University of Western Australia

Faculty of Engineering and Mathematical Sciences

School of Electrical, Electronic and Computer Engineering

SIFT-like Keypoint Cluster-Based Traffic Sign Recognition with Deep Learning

Author:

Jordan King (21118935)

Supervisor:

Prof. Thomas Bräunl

School of Electrical, Electronic and Computer Engineering

A thesis submitted in partial completion of the requirements for the degree of

Master of Professional Engineering

(Electrical and Electronic)

at

University of Western Australia

October 2019

Word Count: 7994

Thesis Declaration

I, Jordan King, certify that:

This thesis has been substantially accomplished during enrolment in this degree.

This thesis does not contain material which has been submitted for the award of any other degree or diploma in my name, in any university or other tertiary institution.

In the future, no part of this thesis will be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of The University of Western Australia and where applicable, any partner institution responsible for the joint-award of this degree.

This thesis does not contain any material previously published or written by another person, except where due reference has been made in the text and, where relevant, in the Authorship Declaration that follows.

This thesis does not violate or infringe any copyright, trademark, patent, or other rights whatsoever of any person.

This thesis contains published work and/or work prepared for publication, some of which has been co-authored.

Signature:

Date:

Abstract

State-of-the-art convolutional neural network-based architectures similar to MobileNet were shown to have the capability to surpass human performance in the German Traffic Sign Recognition Benchmark (GTSRB) held for IJCNN 2011; however, traffic sign detection systems that operate in real-time for ARM-based devices with limited computational power (e.g., less than 200 MFLOPS) have remained elusive. In this paper, a traffic sign detection pipeline is proposed for inexpensive autonomous driving robots and driver-assistance systems using a novel approach for ROI candidate generation with SIFT-like keypoint cluster-generated regions and TensorFlow. The pipeline has been implemented to detect a subset of the traffic signs present in the Carolo Cup, and has been deployed on mobile driving robots at UWA using the EyeBot 7 controller platform and a Raspberry Pi 3B with MobileNet for image classification. The pipeline is highly configurable can operate at an average of 18.80 Hz for QVGA-sized true-colour video if detecting a subset of signs, achieve an accuracy of 89.10% in the GTSRB, and yield an IoU of 0.7273 for a modified KITTI semantic segmentation dataset. The novel detection architecture brings feasibility to inexpensive monocular driver-assistance systems or autonomous driving robots, but the skeleton may be repurposed. The ROI proposal method lend itself to applications where computational power is limited, SIFT-like keypoints are already generated, or when the environment is sparse; for example, when using inexpensive ARM-based devices such as Raspberry Pi, if the robot is already using ORB-SLAM, or in underwater environments.

Contents

1 Introduction

Spearheaded by initiatives such as the Renewable Energy Vehicle (REV) Project and the Carolo Cup, the future of mobility will precipitate two major technological revolutions: electric cars, and then autonomous cars. The integration of fully-autonomous vehicles for real-world, complex scenarios has captured the attention of researchers from many fields, industry, and the general public; however, the tremendous benefits of state-of-the-art advances in robotics and computer vision are not exclusive to fully-autonomous cars.

SAE International, otherwise known as the Society of Automotive Engineers, have defined six levels of driving automation in their J3016 standard that encapsulate a range of automation levels including no automation, Advanced Driver-Assistance Systems (ADAS), and full self-driving automation [1]. Vehicle autonomy is not solely related to the capability of the vehicle to drive without human intervention, but may also relate advanced technology that integrates a given aspect of intelligence.

In this paper, the topic of traffic sign detection and recognition is explored and three different methods are proposed. The aim is to develop a traffic sign detection and recognition system that can be deployed to participate in the Carolo Cup autonomous driving student competition, but is also scalable to the real-world. In theory, the developments could ported to a mobile phone, or be used as the module in an inexpensive monocular driver assistance system.

1.1 The Carolo-Cup

The Carolo-Cup, organised by the Technischen Universität Braunschweig (in English: Technical University of Braunschweig, Germany), is an international competition for students to implement solutions for autonomous vehicles in areas such as lane following, parking on a sideways parking strip, and obstacle detection and avoidance [2].

This competition and others similar offer a platform to develop solutions for autonomous vehicles that are not practical or safe to implement on the real-scale vehicles. In the scope of sign recognition, there are 17 signs that are introduced as can be seen below.

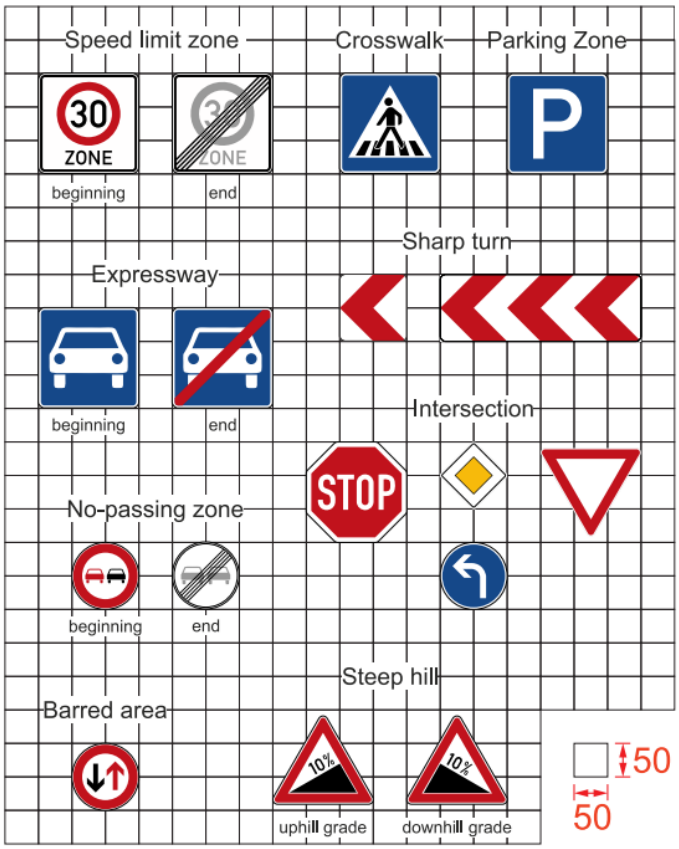

Figure 1: The traffic signs in the Carolo-Cup [3]

There are two categories of events in the Carolo Cup: static events and dynamic events. In the dynamic events, the performance of the driving robots is evaluated with respect to the Free Drive and Parking discipline, and the Obstacle Evasion Course.

Category 1: Free Drive and Parking

In the first dynamic event of the Carolo Cup, vehicles autonomously traverse the furthest possible track distance in a given time [4]. In addition to lane following, the cars are evaluated with respect to their capacity to detect and recognise parking signs.

If a parking zone is identified, the car's capability to perform parallel and perpendicular parking is assessed.

Category 2: Obstacle Evasion Course

The second of the dynamic events adds both static and dynamic obstacles to the road, and removes the parking manoeuvre challenges.

Team KitCar won the Basic Cup in 2018 with entry *Dr. Drift* [5]. There are no papers that are immediately available that may be used to compare their implementation of traffic sign recognition and detection to that proposed in this paper.

1.2 Robot architecture

EyeBot is an embedded controller platform that has been developed by Prof. Bräunl and the Robotics and Automation lab at UWA for a family of small autonomous mobile robots which include driving robots, walking robots, autonomous airplanes and autonomous underwater vehicles [6] [7] [8]. The current iteration of robots, Eyebot 7, employs a Raspberry Pi 3B+ single-board computer for high-level control of the robot, connecting with additional sensors such as LiDAR or IMU, and user-interfacing [7].

The Eyebot I/O USB expansion board version 2.63, called the Eyebot I/O Board, is physically linked to the Pi via USB, and RoBIOS-7 software installed on the Pi allows the Pi to employ the I/O board functionality [6]. The I/O board allows for the user to control low-level function of the robot including motors, servos and PSDs [6].

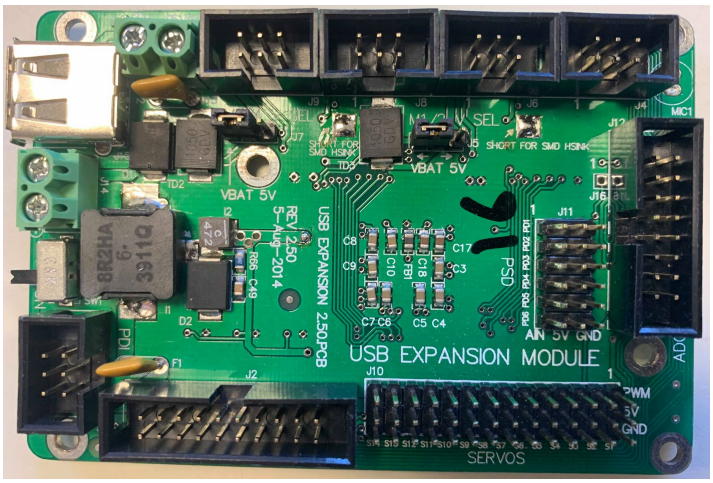

Figure 2: TheEyeBot7 I/O Board

The I/O board may be deployed in a host of robot builds; however, the robot that is the focus of the thesis will be the SoccerBot as shown below. The construction of the SoccerBot is described in tremendous detail in [6]. The SoccerBot is a differential-drive, wheeled robot equipped with an LCD screen; PSDs located on the left-hand side, right-hand side, and front of the robot; and a PiCamera attached on top of a servo [6].

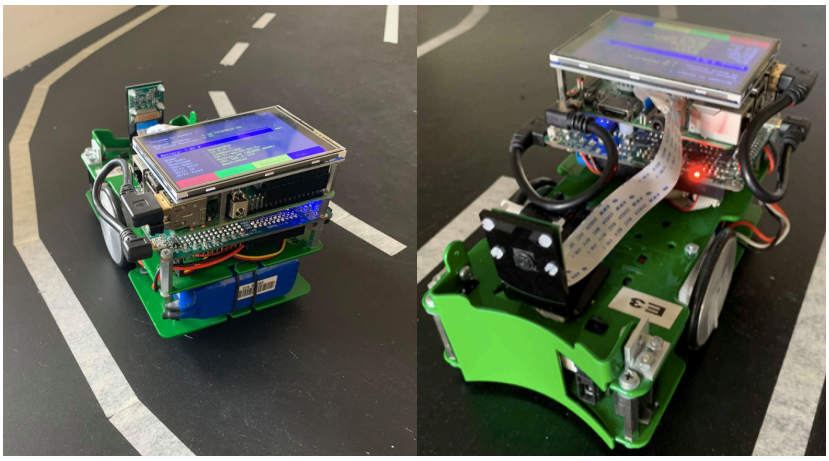

Figure 3: SoccerBot mobile driving robot

1.3 Problem Statement

This work is written as part of the research conducted by the Robotics and Automation Lab at UWA; specifically, the autonomous driving robots research group that has been assembled to complete the tasks presented by the Carolo Cup competition.

The sign recognition, parking, and obstacle avoidance challenges are contingent on the ground robots being able to detect traffic signs and obstacles. A traffic sign recognition that operates in real-time with inexpensive and computationally limited hardware—e.g., the Raspberry Pi 3B—is the underlying goal of this work

In terms of hardware compatibility, it may only use the existing sensors on the robot: a front-facing camera and front, left and right-facing PSDs. In terms of software compatibility, it must be able to be run on the robot in conjunction with the lane-detection program developed for the Carolo-Cup competition.

To operate in real-time, the system must be able to run at a minimum of 8Hz for a QVGA-size (320x240 pixels) image. If feasible, any solution should endeavour to operate at a higher rate than 8Hz or use a resolution greater than QVGA.

In terms of the larger body of literature at UWA, the work aims to be scalable and to have the capability to be implemented in real-world environments. In order to achieve this goal, benchmarks are discussed with respect to real-world datasets and state-of-the-art techniques and methods.

1.4 Document Structure

In this first chapter, the background information, goal and context for the thesis project has been defined thereby forming the basis for the rest of the document.

The related work and theory are discussed in-depth in Chapter 2. It includes an analysis of current state-of-the-art methods and information to provide a backdrop to interpret the rest of the paper.

An outline of the overall architecture and pipeline of the traffic sign recognition and detection system is provided in Chapter 3.

The techniques and methods for region proposal are proposed in Chapter 4. There are three techniques discussed in detail including keypoint-dense region proposal and binary semantic image segmentation.

Chapter 5 will delve into the recognition module proposed and provide a significant level of insight relating to benchmarking CNNs on the Raspberry Pi 3B hardware.

Chapter 6 serves to discuss the localisation and reliability of the detections, and what is termed 'profiling' which is a essentially a time-wise committee of CNNs.

Chapter 7 is the conclusion of the thesis and includes a reflection on the results, an outlook and any recommendations or any modifications to method.

2 Related Work and Theory

In this section of the paper, a review is completed of the literature relating to the following: (i) traffic sign recognition approaches, and (ii) deep neural network-based object detection approaches. The two sets of approaches may be referred later in the paper as 'specialised' traffic sign recognition methods or 'generic' object detection methods respectively.

It should be noted by the reader that the majority of the literature review is completed in this section; however, for comparison purposes additional sources will be referred to in the corresponding chapters.

2.1 Traffic Sign Recognition Methods

In-depth overviews of specialised traffic sign detection and recognitions approaches are available in the following papers: [9] [10] [11]. In the interest of completion, the author has opted to include a brief overview.

Specialised methods for traffic sign recognition are frequently composed of three stages or processes: pre-processing, detection, and recognition [11].

