalopolou, N Paragios, and MP Revel. Deep learning: definition and perspectives for thoracic imaging. *European radiology*, 2020.

Sylvain Christin, Éric Hervet, and Nicolas Lecomte. Applications for deep learning in ecology. *Methods in Ecology and Evolution*, 10(10):1632–1644, 2019. doi: https://doi.org/10.1111/2041-210X.13256. URL https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/2041-210X.13256.

Ulysse Côté-Allard, Cheikh Latyr Fall, Alexandre Campeau-Lecours, Clément Gosselin, François Laviolette, and Benoit Gosselin. Transfer learning for semg hand gestures recognition using convolutional neural networks. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1663–1668. IEEE, 2017.

Li Deng. A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing*, 3:e2, 2014. doi: 10.1017/atsip.2013.9.

Ashwathy Dev. A novel approach for car license plate detection based on vertical edges. In *2015 Fifth International Conference on Advances in Computing and Communications (ICACC)*, pages 391–394. IEEE, 2015.

Syed Ali Haider and Khurram Khurshid. An implementable system for detection and recognition of license plates in pakistan. In *2017 International Conference on Innovations in Electrical Engineering and Computational Technologies (ICIEECT)*, pages 1–5. IEEE, 2017.

H. Hassani and E. S. Silva. The role of chatgpt in data science: how ai-assisted conversational interfaces are revolutionizing the field. *Big data and cognitive computing*, 7(2):62, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Daniel Isereau, Christopher Capraro, Eric Cote, Mark Barnell, and Christopher Raymond. Utilizing high-performance embedded computing, agile condor, for intelligent processing: An artificial intelligence platform for remotely piloted aircraft. In *2017 Intelligent Systems Conference (IntelliSys)*, pages 1155–1159. IEEE, 2017. doi: 10.1109/IntelliSys.2017.8324277.

Yuhan Jia, Jianping Wu, and Yiman Du. Traffic speed prediction using deep learning method. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1217–1222, 2016. doi: 10.1109/ITSC.2016.7795712.

R. Colin Johnson. Neural learning on the edge. *Communications of the ACM*, page 3, 2019. URL https://cacm.acm.org/news/234063-neural-learning-on-the-edge/fulltext.

Pejman Khadivi, Ravi Tandon, and Naren Ramakrishnan. Flow of information in feed-forward deep neural networks, 2016.

Dong-Su Kim and Sung-Il Chien. Automatic car license plate extraction using modified generalized symmetry transform and image warping. In *ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No. 01TH8570)*, volume 3, pages 2022–2027. IEEE, 2001.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Ernest Kisingo, Ndyetabura Hamisi, Hashim U Iddi, and Baraka J Maiseli. Multi-vehicle speed estimation algorithm based on real-time inter-frame tracking technique. *Tanzania Journal of Science*, 47(3):1125–1137, 2021.

Rafael Laroca, Everton Severo, Luiz A Zanlorensi, Luiz S Oliveira, Gabriel R Gonçalves, William R Schwartz, and David Menotti. A robust real-time automatic license plate recognition based on the yolo detector. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10. IEEE, 2018.

Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature Cell Biology*, 521(7553): 436–444, May 2015. ISSN 1465-7392. doi: 10.1038/nature14539. Funding Information: Acknowledgements The authors would like to thank the Natural Sciences and Engineering Research Council of Canada, the Canadian Institute For Advanced Research (CIFAR), the National Science Foundation and Office of Naval Research for support. Y.L. and Y.B. are CIFAR fellows. Publisher Copyright: © 2015 Macmillan Publishers Limited. All rights reserved.

Yen-Lin Lee, Pei-Kuei Tsung, and Max Wu. Techology trend of edge ai. pages 1–2, 04 2018. doi: 10.1109/VLSI-DAT.2018.8373244.

Cheng-Jian Lin, Shiou-Yun Jeng, and Hong-Wei Lioa. A real-time vehicle counting, speed estimation, and classification system based on virtual detection zone and yolo. *Mathematical Problems in Engineering*, 2021:1–10, 2021.