The methods of traffic sign detection are traditionally divided into colour-based, shape-based, and learning-based approaches [9] [10]. Learning-based methods can be further sub-divided into those that employ deep learning techniques such as CNNs.

It is not feasible to discuss an implementation of each approach, but they will be touched upon when comparing results in Chapter 5. The rest of this section will focus on a machine-learning approach that has seen a significant amount of success: Real-Time Detection and Recognition of Road Traffic Signs [12]. Similar methods were employed to win the German Traffic Sign Detection Benchmark, and it is possible that similar methods are used in conjunction with GPS data in industry.

2.1.1 Support-Vector Machine of Histogram of Orientated Gradient Features

In Greenhalgh and Mirmehdi's paper (2012), a real-time approach for traffic sign detection is proposed with traffic signals being recognised using Histogram of Gradients (HOG) features, and a cascade of linear Support Vector Machine (SVMs) classifiers [12].

The proposed method can be divided into a detection and recognition stage. In the detection stage, the image undergoes thresholding at a range of levels. If a region maintains its shape then it is designated as a maximally stable extremal region (MSER). [12]

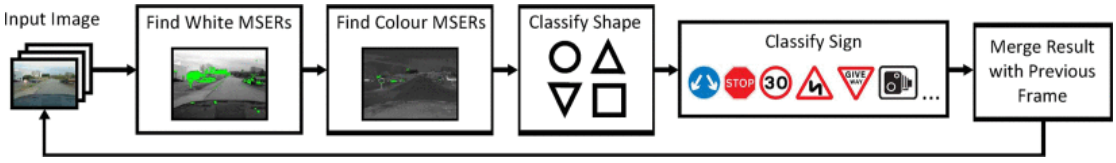

Figure 4: The flow diagram of the proposed solution [12]

These regions are then classified using a cascade of multiclass SVMs [12]. Support vector machines are simple algorithms that can be used to classify an object after it has been given a set of supervised training examples. In simple terms, it optimally specifies a line, otherwise known as a hyperplane, that divides a multidimensional space containing descriptions of different object classes. A HOG vector is used to classify the shape of the region.

There are algorithms similar to that presented in this paper [12]—that is, using HOGs and SVM, or employing MSERs—that will be used for evaluation and comparison purposes later in the paper. The paper is one of the most heavily cited sources on traffic sign detection, and it is possible that ADAS manufacturers may employ related methods for real-time detection.

2.2 Deep Learning-based Approaches

2.2.1 Neural Networks

An artificial neural network (ANN), otherwise referred to as a neural network (NN), is a multivariate statistical model that broadly simulates the structure of a brain [13] [14]. They are composed of simple units, called 'neurons', divided into layers and interlinked by weight connections and biases [15]. In the analogy, the input to each 'neuron' represents the dendrite and the output from the neuron represents the axon terminals. The input layer of the ANN contains an explanatory tensor that is used to calculate a dependent tensor contained in the output layer [13].

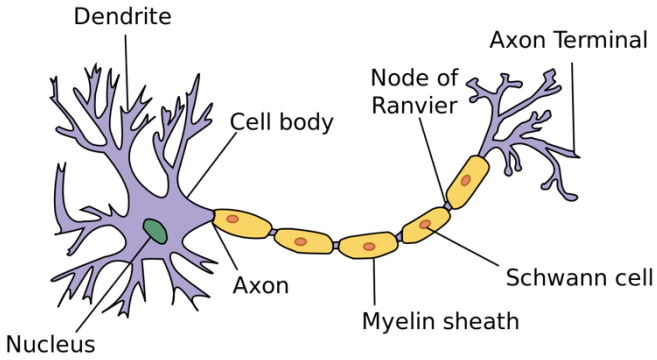

Figure 5: The anatomy of a neuron via Wikimedia Commons, used under the Creative Commons Attribution-Share Alike 3.0 Unported license

Neural networks lie in the field of machine learning, and can be considered a general purpose mathematical model. At a high-level, a neural network can be trained via 'supervised learning'; that is, by providing a set of training data and its corresponding answer. They are differentiable and therefore if there is any error, the error can be backpropagated and weights amended. In order to grasp the inner workings of an ANN, it is imperative to understand what is happening at the neuron-level.

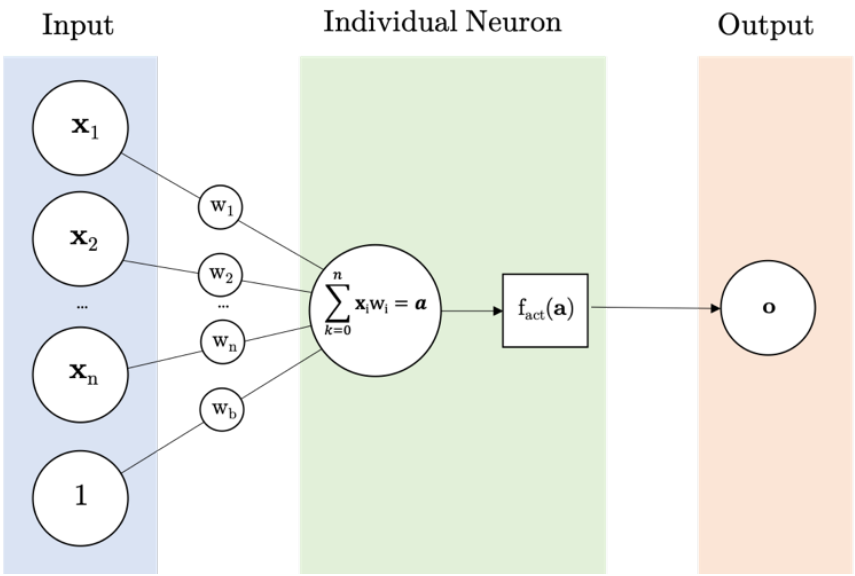

Figure 6: Representation of a singular neuron

In the Perceptron model [16], each neuron in a hidden layer—that is, not in the input or output layer—is connected to every single neuron in the layer before and a bias by a weight. The output of each neuron is equal to the activation function applied to the value stored within the neuron itself. There are a variety of activation functions that exist, as will be discussed in the following sections.

2.2.2 Convolutional Neural Networks

Convolutional neural network layers differ from the fully-connected neural networks in how each neuron in the hidden layers is calculated. In a convolutional layer, weights are assigned to a kernel, frequently referred to as a filter, rather than between each neuron-pair.

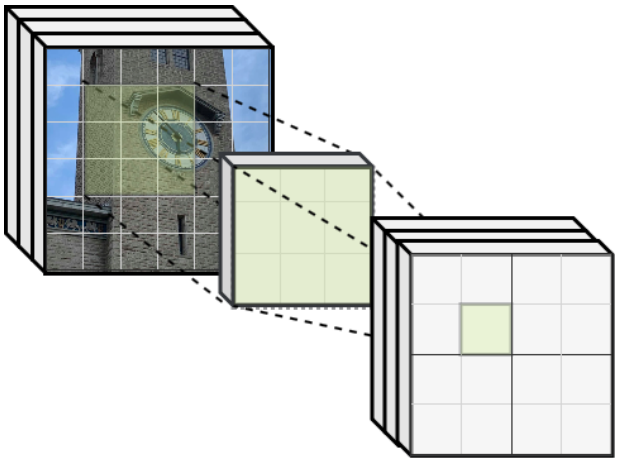

Figure 7: A demonstration on how convolutional layers work in a CNN using an image of UWA's Winthrop Hall as an example

The kernel size and number of kernels in a convolutional layer are subject to the choice of the developer. The kernel is convolved over the input tensor according to a given stride length, and the sum of each product between the input tensor and the kernel weights corresponds to the neuron value in the next proceeding layer.

2.2.3 Activation Functions

The Perceptron model and convolutional neural networks both employ activation functions. The purpose of an activation function can be reasoned as two-fold: they introduce non-linearity to the network, and they can they can be set to encourage convergence during training. If an activation function were not used, then the network would default to linear regression which may or may not be useful depending on the application.

The activations functions discussed in the paper will be limited to Sigmoid, ReLU, Leaky-ReLU and Softmax.

Sigmoid:

$$f_{act}(x) = \frac{1}{1 - e^{-x}}$$

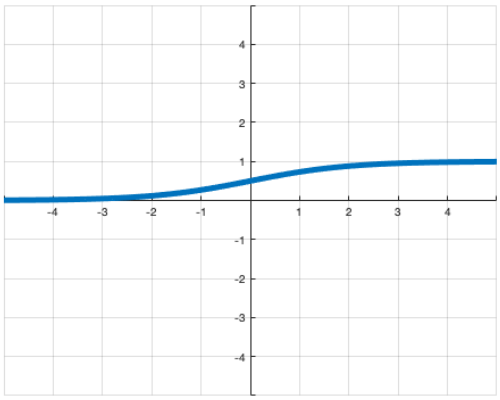

Figure 8: Visualisation of the sigmoid activation function

Rectified Linear Unit (ReLU):

$$f_{act}(x) = \begin{cases} x, & x \geq 0 \\ 0, & o.w. \end{cases}$$

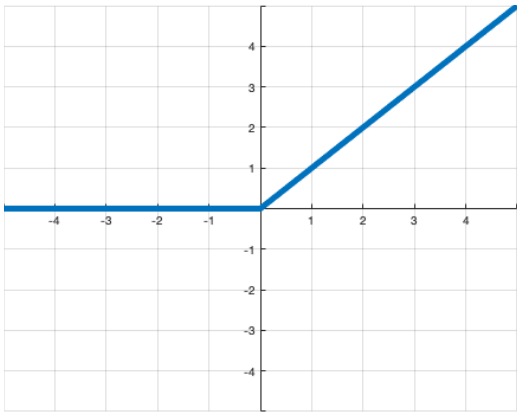

Figure 9: Visualisation of the Rectified Linear Unit (ReLU) activation function

Leaky-ReLU:

$$f_{act}(x) = \begin{cases} x, & x \geq 0 \\ ax, & o.w. \end{cases}$$

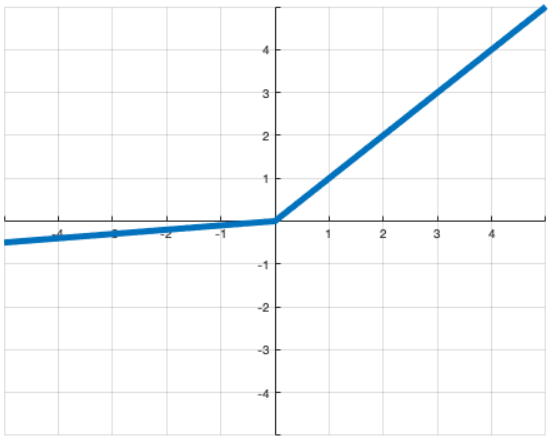

Figure 10: Visualisation of the Leaky-ReLU Activation Function

Softmax:

$$f_{act}(\boldsymbol{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$$

Softmax is frequently used as the final activation function in classifier architectures. The value output is dependent not only on the input, but all neurons in the preceding layer.

2.2.4 Pooling Layers and Upsampling

There are a wide array of pooling and upsampling layers, but the types employed in this thesis have been max pooling, mean pooling, and upsampling.

Max Pooling:

An N-pool size max pooling layer ascertains the maximum local value in a N-N region and uses that value to populate the next layer which will be 1/Nth the size of the input layer.

Mean Pooling:

Similar to max pooling above, but the mean local value populates the proceeding layer instead of the maximum value.

Upsampling:

An N-sized upsampling layer will repeat each row and column of data N times in the output layer effectively increasing the width and height of the layer.

2.2.5 MobileNet

Howard et al. proposed MobileNets as a class of lightweight convolutional neural networks based on the principle of depthwise separable convolutional layers [17]. The key contribution of the paper was the implementation of depthwise separable convolutional layers to dramatically decrease model size and the number of multiply-accumulate operations required for inference.

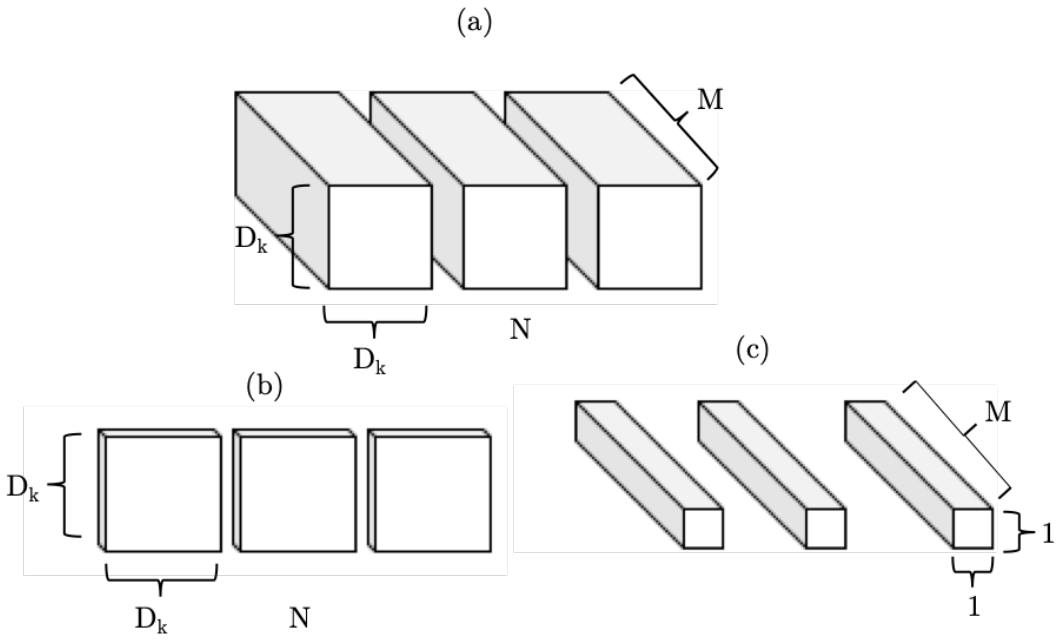

Figure 11: (a) Typical kernels for CNNs, referred to in this paper as standard convolution (b) Depth-wise kernels. (c) Point-wise kernels. D_k is the width of the kernel, M is the depth of the input layer, N is the depth of the output layer [17].

A depthwise separable convolution layer is not a single convolution but a depthwise convolution followed by a pointwise convolution. A 3×3 depthwise separable layer has the theoretical underpinnings to run 8 to 9 times faster than a standard a 3×3 convolutional layer of equal size with only a small loss to accuracy [17].

The structure of a MobileNet is given in Table 1 below. MobileNet employs ReLU activation functions for the hidden layers, and Softmax at the end of the network.

Table 1: The Body Architecture of MobileNet [17]

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

2.2.6 Region-proposal CNNs and Single-Stage Object Detection

The advent of the region-proposal family of convolutional neural networks has dramatically increased the prevalence of CNN-based detectors in the object recognition sphere. For a more in-depth review of the state-of-the-art in CNNs, the author would recommend reading [18] [19] [20] [21].

The rest of this section will entail a discussion about single-stage CNNs that can be deployed to low-power and low-cost embedded devices for real-time large-scale object detection. They operate at real-time for a slew of devices including mobile phones and a range of embedded devices such as Jetson Nano.

As opposed to the R-CNN family of detectors, the You Only Look Once (YOLO) model [22], RetinaNet [23], and the Single Shot MultiBox Detector (SSD) [24] do not propose a set of ROIs using a Selective Search algorithm. In these architectures, an image in

processed via a single CNN which provides a confidence for a classification as well as bounding box coordinates for any likely objects.

The relatively computationally-limited nature of the Raspberry Pi 3B means that it is not possible to simply use an object detection architecture such as SSD MobileNet-V2 [24] [17]. Nvidia have performed benchmarks for the Raspberry Pi 3 and similar single-board computers and found that for inputs with resolution 480×272 or greater that SSD MobileNet-V2 did not run [25]. Input images of 300×300 on SSD MobileNet-V2 ran at 1 Hz, and 416×416 Tiny YOLO V3 is able to run at 0.5 Hz [25]. Alternatively, using image classification instead of object detection, MobileNet-V2 can be run at 2.5 Hz for 300×300 images [25].

2.2.7 SegNet

SegNet is an encoder-decoder fully convolutional neural network architecture for pixelwise semantic segmentation; that is, it attempts to label each pixel in a given image according to a set of classes [26].

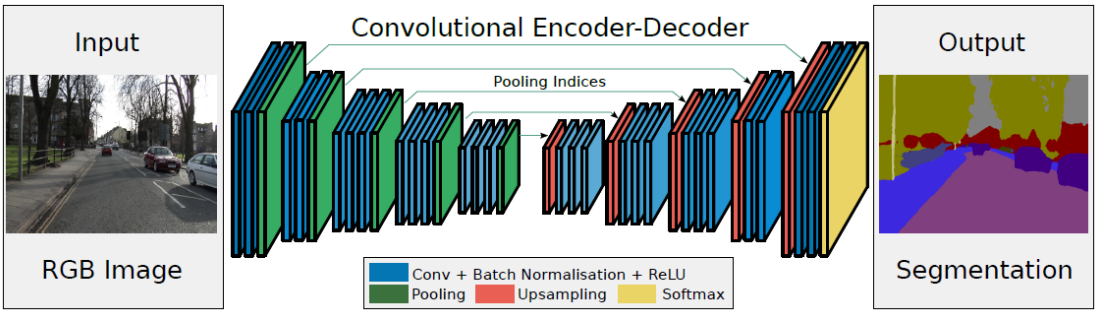

Figure 12: The architecture of SegNet [26]

The encoder-decoder structure of the net is practical for two primary reasons: it will aid convergence and it has the capability to run in real-time on high-end GPUs. SegNet relies heavily on the pooling and upsampling layers that were discussed in Section 2.2.4.

3 Methodology

The following chapter will provide a high-level overview of the design methodology. The subsequent chapters will provide an in-depth explanation of each module; however, it will be beneficial to the reader to understand the interaction between each module and the role they play individually.

3.1 The Founding Principle, or *Arche*

The field of robotic vision largely overlaps with that of computer vision, but the two differ with respect to their overarching goal: in the domain of robotic vision, problems operate in real-time, run in parallel to other software, and should prioritise reliability. In this capacity, the traditional computer vision-orientated object detection pipeline may not be optimal for robotic vision tasks.

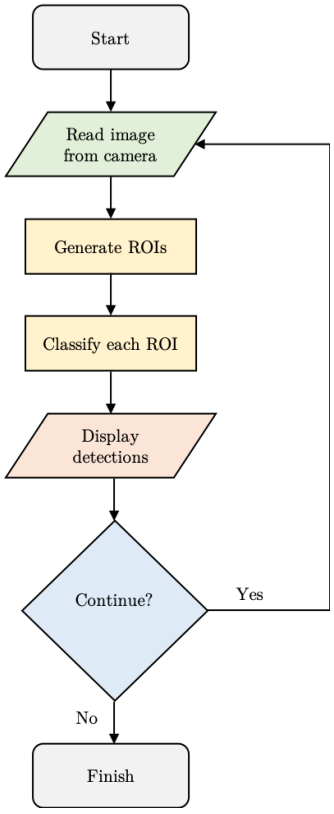

Figure 13: The traditional neural network-based object detector pipeline

The pipeline proposed in this paper seeks to remedy two aspects of the traditional object detection pipeline. The proposed architecture seeks to build up a 'detection profile' for each given object. When an object is detected, rather than the robot erasing data about the object in the next frame, objects are tracked between frames and each detection of the object is added to its detection profile to measure the 'reliability' of that object being detected correctly. It can be considered similar to a time-wise committee of convolutional neural network classifiers and increases redundancy.

3.2 Translation and Rotational Invariant Keypoints

The key property of SIFT-like keypoints that may be exploited for this problem is their ability to be tracked between frames irrespective of scale and transform variance. They may naïvely be understood as a corner with a given orientation and magnitude

The property of visual salience loosely corresponds to how well a region stands out and gains visual attention. It may be subdivided into high-level and low-level visual salience. An example of high-level visual salience would be the colour red: as a species, humans are evolutionary hardcoded to view the colour red and think danger.

Corners may be considered an example of low-level visual salience. A count of the number of SIFT-like keypoints in a region can be used as a metric to order ROIs.

3.3 Module Interaction

The pipeline begins by reading an image from the camera sensor. A SIFT-like feature detector then calculates all the keypoints in the image for a given set of parameters. These will act as a metric for low-level visual salience.

In the next stage, region proposal occurs: the true-colour image is processed and a binary image is generated. This binary image will be used to represent high-level visual

saliency. The binary image will be used to cluster the previously generated keypoints, and then each of these clusters is used to specify a region of interest (ROI). A list of ROIs is generated and ordered by the priority. The number of ROIs classified in each frame is limited because a high-accuracy image classifier is computationally expensive. The second purpose to clustering keypoints is to constrain matches. Keypoint matching between frames and tracking object detections is computationally expensive if each keypoint from a frame is trying to match with each keypoint from another frame. To reduce the power required and to ensure that the system operates in real-time, keypoint matches between frames are constrained to each nearby cluster.

The tracking and localisation module itself plays a role before classification occurs for each frame and essentially supresses any ROIs that likely contain objects that have been detected and classified recently. As mentioned previously, keypoints are matched between nearby clusters between frames; therefore, the set of possible keypoints are constrained and the program is able to run at a greater FPS. When a cluster is identified as belonging to an existing detection profile, the bounding box describing the location of the detection is updated. If sufficient time has passed since the last detection, then the new ROI is classified and the detection profile updated. If an object can no longer be tracked – for example, if the robot has moved and the traffic sign is no longer in view—then the detection profile is erased.

3.4 The Keypoint Cluster-Based Object Detection Pipeline

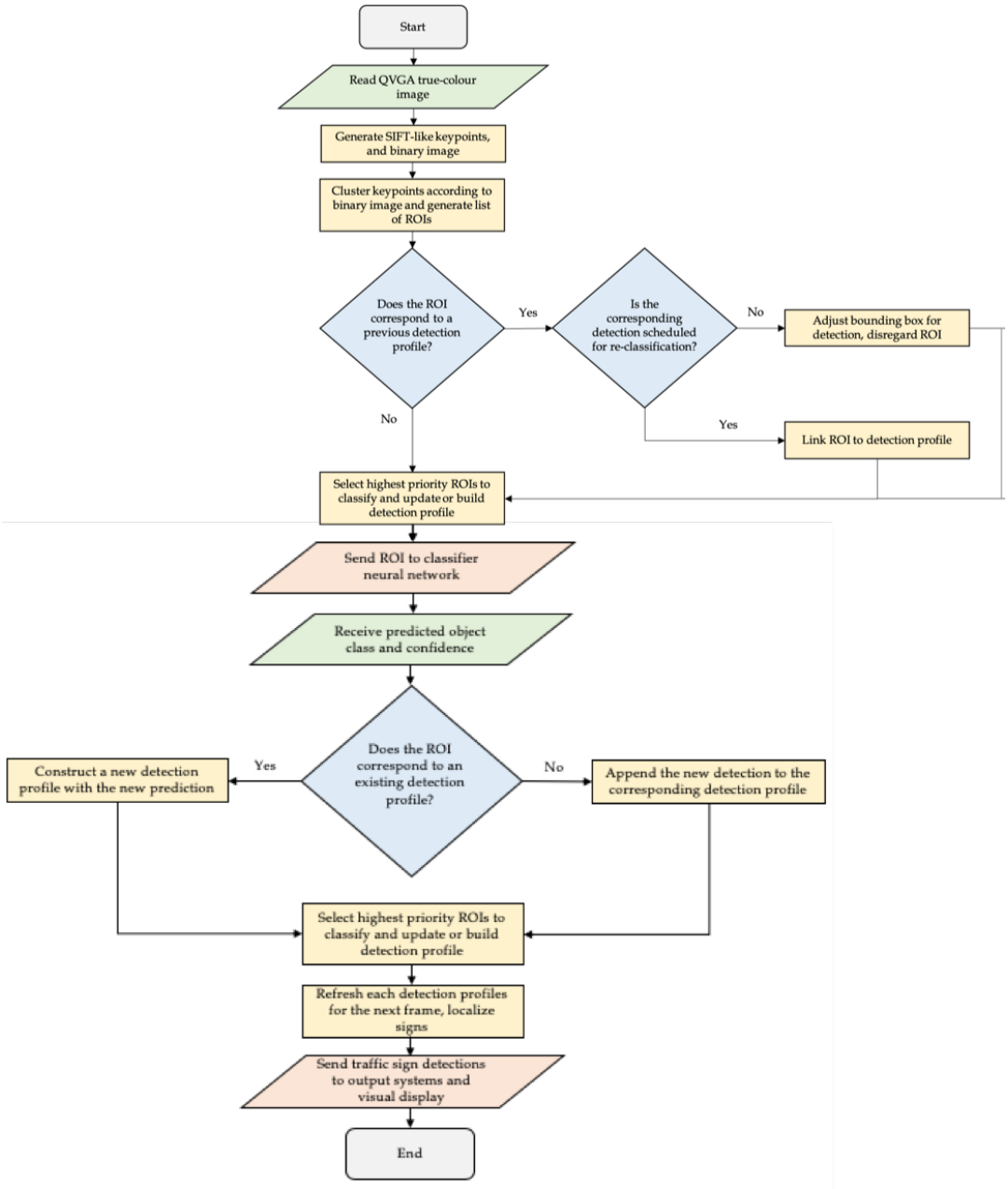

Figure 14: The proposed pipeline for traffic sign detection and recognition

4 Region Proposal

In this section of the paper, three methods employed for region proposal throughout the project will be detailed. The first approach may be considered a more traditional approach to region proposals for traffic signs; however, the latter methods may be considered novel approaches within the field of robotic vision. The majority of this section will dedicated to the third approach which will be compared in detail with state-of-the-art segmentation algorithms and represents a key contribution of the paper.

4.1 Binary Histograms

The first method for region proposals is to employ colour segmentation-based candidate generation using binary histograms. The process itself is relatively simple and straightforward so it will not be covered in detail for conciseness.

A true-colour image is captured by the PiCamera, and then is transformed form the RGB to HSI colourspace. A lookup table of empirically defined values is then used to binarize the image and generate a mask. A histogram is calculated for each axis where the foreground pixels vote into each corresponding bin.