Kaizhan Liu, Yunming Ye, Xutao Li, and Yan Li. A real-time method to estimate speed of object based on object detection and optical flow calculation. In *Journal of Physics: Conference Series*, volume 1004, page 012003. IOP Publishing, 2018a.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016.

Yujie Liu, He Huang, Jinde Cao, and Tingwen Huang. Convolutional neural networks-based intelligent recognition of chinese license plates. *Soft Computing*, 22(7):2403–2419, 2018b.

Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. Traffic flow prediction with big data: A deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865–873, 2015. doi: 10.1109/TITS.2014.2345663.

Polina Mamoshina, Armando Vieira, Evgeny Putin, and Alex Zhavoronkov. Applications of deep learning in biomedicine. *Molecular Pharmaceutics*, 13(5):1445–1454, 2016. doi: 10.1021/acs.molpharmaceut.5b00982. URL https://doi.org/10.1021/acs.molpharmaceut.5b00982. PMID: 27007977.

Jiri Matas and Karel Zimmermann. Unconstrained licence plate and text localization and recognition. In *Proceedings. 2005 IEEE Intelligent Transportation Systems, 2005.*, pages 225–230. IEEE, 2005.

Sergio Montazzolli and Claudio Jung. Real-time brazilian license plate detection and recognition using deep convolutional neural networks. In *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 55–62. IEEE, 2017.

Nabeel Mufti and Syed Ali Shah. Automatic number plate recognition: A detailed survey of relevant algorithms. *Sensors*, 21(9):3028, 2021.

Muhammad S Murshed, Colm Murphy, Ding Hou, Nabeel Khan, Ganesh Ananthanarayanan, and Fakhra Hussain. Machine learning at the network edge: A survey. *ACM Computing Surveys (CSUR)*, 54(8):1–37, 2021.

Paul-Emmanuel Novac, Ghada Boukli Hacene, Adrien Pegatoquet, Benjamin Miramond, and Vincent Gripon. Quantization and deployment of deep neural networks on microcontrollers. *Sensors*, 21(9):2984, 2021.

Liang Pang, Bin Li, Fan Zhang, Xiangyu Meng, and Liangpei Zhang. A lightweight yolov5-mne algorithm for sar ship detection. *Sensors*, 22(18):7088, 2022.

Jamshid Pirgazi, Mohammad Mehdi Pourhashem Kallehbasti, and Ali Ghanbari Sorkhi. An end-to-end deep learning approach for plate recognition in intelligent transportation systems. *Wireless Communications and Mobile Computing*, 2022:1–13, 01 2022. doi: 10.1155/2022/3364921.

K Prazdny. Determining the instantaneous direction of motion from optical flow generated by a curvilinearly moving observer. *Computer Graphics and Image Processing*, 17(3):238–248, 1981.

Muhammad Tahir Qadri and Muhammad Asif. Automatic number plate recognition system for vehicle identification using optical character recognition. In *2009 International Conference on Education Technology and Computer*, pages 335–338, 2009. doi: 10.1109/ICETC.2009.54.

Xu Qimin, Li Xu, Wu Mingming, Li Bin, and Song Xianghui. A methodology of vehicle speed estimation based on optical flow. In *Proceedings of 2014 IEEE International Conference on Service Operations and Logistics, and Informatics*, pages 33–37, 2014. doi: 10.1109/SOLI.2014.6960689.

Raspberry Pi Trading Ltd. Raspberry pi 4 model b. [Online]. Available: https://www.amazon.com/Raspberry-Model-2019-Quad-Bluetooth/dp/B07TC2BK1X, 2019.

Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

Joachim H Rieger and Daryl T Lawton. Determining the instantaneous axis of translation from optic flow generated by arbitrary sensor motion. Technical report, MASSACHUSETTS UNIV AMHERST DEPT OF COMPUTER AND INFORMATION SCIENCE, 1983.

A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM J. Res. Dev.*, 3(3):210–229, jul 1959. ISSN 0018-8646. doi: 10.1147/rd.33.0210. URL https://doi.org/10.1147/rd.33.0210.