(a)

(b)

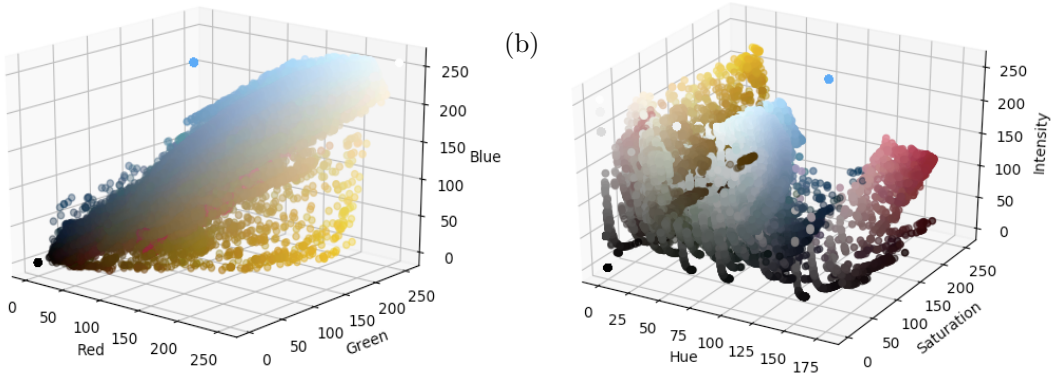

Figure 15: (a): the true-colour input image captured on the PiCamera (b): on the left each pixel assigned within the RGB colour-space, and on the right is each pixel assigned within the HSI colour-space. The hue is in the range [0,180) due to the OpenCV implementation's use of the uint8 datatype

It is difficult to choose thresholds to generate the binary image within the RGB colour-space as can be seen in the Figure 15 (b). In order to account for the effects of changes in illumination, reflections and other spectral phenomena, the image is translated to the HSI colour-space.

This approach is able to function in EyeSim and generate all the signs of interest; however, as can be seen in Figure 15 (a), the presence of white light especially in the walls of the Mobile Robots Laboratory make it difficult to function in the real-world.

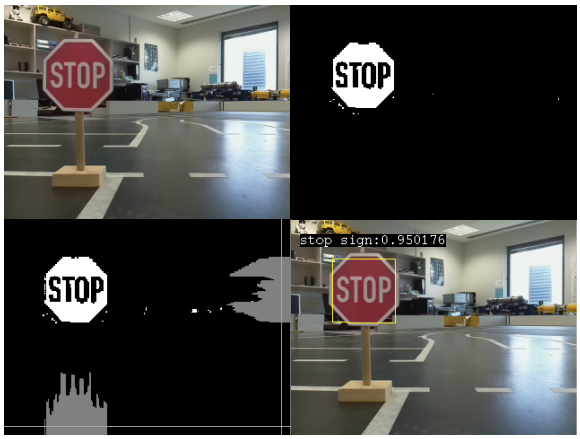

Figure 16: An example of the traffic sign detection pipeline ignoring white light. (Top Left) a true-colour input image of a stop sign on the Carolo Cup track recreated at the UWA Mobile Robots Lab. (Top Right) HSI thresholding has been applied to the input image to generate a binary image. (Bottom Left) histograms are generated for both the x-axis and the y-axis as shown in dark grey, a threshold for each axes in shown in light grey. (Bottom Right) the input image with a bounding box (yellow), classification and confidence displayed

The binary histogram approach is the simplest of the approaches employed in the project for region proposal. It is also the fastest as it is able to achieve a mean FPS of 18.80Hz with object recognition. In its present form it comes with a caveat: only a critical subset of the traffic signs may be detected which includes stop signs, parking signs, pedestrian crossing signs and give way signs. The non-coloured speed limit and unlimit signs are invisible to this method of detection.

4.2 Keypoint-Dense Regions

The method proposed in this section will again only comprise a brief section, but is an interesting contribution that may be useful for similar tasks.

The principle is to use a sliding window across an input that has keypoints generated and mark all window positions greater than a certain thresholds. As discussed previously, keypoints can act as a metric for low-level visual salience and keypoint-dense areas may correspond to areas with objects. It should be understood that clustering keypoints based on square regions is not very efficient; however, it is an approach that could be used to detect any large scale object in a sparse environment.

It does not rely on any training and could be deployed for a range of tasks where accurate localisation is not required.

Algorithm 1 *Keypoint-Dense Region Proposal*

INPUT: Input Image with width w and height h
OUTPUT: List of rectangles, *ROIs*
DATA: Vector of SIFT-like keypoints, *keypoints*
$j \leftarrow 0$
while $j < h$ **do:**
    $i \leftarrow 0$
    while $i < w$ **do:**
        $count \leftarrow 0$
        for all *kp* such that $kp \in keypoints$ **do:**
            if *kp* in RECT(*i, j, i+xs, j+ys)* **do:**
                $count \leftarrow count+1$
            end if
            if $count \geq THRESHOLD$ **do:**
                append RECT(*i, j, i+xs, j+ys) to ROIs*
            end if
        end for
        $i \leftarrow i$ + STEP
    end while
    $j \leftarrow j + STEP$
end while
return *ROIs*

The sliding window generates a great deal of ROIs in most scenarios; therefore, non-maxima suppression is then applied as can be seen in Figure 17 below. The Carolo Cup map itself is sparse enough that this approach can be used; however, it does fail when using real-life camera feeds. The most similar method that could be found was using keypoint density to optimise R-CNN

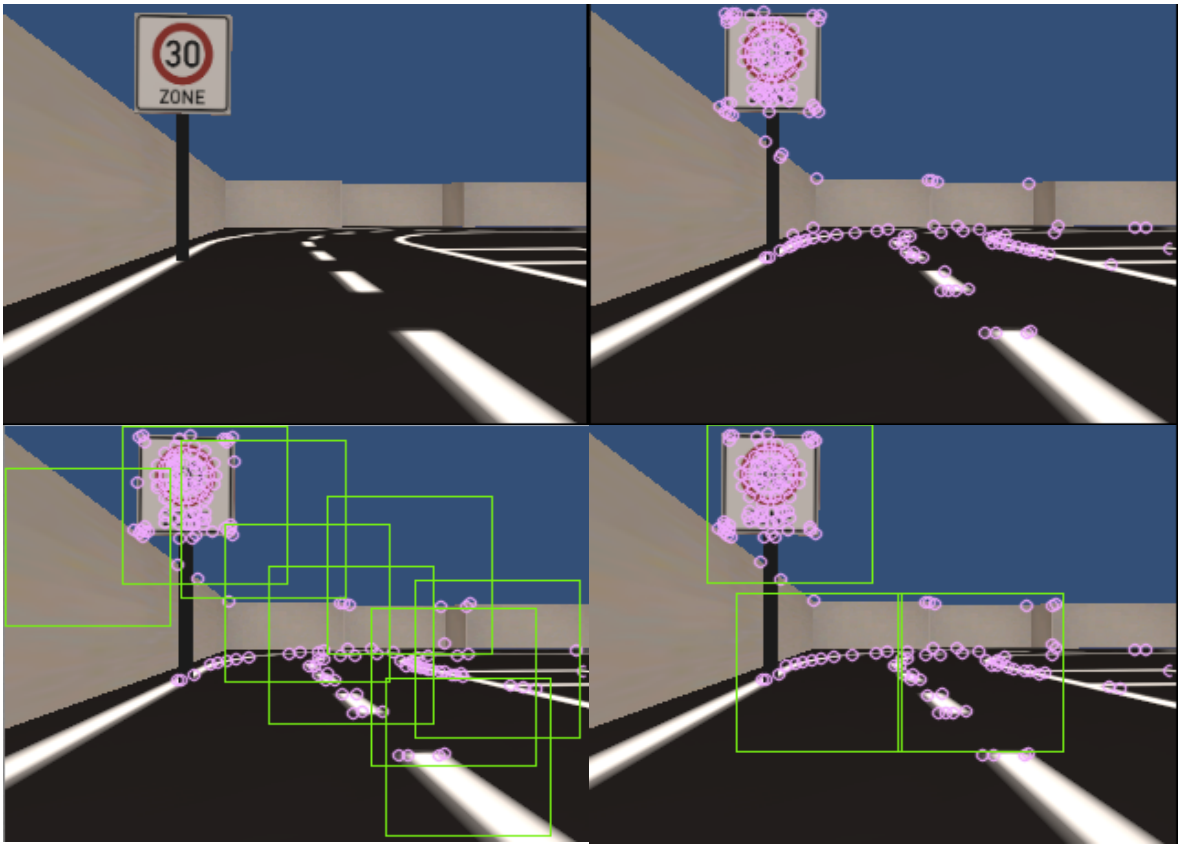

Figure 17: An example of Keypoint-Dense Region Proposal using EyeSim

4.3 Binary Pseudo-Pixelwise Image Segmentation

In this section, a novel low-computational cost binary image segmentation network architecture inspired by SegNet and MobileNet is proposed and evaluated in comparison to existing semantic pixel-wise labelling algorithms. Figure 18 demonstrates a sample of the results using this approach with a modified KITTI semantic segmentation dataset [27].

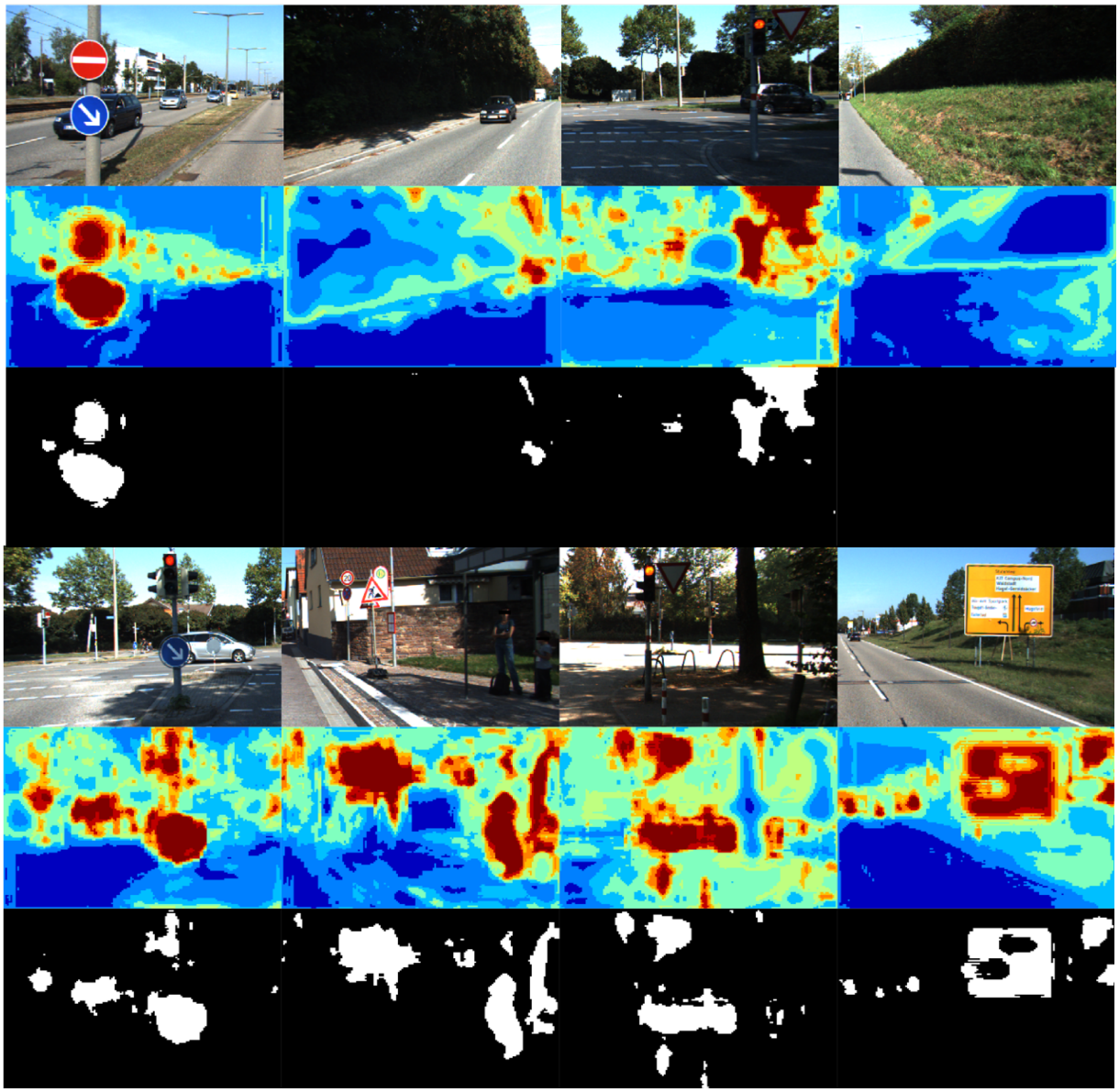

Figure 18: A sample of results of the pixel-wise traffic sign detection approach. The confidence threshold employed for binarization here is 0.9

4.3.1 Design Philosophy

The goal is to implement a real-time convolutional neural network architecture to label each pixel of an image as foreground or background where foreground corresponds to regions labelled as a traffic sign..

In order to operate in real-time on a Raspberry Pi 3B, the network must be minimal and very efficient.. The network is too shallow to achieve results not dissimilar to a colour filter with limited spacial information; however, the rest of the pipeline architecture should be able to account for this matter.

4.3.2 Proposed Architecture

The architecture is heavily inspired by the encoder and decoder convolutional layers of SegNet; however, it aims to implement MobileNet's depth-wise separable convolutional layers instead of the two-dimensional convolutional layers of SegNet. There are a number of tuneable hyperparameters for latency, size and accuracy.

In order to ensure that the network will be able to achieve the task satisfactorily in terms of accuracy and speed, there are five different hyperparameters proposed:

i. The width multiplier, derived from MobileNet, is directly related to depth of the hidden layers. It is denoted 'α'.

ii. The resolution multiplier, also related to MobileNet, directly resizes the input image size. It is denoted 'β'.

iii. The resizing coefficient is used to control the resizing in the encoder and decoder layers. It is denoted 'γ'.

iv. The depth gradient is used to control the rate at which the depth of the hidden layers increases in the encoder layers and decreases in the decoder layers. It is denoted 'ρ'.

v. The initial depth is used to specify the initial depth of hidden layers. Larger values will likely provide more accurate results at the cost of performance. It is denoted 'δ'.

4.3.3 Training

The fundamental issue that affects training the convnet is the lack of datasets available for semantic pixelwise segmentation. The amount of work required to make a semantic pixel-wise labelled data is tremendous; therefore, it was deemed infeasible to construct a dataset during the timespan given. There were three datasets found that offered potential for training: KITTI [27], Cityscapes Dataset [28], and WildDash [29].

The KITTI dataset was in the end able to be used albeit with a number of modifications. The KITTI dataset only contains 200 images for training. It also has an input resolution of 1242×375, which is very impractical for the proposed network which accepts images of resolution 320×240. The solution was to divide the images into two horizontally-wise, and then resize each to 320×240.

Figure 19: An example post-processed image from the KITTI dataset (L) with corresponding binary mask (R) [27]

The resultant dataset was heavily imbalanced; that is, there was an unequal representation of instances corresponding to foreground and background.

There are two options to account for the imbalanced relative class frequencies: i) apply a sample weight to each input image, or ii) apply a class weight for the true label of each image. If this were not to occur, in this case it would label every single pixel black and not converge.

In order to facilitate training the following split was employed: a randomly selected 10% of the total images were designated for training purposes, 80% of the remainders were designated as the training set and the rest comprised the validation set. In order to expand the training dataset, the training set was augmented via Keras [30] using the parameters given in Table 2.

Table 2: Image Data Generator Parameters (Region Proposal)

Parameter	**Value**
Brightness	[-0.10, 0.10]
Rotation	[-0.05, 0.05]
Shear	[-0.05, 0.05]
Width Shift	[-0.10, 0.10]
Height Shift	[-0.10, 0.10]
Zoom	[0, 0.10]

Prior to training each combination of hyperparameters, a learning rate range test was conducted after 30 epochs to evaluate which learning rate should be selected for long-term training. The test is similar to that devised by Leslie N. Smith [31].

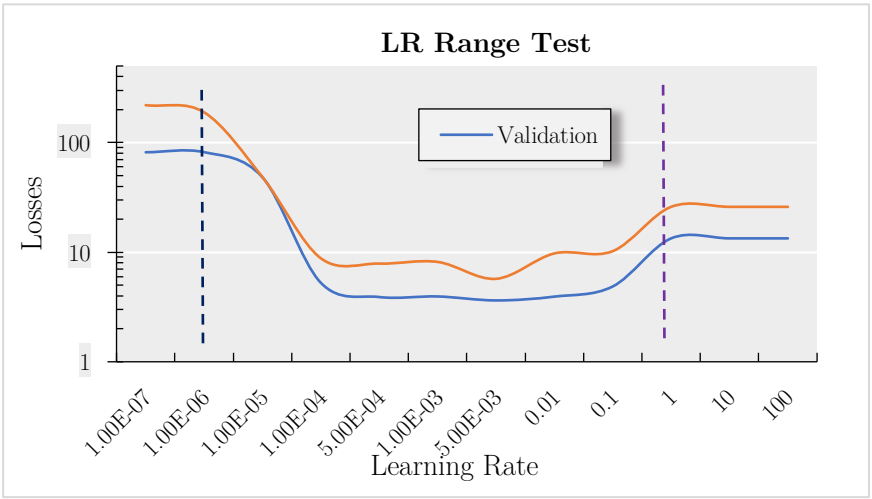

Figure 20: Learning rate hyperparameter test (Region Proposal)

In Figure 20 there is a dark blue line on the left hand side of the graph, this corresponds to the region where the learning rate hyperparameter is too low the loss is trapped in a local minima. The purple line on the right hand side of the graph corresponds to a region where the learning rate is too high so it does not converge.

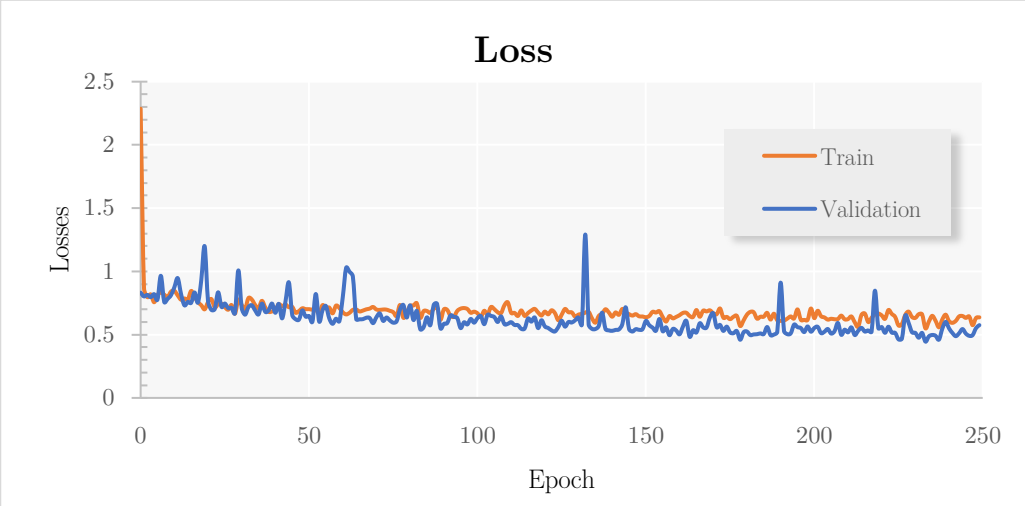

Figure 21: Training and Validation Loss vs. Epoch

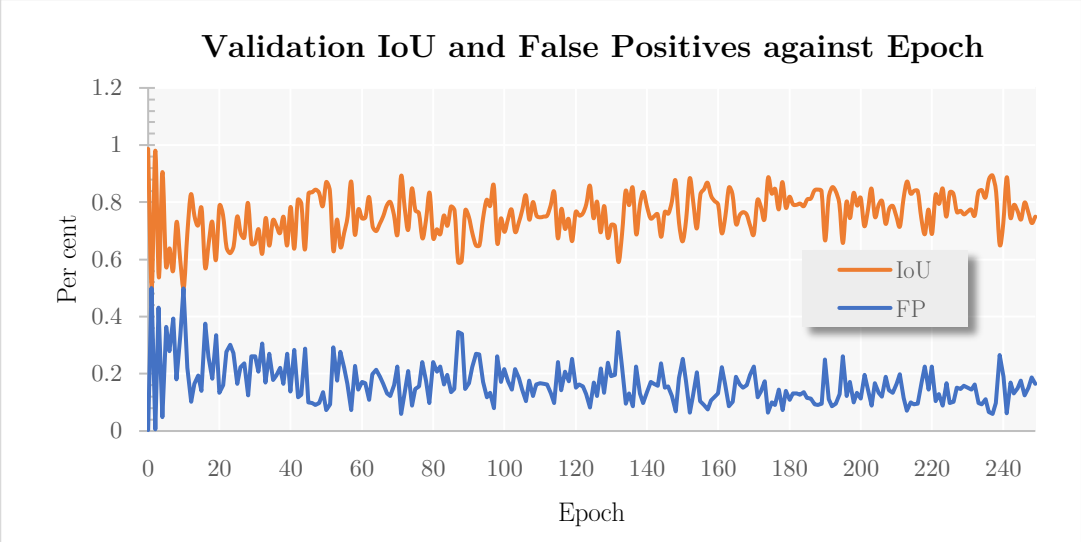

Figure 22: Validation IoU and FPs over the range of epochs

4.3.4 Evaluation

In this section, the results using the binary pseudo-pixelwise segmentation method are presented. It should be noted that only a subset of all possible network architectures will be discussed because there are many different combinations of hyperparameters.

The output from the network itself after inference is two-dimensional matrix containing floating point numbers from zero to one. In similar networks, these values are referred to as the confidence of the network. It should be noted that these values are calculated via a loss function, and are not a probability in any real capacity. In Figure 23, an example output is presented with a mapping.

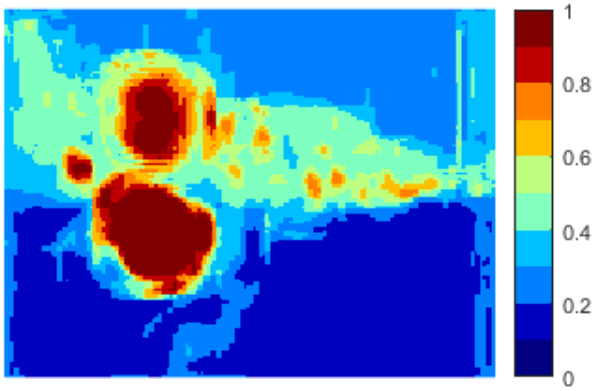

Figure 23: An example output of the network

The metric employed to compare methods in the KITTI dataset is intersection-over-union, otherwise referred to as 'IoU' or the Jaccard Index. It is defined as the area of intersection divided by the area of union as exemplified in Figure 24.

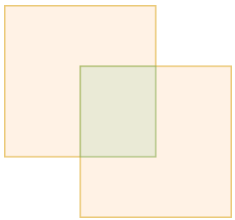

Figure 24: An example of IoU. The green region represents the intersection, and the union is comprised of the orange areas and the green area

A pixel is designated as a True Positive (TP) if it is classified as foreground when it is a foreground pixel; inversely, a pixel is designated as True Negative (TN) if it is

classified as a background pixel and it is a background pixel. If a pixel is designated foreground when it is background, then it is a False Positive (FP). Likewise, if it is classified as background and it is foreground then it is a False Negative (FN).

The PASCAL VOC definition for IoU is employed by KITTI to compare methods for semantic segmentation as given below:

$$IoU = \frac{N(TP)}{N(TP) + N(FP) + N(FN)}$$

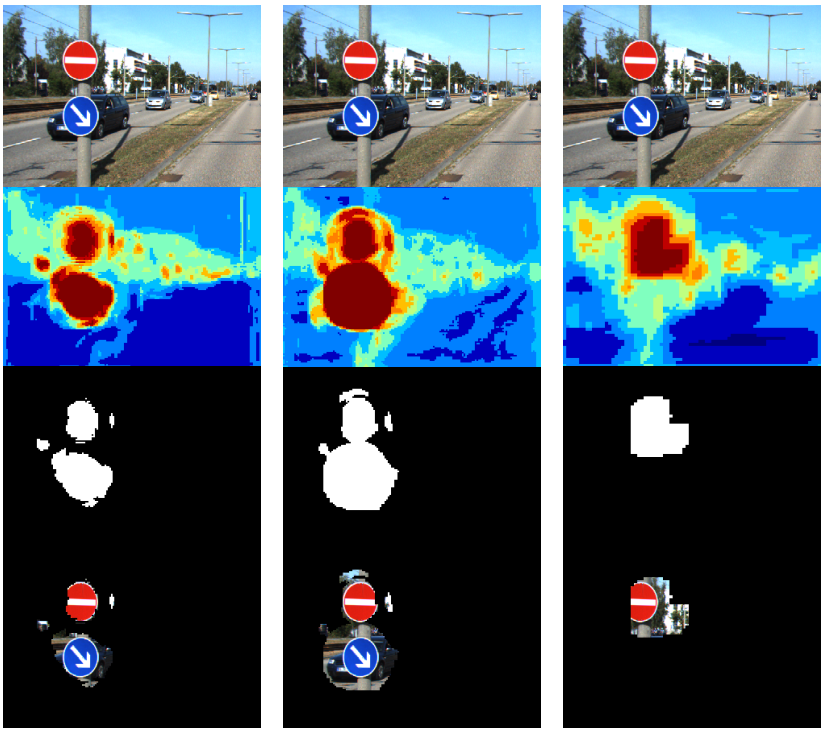

Figure 25: Comparison between outputs of three CNNs when using different hyperparameters expressed in the form $(\alpha, \beta, \gamma, \rho, \delta)^T$. From left to right: $(1.0, 1, 2, 2, 64)^T$, $(0.25, 1, 2, 2, 64)^T$, $(0.25, 2, 4, 2, 8)^T$. From top to bottom: true-colour image input, confidence map output from CNN, binary mask with confidence threshold of 0.9, and bitwise AND operation between mask and input image

The pixel-wise labels for the test set of images are not publicly available so comparisons with existing methods are only able to completed using the validation set of images.

*Table 3: Evaluating and Comparing Semantic Segmentation Methods on the KITTI dataset. *Method only detects traffic signs and no other classes, and IoU was calculated on validation set*

Method	Class IoU
Proposed method where $(\alpha, \beta, \gamma, \rho, \delta)^{\mathrm{T}} = (1.0, 1, 2, 2, 64)^{\mathrm{T}}$	0.7273*
Improving Semantic Segmentation via Video Propagation and Label Relaxation [32]	0.7282
Unsupervised Domain Adaption to Improve Image Segmentation Quality Both in the Source and Target [33]	0.5950
SegStereo: Exploiting Semantic Information for Disparity Estimation [34]	0.5910
SDNet: Semantic Guided Depth Estimation Network [35]	0.5114
Pixel-wise Attentional Gating for Parsimonious Pixel Labelling [36]	0.4796

In terms of real-time performance on the Raspberry Pi, irrespective of the hyperparameters used the fastest it is able to run is at a rate of 1.62Hz. This translates to it falling short of real-time inference speed goal; however, it is likely that it would run in real-time on a Raspberry Pi with either a Movidius Neural Compute Stick, a Google Coral Accelerator or a Jetson Nano.

When compared to similar methods using the KITTI dataset, it must be understood that it is not an equal comparison and in actuality the proposed network would perform worse than the results indicate in Table 3; however, the proposed method has shown that it is able to obtain respectable results for simple cases such as those provided in Figure 25, and shown that it may be feasible with training hyperparameter tweaking. The greatest improvement will come from training with a larger set of training images.