Todd Nelson Schoepflin. *Algorithms for estimating mean vehicle speed using uncalibrated traffic management cameras*. University of Washington, 2003.

Nigar M. Shafiq Surameery and Mohammed Y. Shakor. Use chat gpt to solve programming bugs. *International Journal of Information Technology & Computer Engineering (IJITC) ISSN : 2455-5290*, 3(01):17–22, 2023.

Nusrat Sharmin and Remus Brad. Optimal filter estimation for lucas-kanade optical flow. *Sensors*, 12(9):12694–12709, 2012.

Xifan Shi, Weizhong Zhao, and Yonghang Shen. Automatic license plate recognition system based on color image processing. In *Computational Science and Its Applications–ICCSA 2005: International Conference, Singapore, May 9-12, 2005, Proceedings, Part IV 5*, pages 1159–1168. Springer, 2005.

Ibtissam Slimani, Abdelmoghit Zaarane, Abdellatif Hamdoun, and Issam Atouf. Vehicle license plate localization and recognition system for intelligent transportation applications. In *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 1592–1597. IEEE, 2019.

Xiaojing Song, Lakmal D Seneviratne, Kaspar Althoefer, and Zibin Song. Vision-based velocity estimation for unmanned ground vehicles. *International Journal of Information Acquisition*, 4 (04):303–315, 2007.

Xiaojing Song, Zibin Song, Lakmal D. Seneviratne, and Kaspar Althoefer. Optical flow-based slip and velocity estimation technique for unmanned skid-steered vehicles. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 101–106, 2008. doi: 10.1109/IROS.2008.4651025.

Fahd Sultan, Khurram Khan, Yasir Ali Shah, Mohsin Shahzad, Uzair Khan, and Zahid Mahmood. Towards automatic license plate recognition in challenging conditions. *Applied Sciences*, 13(6), 2023. ISSN 2076-3417. doi: 10.3390/app13063956. URL https://www.mdpi.com/2076-3417/13/6/3956.

Chellammal Surianarayanan, John Lawrence, Pethuru Raj Chelliah, Edmond Prakash, and Chaminda Hewage. A survey on optimization techniques for edge artificial intelligence (ai). *Sensors*, 23:1279, 01 2023. doi: 10.3390/s23031279.

Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57:137–154, 2004.

Wikipedia contributors. Radar speed gun — Wikipedia, the free encyclopedia, 2023a. URL https://en.wikipedia.org/w/index.php?title=Radar_speed_gun&oldid=1143413944. [Online; accessed 26-April-2023].

Wikipedia contributors. Lidar traffic enforcement — Wikipedia, the free encyclopedia, 2023b. URL https://en.wikipedia.org/w/index.php?title=LIDAR_traffic_enforcement&oldid=1149224528. [Online; accessed 27-April-2023].

Wikipedia contributors. Vehicle registration plate — Wikipedia, the free encyclopedia, 2023c. URL https://en.wikipedia.org/w/index.php?title=Vehicle_registration_plate&oldid=1151907731. [Online; accessed 30-April-2023].

Zhenbo Xu, Wei Yang, Ajin Meng, Nanxue Lu, and Huan Huang. Towards end-to-end license plate detection and recognition: A large dataset and baseline. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 255–271, 2018.

Jia Yao, Jiaming Qi, Jie Zhang, Hongmin Shao, Jia Yang, and Xin Li. A real-time detection algorithm for kiwifruit defects based on yolov5. *Electronics*, 10(14):1711, 2021.

Huaifeng Zhang, Wenjing Jia, Xiangjian He, and Qiang Wu. Learning-based license plate detection using global and local features. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 2, pages 1102–1105. IEEE, 2006.

Xunxun Zhang and Xu Zhu. Vehicle detection in the aerial infrared images via an improved yolov3 network. In *2019 IEEE 4th International Conference on Signal and Image Processing (ICSIP)*, pages 372–376. IEEE, 2019.

Xinyi Zhou, Wei Gong, WenLong Fu, and Fengtong Du. Application of deep learning in object detection. In *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, pages 631–634, 2017. doi: 10.1109/ICIS.2017.7960069.