5 Classification

In this chapter of the paper, an overview is given with respect to the current state-of-the-art methods for real-time traffic sign recognition, the implementation of the CNN-based classifier will be detailed, metrics are provided and the implemented method for region classification will be evaluated with respect to existing solutions.

5.1 Overview

Traffic sign recognition remains a challenging area of research with many contributors due in part to the multitude of challenges introduced through lighting and blurring effects, occlusion and partial-occlusion, and sign deterioration. To complicate the task further, applications will generally require any solutions to operate in real-time.

The convolutional neural network architecture MobileNet was employed or its well-documented large-scale image classification ability as exemplified through Imagenet accuracy, and because it is able to run in real-time on embedded devices [17].

5.2 The German Traffic Sign Recognition Benchmark

The German Traffic Sign Recognition Benchmark (hereafter referred to as the 'GTSRB') was a multi-class traffic sign recognition classification competition held at the International Joint Conference on Neural Networks (IJCNN) 2011 [37]. Stallkamp et al., built a dataset consisting of more than 50, 000 traffic sign training images in 43 different classes, with two test sets containing more than 10, 000 images each [37]. The GTSRB dataset represents the most extensive traffic sign recognition dataset publicly available.

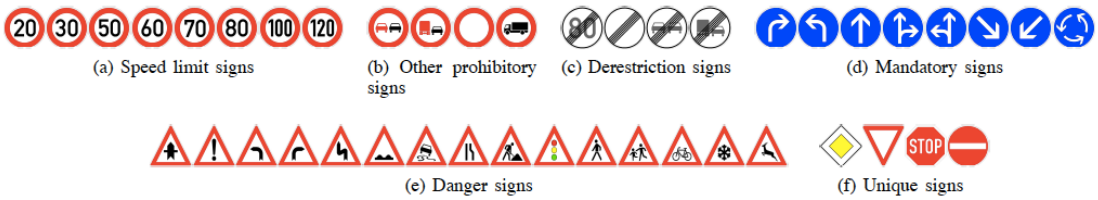

Figure 26: The 43 classes of traffic signs in the GTSRB dataset [37].

The GTSRB dataset is challenging and life-like: the effect of blur, lighting effects, occlusion are all present as may be seen in Figure 27. The goal with training on this dataset is to demonstrate the feasibility of using the chosen classification CNN architecture for image classification when given a wide range of traffic signs under diverse array of conditions.

Figure 27: A sample of the GTSRB dataset [37].

5.3 Training

As discussed earlier within the paper, MobileNet has seen significant success for the purpose of large-scale image recognition especially when considering its latency and size. It is due to these properties that it is likely that MobileNet will be used for a range of object detection going into the future for robotics applications; therefore, in this section there will be a significant amount of data relating to the training of MobileNets of a wide range of input sizes and width multipliers for use with the Raspberry Pi that should be insightful for readers whom are working in fields that relate to robotic vision.

5.3.1 Training Set

The dataset was generated by annotating video sequences recorded on German roads in March, October and November in 2010 [37]. The size of the images vary from 15×15 to 222×193 and are stored in RGB colour format.

There are two sets of data provided: a training set, and a test set. The test set is not to be used for learning, but to evaluate the method. Techniques involving deep learning require developers to further subdivide the training dataset into an aptly named 'training set', and a 'validation set'. A randomly selected 80% went to be used directly as the training set, and the remaining 20% was used for validation.

The dataset is unbalanced as demonstrated by [37] below in Figure 28 – that is, there was an unequal representation of each traffic sign class.

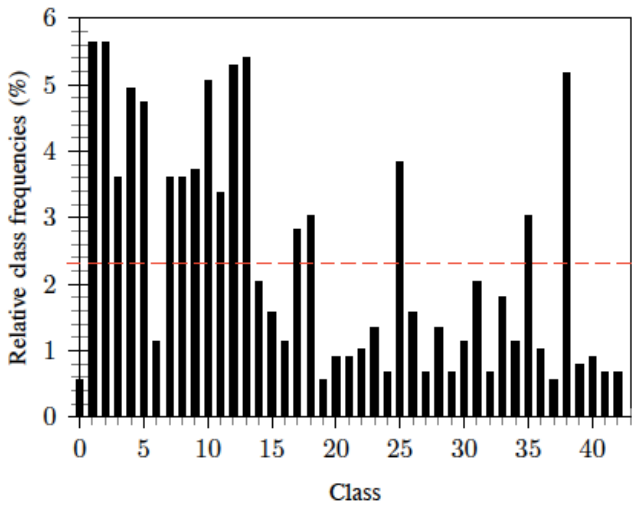

Figure 28: The relative class frequency among the dataset [37]

There are two options to account for the imbalance relative class frequencies: i) apply a sample weight to each input image, or ii) apply a class weight for the true label of each image. In either case, the weight will then be employed by the loss function to compensate for the dataset imbalance. If this were not to occur, the neural network would likely not try to adapt for any lower relative frequency classes and thereby lose

overall accuracy. In the context of this dataset, a class weight was applied as there was no benefit to assigning a sample weight for each of the 10, 000 images.

5.3.2 Dataset Augmentation

The GTSRB dataset is extensive and contains a diverse array of life-like images; however, the complexity of a CNN-based classification architecture like MobileNet leans towards a propensity to overtrain. In order to mitigate overtraining and to encourage the 'generalisation' capabilities of the convolutional neural network, data augmentation was undertaken via Keras [30].

Table 4: Image Data Generator Parameters (Classification)

Parameter	**Value**
Brightness	[-0.15, 0.15]
Rotation	[-0.15, 0.15]
Shear	[-0.10, 0.10]
Width Shift	[-0.05, 0.05]
Height Shift	[-0.05, 0.05]
Zoom	[0, 0.15]

The data augmenter accepts a batch of images before they are fed to the CNN for training and undertakes the parameterised set of randomised transformations specified above. The batch of training images is then sent to the CNN for training after each transformation has been completed.

5.3.3 Training Results

In this section, the details following the end-to-end training of MobileNet for the task of traffic sign recognition are discussed. There were thirty MobileNets trained in total with varying hyperparameters where the width multiplier, $\alpha \in \{0.125, 0.25, 0.5, 0.75,$

and 1.0}, and where the input size, $\beta \in$ {32×32, 48×48, 64×64, 80×80, 96×96, 128×128}. The choice to evaluate such a wide array is for future use: MobileNets have proven powerful for a wide range of tasks, and benchmarking them will prove useful for future developers. The results will primarily focus on a full-sized 128×128 network, but the accuracies for each will be presented.

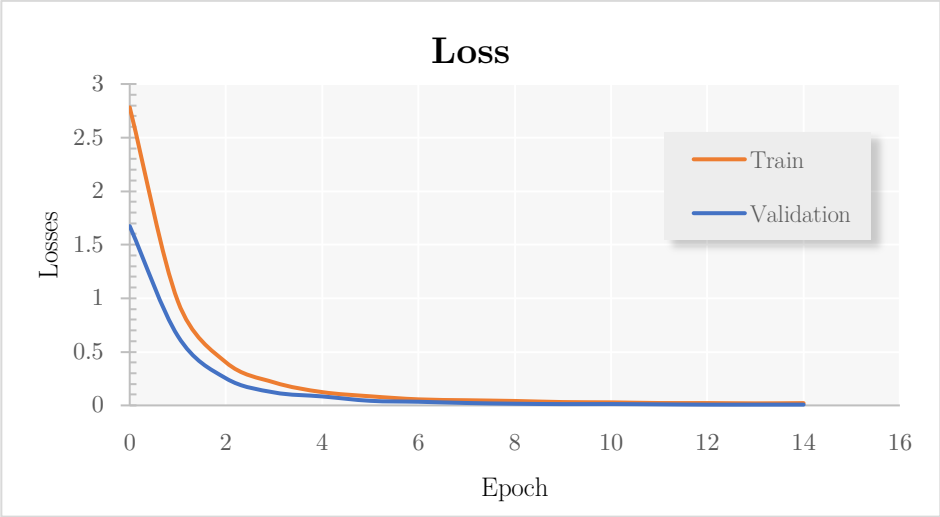

Figure 29: Training and validation loss where α=1.0, and β= 128×128

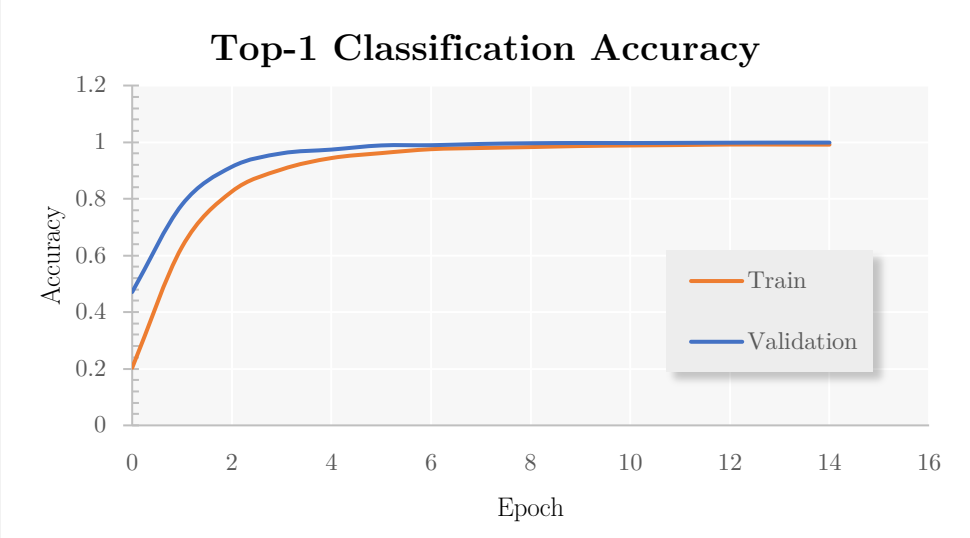

Figure 30: Top-1 Predictive Accuracy for the training and validation datasets where $\alpha = 1.0$, and $\beta = 128 \times 128$

As can be seen in the Figure 29 above, the loss begins to converge after training begins and only fifteen epochs are required to achieve a near 100% accuracy. It should be noted that the training accuracy is lower than the validation accuracy which alludes to the neural network being trained on sets more difficult than the training sets. It takes longer to converge because it is learning ways to compensate for changes in data – for example, it may need to compensate for reduced luminosity.

In order to mitigate the effect of randomness, a seed was set for the random number generators in the TensorFlow and NumPy backends. It should be possible to entirely replicate these results.

Table 5: Comparing Top-1 Accuracy, Top-5 Accuracy, and F_1 Score against MobileNet Input Resolution and Depth Multiplier

Hyperparameters		Top-1 Accuracy	Top-5 Accuracy	F_1 Score
Input Resolution	Depth Multiplier, α			
128×128	1.0	88.73%	98.00%	0.89
128×128	0.75	88.86%	97.97%	0.89
128×128	0.5	87.40%	97.38%	0.87
128×128	0.25	88.46%	98.26%	0.89
128×128	0.125	86.89%	97.96%	0.87
96×96	1.0	84.53%	96.88%	0.85
96×96	0.75	86.35%	97.40%	0.86
96×96	0.5	81.77%	97.09%	0.82
96×96	0.25	78.00%	96.54%	0.78
96×96	0.125	72.01%	95.57%	0.73

Hyperparameters		Top-1 Accuracy	Top-5 Accuracy	F_1 Score
Input Resolution	Depth Multiplier, α			
80×80	1.0	89.10%	98.10%	0.89
80×80	0.75	85.36%	97.76%	0.85
80×80	0.5	82.19%	96.94%	0.82
80×80	0.25	77.05%	95.72%	0.77
80×80	0.125	62.94%	91.67%	0.63
64×64	1.0	82.32%	96.58%	0.82
64×64	0.75	81.76%	96.90%	0.82
64×64	0.5	80.82%	96.62%	0.81
64×64	0.25	64.70%	91.96%	0.65
64×64	0.125	48.84%	83.75%	0.49
48×48	1.0	78.99%	96.52%	0.79
48×48	0.75	74.22%	95.47%	0.74
48×48	0.5	63.47%	90.51%	0.64
48×48	0.25	56.93%	89.35%	0.58
48×48	0.125	39.64%	79.77%	0.41
32×32	1.0	69.63%	95.03%	0.70
32×32	0.75	65.46%	92.64%	0.66

Hyperparameters		Top-1 Accuracy	Top-5 Accuracy	F_1 Score
Input Resolution	Depth Multiplier, α			
32×32	0.5	61.35%	89.99%	0.61
32×32	0.25	67.51%	93.71%	0.68
32×32	0.125	34.65%	76.74%	0.35

The table above summarises a great wealth of information and provides a detailed summary of the recognition capability of each model. It should prove invaluable when selecting MobileNet training hyperparameters for a slew of applications. The general pattern is that decreasing the input resolution or width multiplier leads to a drop-off in accuracy; however, it does not always hold true. Interestingly, the most accurate model according to the test set has an input resolution of 80×80 pixels. These discrepancies are likely due to the larger neural networks overtraining or becoming trapped in local minima. They could however also be the result of image resizing before being classified.

5.4 Evaluation with State-of-the-Art Recognition Methods

In this section, the results obtained are compared the other approaches proposed for the GTSRB. The first comparisons that will be made are with state-of-the-art solutions that have been developed since the competition started.

Table 6: Benchmarking MobileNet for Traffic Sign Recognition VS State-of-the-Art Methods

Method	Speed Limits	Other Prohibitions	Derestriction	Mandatory	Danger	Unique	Total
128×128 MobileNet where $\alpha = 1.0$	92.45%	92.45%	96.63%	84.47%	81.14%	86.17%	88.73%
CNN with 3 Spatial Transformers [38]	**99.47%**	99.87%	98.89%	99.77%	**99.07%**	99.22%	**99.71%**
Committee of CNNs [39]	99.09%	**99.93%**	**99.72%**	**99.89%**	97.96%	99.51%	99.46%
Color-blob-based COSFIRE filters for object recognition [40]	97.63%	**99.93%**	94.17%	99.83%	98.67%	**100.00%**	98.97%
Human Performance [37]	98.61%	**99.93%**	98.89%	99.72%	98.03%	98.63%	98.84%
Multi-Scale CNNs [41]	95.95%	99.87%	94.44%	97.18%	92.08%	98.73%	98.31%
Random Forests [42]	95.37%	99.13%	87.50%	99.27%	93.73%	98.63%	96.14%
LDA on HOG [43]	91.44%	96.80%	85.83%	97.18%	90.61%	98.43%	95.68%

The confusion matrix visualises cases of mistaken identity; that is, when the classifier predicts the incorrect class. The diagonal line indicates that the majority of the classifications were correct. The error will naturally be more dense in regions of the corresponding to traffic signs having a similar appearance.

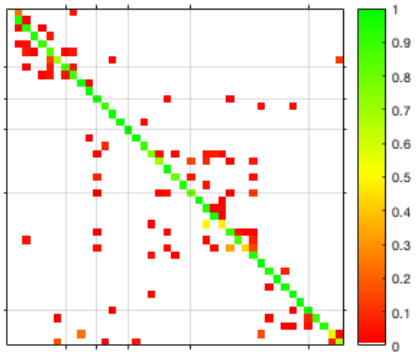

Figure 31: Confusion Matrix for 128×128 MobileNet model where $\alpha = 1.0$

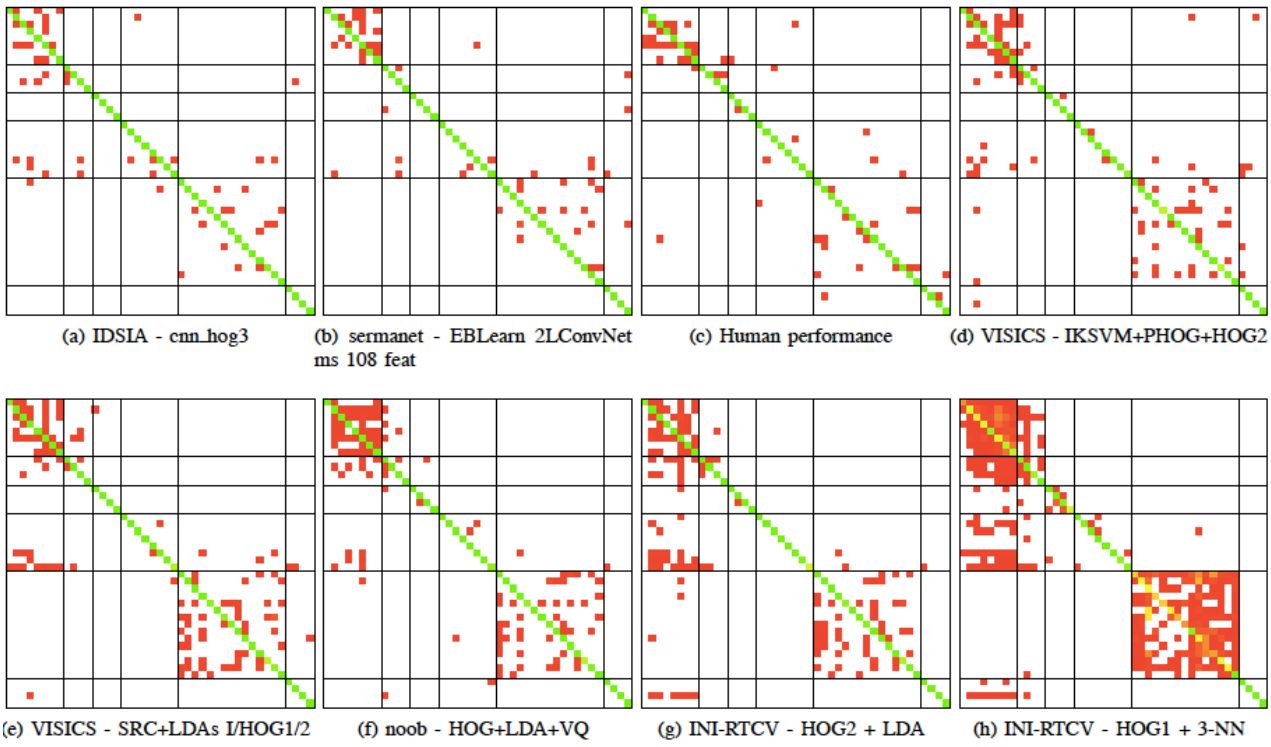

Figure 32: Confusion matrices for GTSRB competitors in the 2011 competition [37]

The results of state-of-the-art solutions on the same test set may be seen in Figure 32. The top-left hand corner of the confusion matrices correspond to the speed limit signs.

It should be expected that these are the most difficult to classifier for each algorithm. The effect of blur in particular can make distinguishing the numbers particularly difficult even for a human as can be seen in (c). These comparisons are not exactly fair as MobileNet has been designed to run in real-time even on embedded systems such as the Raspberry Pi 3B. The results do indicate however, that MobileNet can provide a valuable tool for developing traffic sign recognition systems on inexpensive hardware especially when there is a limited subset of signs such as those within the Carolo Cup.

5.5 Runtime Optimisation

In this section, results relating to the runtime optimisation of MobileNet for a Raspberry Pi 3B will be presented. This data should be particularly useful for readers tuning their own implementations for latency, size, and input resolution.

In order to conduct the following tests, each model inferred the class of 1000 images of randomised RGB noise. The time taken for each inference was recorded in order to calculate the mean, maximum and minimum inference times for each model. This information should be useful readers wishing to develop embedded system MobileNet-based object detection systems.

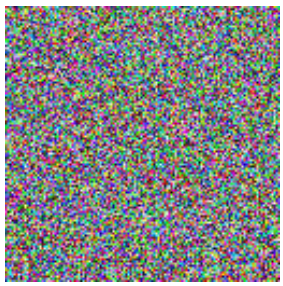

Figure 33: A randomised 128×128 RGB noise image

Table 7: Benchmarking the relationship between input resolution, width multiplier, model size, and mean inference time on a Raspberry Pi 3B for TensorFlow Lite

Architecture	Input Resolution	Width Multiplier, α	Model Size (MB, 3 s.f.)	Mean Inference Time, $\overline{T_i}$ (ms)
MobileNet [17]	32×32	0.125	0.0967	0.11
	32×32	0.25	0.254	0.21
	32×32	0.5	0.875	0.51
	32×32	0.75	1.89	0.99
	32×32	1.0	3.29	1.53
	48×48	0.125	0.0967	0.23
	48×48	0.25	0.254	0.44
	48×48	0.5	0.875	0.97
	48×48	0.75	1.89	1.92
	48×48	1.0	3.29	3.15
	64×64	0.125	0.0967	0.42
	64×64	0.25	0.254	0.73
	64×64	0.5	0.875	1.80
	64×64	0.75	1.89	3.61
	64×64	1.0	3.29	5.87
	80×80	0.125	0.0967	0.51
	80×80	0.25	0.254	1.05
	80×80	0.5	0.875	2.75
	80×80	0.75	1.89	5.47
	80×80	1.0	3.29	8.46
	96×96	0.125	0.0967	0.74
	96×96	0.25	0.254	1.54
	96×96	0.5	0.875	4.15
	96×96	0.75	1.89	8.31
	96×96	1.0	3.29	13.34
	128×128	0.125	0.0967	1.16
	128×128	0.25	0.254	2.76
	128×128	0.5	0.875	7.16
	128×128	0.75	1.89	14.56
	128×128	1.0	3.29	24.03

It should be noted that the model size is dependent singularly on the width multiplier. This aligns with expectations as it corresponds to each of the kernel weights of the CNN architecture. The inference time is dependent on both the width multiplier and input resolution because they control the number of multiply accumulate floating point operations that will need to be completed by the Raspberry Pi CPU.

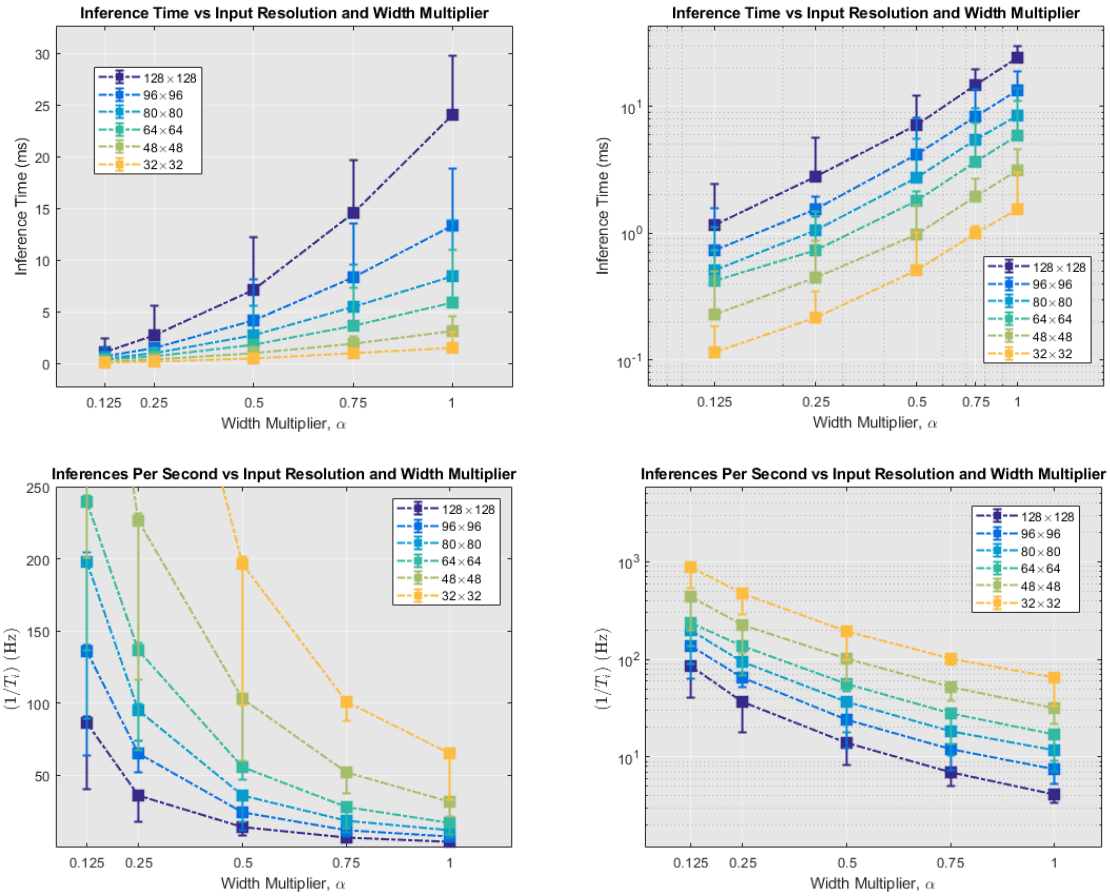

Figure 34: Benchmarks for MobileNet inference time on the Raspberry Pi 3B against width multiplier where $\omega \in \{0.125, 0.25, 0.50, 0.75, 1.00\}$. The inverse of the inference time is useful to calculate the maximum FPS attainable. For example, the upper limit for $\omega = 0.75$ indicates the maximum FPS for a 128x128 MobileNet three-quarters the size of a typical 128x128 MobileNet is approximately 10Hz

The benchmarks above may be used to ascertain which models can be used when attempting to achieve a given inference time. Together with the accuracy results given in the training results section, a user can ascertain the most appropriate input resolution and width multiplier combination to meet accuracy, latency and size requirements.

The intended use of the traffic sign detection and recognition system will be on the SoccerBots to complete the obstacles presented in the Carolo Cup; therefore, the power to disseminate between similar signs is not paramount. In order to minimise the computational cost of the traffic sign recognition, a classifier neural network of size 80×80 and width multiplier of 0.25 was employed. This network has been shown to be infer a Top-5 accuracy of greater than 95%, and hence it is likely to be sufficient for the Carolo Cup. The specified combination of hyperparameters should also have the capability to classify approximately 100 image frames per second on the Raspberry Pi 3B.

6 Tracking and Reliability

The following chapter details the tracking and reliability modules of the traffic sign detection. It explores the two methods employed for tracking, and the novel approach to ascertain the reliability of a detection.

6.1 Tracking

In the lifetime of the project, there are two methods that have been employed to track ROIs between adjacent frames:

i. firstly, the Euclidean distance between the centres of ROIs if given below a specified threshold;

ii. secondly, the proportion of SIFT-like keypoints matched clusters exceeds a specified threshold.

The former requires less computational power; however, it may be exploited and there is no guarantee that it will actually be tracking the object. It is susceptible to error, and may be inappropriate for applications where a high degree of reliability is mandatory.

The latter approach requires forethought to implement, but it has the capability to operate in real-time on the Raspberry Pi 3B+ and ensures a degree of similarity when tracking objects.

6.2 Reliability

This section will expand on the framework discussed in Chapter 3. It will detail how each detection is used to build up a detection profile for an object and the calculations involved. The operating principle is that it should emulate a committee of classifiers, but do so over time.

6.3 The Class Reliability Metric

The reliability coefficient is composed of two elements: the first accounts for time redundancy, and the second accounts for parallel reliability. The reliability coefficient metric is defined as

$$Z_i = \left(1 - \frac{1}{\alpha^{\beta n_i}}\right) \prod_{j=1}^{n_i} (1 - c_{i,j}) \tag{6.1}$$

where Z_i is the reliability coefficient of object ; $c_{i,j}$ is the confidence of the jth prediction of object i ; n_i is the number of detections of object i; and, α and β are empirically tuned constants.

The first element accounts for the number of detections of a particular object class. If a particular object class has been detected multiple time the reliability should increase, but if it has only been detected a very limited number of times it should be capped at a maximum limit. The maximum limit is effectively set by the first element.

The second element factors in the confidence output by the neural network classifier. The confidence will lie between 0 and 1. The detections can be considered as parallel reliability elements and the reliability should increase the more a particular object class is detected.

A reliability coefficient is calculated for each object class detected and they form a vector referred to as the class reliability coefficients.

6.4 The Weighted Class Reliability Metric

To account for situations where more than one class has been ascribed to an object, a weighted reliability coefficient is employed.

The Softmax function, otherwise known as the Boltzmann or Gibbs distribution, is frequently employed in neural networks as the activation function. In this case, it is

being used to weight each reliability coefficient where the reliability coefficient is non-zero. The Softmax function is given as

$$\sigma(\boldsymbol{Z})_i = \frac{e^{Z_i}}{\sum_{j=1}^{M} e^{Z_i}} \tag{6.2}$$

where Z_i is the reliability coefficient of object i; $\boldsymbol{Z}$ is the vector of class reliability coefficients; and M is the total number of different object classes that have been detected for a given detection profile.

Subsequently, each reliability coefficient is weighted according to the corresponding output of the Softmax function as

$$R_i = Z_i \cdot \sigma(\overline{\boldsymbol{Z}})_i \tag{6.3}$$

where R_i is the weighted reliability coefficient of object i; Z_i is the reliability coefficient of object; and $\sigma(\overline{\boldsymbol{Z}})_i$ is the Softmax function applied to the vector of non-zero class reliability coefficients.

The object class with the maximum weighted reliability coefficient is the object class that the pipeline predicts the detected object belongs. The corresponding weighted reliability coefficient may be used to measure provide a likelihood that the detected object belongs to a particular class.

There are no similar metrics that the author is aware of that can be used for comparison purposes; however, it can be related to a committee of CNNs method.

7 Conclusion

This work has been written to implement and evaluate traffic sign recognition and detection on the ground robots at the UWA Robotics and Automation Lab. The use-case of the software is for it to be employed for the Carolo Cup autonomous driving student competition. In undertaking the project, a significant amount of work was completed benchmarking and optimising for the Raspberry Pi 3B and similar embedded devices. The approach was multi-faceted and the problem divided methods used for region proposal and classification.

Three methods were employed for region proposal, the latter two being novel albeit with mixed success. The first approach was successful in the regard that it met the inference time benchmark of 8Hz; however, only a subset of the signs were predicted. The second approach was able to operate in the benchmark specified and had the potential to detect all signs, but it was unscalable for the real-world. The third method involved end-to-end training of a new pixel-wise image segmentation CNN that would theoretically be scalable for the real-world, but it was unable to run in real-time for the Raspberry Pi, and furthermore the detections lagged behind state-of-the-art methods as specified in Table 3. This problem however could be rectified with alternative hardware such as the Jetson Nano, and by obtaining more training data. In terms of future developments, two approaches that seem promising are using white and coloured maximally stable extremal regions or further developing the CNN architecture.

In terms of the methods used for classification, the applicability of MobileNet for traffic sign recognition has been thoroughly investigated, and benchmarks are now available that can be used to compare it to state-of-the-art methods that are designed entirely for traffic sign recognition. The data collected should also prove useful for readers interested in computer vision for embedded platforms.

In summary, this thesis has explored a number of techniques that represent novel contributions for the task of traffic sign recognition and detection. The SIFT-like

keypoint cluster-based pipeline itself has the capability to act a committee of CNNs in itself which could be useful for a range of real-time object detection applications. The benchmarks completed also represent a basis from which future students can build upon and evolve future work.

Bibliography

[1] SAE International, "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles," SAE International, 2018.

[2] S. Zug, C. Steup, C. Berger, O. Landsiedel, F. Schuldt, J. Rieken, T. Form, R. Matthaei and J. B. Scholle, "Technical evaluation of the Carolo-Cup 2014 - A competition for self-driving miniature cars," in *2014 IEEE International Symposium on Robotic and Sensors Environments (ROSE) Proceedings*, Timisoara, Romania, 2014.

[3] Technische Universität Braunschweig, "Carolo-Cup Regulations 2018," 19 December 2017. [Online]. Available: https://wiki.ifr.ing.tu-bs.de/carolocup/en/system/files/Carolo-Cup%20Regulations_1.pdf. [Accessed April 2019].

[4] Technische Universität Braunschweig, "Carolo-Cup Regulations 2019," 10 July 2018. [Online]. Available: https://wiki.ifr.ing.tu-bs.de/carolocup/en/system/files/180710_Carolo-Cup%20Regulations.pdf. [Accessed June 2019].

[5] Technische Universitat Braunschweig, "Carolo Cup 2018 - Ergebnisse," 14 February 2018. [Online]. Available: https://wiki.ifr.ing.tu-bs.de/carolocup/system/files/Ergebnisse_2018.pdf. [Accessed March 2019].

[6] T. Bräunl, M. Pham, F. Hidalgo, R. Keat and H. Wahyu, "EyeBot 7 User Guide," 29 August 2018. [Online]. Available: http://robotics.ee.uwa.edu.au/eyebot/EyeBot7-UserGuide.pdf. [Accessed March 2019].

[7] T. Bräunl, "EyeBot: a family of autonomous mobile robots," in *ANZIIS'99 & ANNES'99 & ACNN'99. 6th International Conference on Neural Information Processing. Proceedings (Cat. No.99EX378). Vol. 2. IEEE, 1999. 645–649 vol.2*, Perth, 1999.

[8] T. Bräunl, "EyeBot 7," 2019. [Online]. Available: http://robotics.ee.uwa.edu.au/eyebot/. [Accessed March 2019].

[9] Y. Saadna and A. Behloul, "An overview of traffic sign detection and classification methods," *International Journal of Multimedia Information Retrieval,* pp. 193-210, September 2017.

[10] K. Brkic, “An Overview of Traffic Sign Detection Methods,” Department of Electronics, Microelectronics, Computer and Intelligent Systems Faculty of Electrical Engineering and Computing , Unska, 2010.

[11] P. Dewan, R. Vig, N. Shukla and B. K. Das, “An Overview of Traffic Signs Recognition Methods,” *International Journal of Computer Applications (0975 – 8887),* vol. 168, no. 11, pp. 7-11, 2017.

[12] J. Greenhalgh and M. Mirmehdi, “Real-Time Detection and Recognition of Road Traffic Signs,” *IEEE Transactions on Intelligent Transportation Systems,* vol. 13, no. 4, pp. 1498 - 1506, 2012.

[13] G. Arminger and D. Enache, “Statistical Models and Artificial Neural Networks,” in *Data Analysis and Information Systems. Studies in Classification, Data Analysis, and Knowledge Organisation*, Berlin, Heidelberg, Springer, 1996, pp. 243-260.

[14] A. Voulodimos, N. Doulamis, A. Doulamis and E. Protopapadakis, “Deep Learning for Computer Vision,” *Computational Intelligence and Neuroscience,* vol. 2018, p. 13, 2018.

[15] H. Abdi, D. Valentin and B. Edelman, Neural Networks, London: SAGE, 1999.

[16] F. Rosenblatt, “The Perceptron: A Probabilistic Model for Information Storage and Organisation in the Brain,” *Psychological Review,* vol. 65, no. 6, pp. 386-408, 1958.

[17] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto and H. Adam, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” 2017.

[18] S. Albawi, T. A. Mohammed and A.-Z. Saad, “Understanding a Convolutional Neural Network,” ICET2017, Antalya, 2017.

[19] R. Girshick, J. Donahue, T. Darrell and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” UC Berkeley, 2014.

[20] G. Ross, “Fast R-CNN,” 2015.

[21] S. Ren, K. He, R. Girshick and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”.

[22] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, 2016.

[23] T.-Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, "Focal Loss for Dense Object Detection," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Venice, 2017.

[24] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and A. C. Berg, "SSD: Single Shot MultiBox Detector," 2015.

[25] NVIDIA Corporation, "Jetson Nano: Deep Learning Inference Benchmarks," 2019. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-dl-inference-benchmarks. [Accessed June 2019].

[26] V. Badrinarayanan, A. Kendall and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 39, no. 12, pp. 2481-2495, 2 January 2017.

[27] H. Alhaija, S. Mustikovela, L. Mesch, A. Geiger and C. Rother, "Augmented Reality Meets Computer Vision: Efficient Data Generation for Urban Driving Scenes," *International Journal of Computer Vision (IJCV),* 2018.

[28] M. Cordts, M. Omran, S. Ramos, T. Rehfeld and M. Enzweiler, "The Cityscapes Dataset for Semantic Urban Scene Understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[29] O. Zendel, K. Honauer, M. Murschitz and D. Steininger, "WildDash - Creating Hazard-Aware Benchmarks," in *The European Conference on Computer Vision (ECCV)*, Munich.

[30] F. Chollet, "Keras," 2015. [Online]. Available: https://keras.io.

[31] L. Smith, "Cyclical Learning Rates for Training Neural Networks," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017.

[32] Y. Zhu, K. Sapra, F. A. Reda, K. J. Shih, S. Newsam, A. Tao and B. Catanzaro, "Improving Semantic Segmentation via Video Propagation and Label Relaxation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, 2019.

[33] J.-A. Bolte and M. Kamp and A. Breuer and S. Homoceanu and P. Schlicht and F. Hüger and D. Lipinski and T. Fingscheidt, "Unsupervised Domain Adaptation to Improve Image Segmentation Quality Both in the Source and Target Domain," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, 2019.

[34] G. Yang, H. Zhao, J. Shi, Z. Deng and J. Jia, "SegStereo: Exploiting Semantic Information for Disparity Estimation," in *European Conference on Computer Vision (ECCV)*, Munich, 2018.

[35] M. Ochs, A. Kretz and R. Mester, "SDNet: Semantic Guided Depth Estimation Network," in *German Conference on Pattern Recognition (GCPR)*, Dortmund, 2019.

[36] S. Kong and C. Fowlkes, "Pixel-wise Attentional Gating for Parsimonious Pixel Labeling," in *arXiv 1805.01556*, 2018.

[37] Institut für Neuroinformatik, "The German Traffic Sign Recognition Benchmark," 16 April 2019. [Online]. Available: http://benchmark.ini.rub.de/dev/index.php?section=gtsrb&subsection=news. [Accessed June 2019].

[38] Á. Arcoa-García, J. A. Arcoa-García and L. M. Soria-Morillo, "Deep neural network for traffic sign recognition systems: An analysis of spatial transformers and stochastic optimisation methods," *Neural Networks,* vol. 99, pp. 158-165, 2018.

[39] D. Cireşan, U. Meier, J. Masci and J. Schmidhuber, "Multi-column deep neural network for traffic sign classification," *Neural Networks,* vol. 32, pp. 333-338, 2012.

[40] B. Gecer, G. Azzopardi and N. Petkov, "Color-blob-based COSFIRE filters for object recognition," *Image and Vision Computing,* vol. 57, pp. 165-174, 2017.

[41] P. Sermanet and Y. LeCun, "Traffic sign recognition with multi-scale Convolutional Networks," in *International Joint Conference on Neural Networks*, San Jose, CA, 2011.

[42] F. Zaklouta, B. Stanciulescu and O. Hamdoun, "Traffic sign classification using K-d trees and Random Forests," in *The 2011 International Joint Conference on Neural Networks*, San Jose, CA, 2151-2155.

[43] J. Stallkamp, M. Schlipsing, J. Salmen and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks,* vol. 32, pp. 323-332, 2012.

[44] S. Zug, C. Steup, J.-B. Scholle, C. Berger, O. Landsiedel, F. Schuldt, J. Rieken, R. Matthaei and T. Form, "Technical evaluation of the Carolo-Cup 2014 - A competition for self-driving miniature cars," in *2014 IEEE International Symposium on Robotic and Sensors Environments (ROSE) Proceedings*, Timisoara, 2014.

[45] B. C. Zanchin, R. Adamshuk, M. M. Santos and K. S. Collazos, "On the instrumentation and classification of autonomous cars," in *IEEE Internation Conference on Systems, Man, and Cybernetics (SMC)*, Banff, AB, Canada, 2017.

[46] R. Szeliski, Computer Vision: Algorithms and Applications, Springer, 2010.

[47] S. Singh, "Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey," NHTSA's National Center for Statistics and Analysis, Washington, DC, 2015.

[48] Y. Saadna and A. Behloul, "An overview of traffic sign detection and classification methods," *International Journal of Multimedia Information Retrieval,* vol. VI, no. 3, pp. 193-210, September 2017.

[49] A. R. Pathak, M. Pandey and S. Rautaray, "Application of Deep Learning for Object Detection," *International Conference on Computational Intelligence and Data Science,* pp. 1706-1717, 2018.

[50] A. Koestler and T. Bräunl, "Mobile Robot Simulation with Realistic Error Models," in *Proceedings of the Second International Conference on Autonomous Robots and Agents*, Palmerston North, New Zealand, 2004.

[51] R. Hussain, J. Lee and S. Zeadally, "Autonomous Cars: Social and Economic Implications," *IT Professional,* vol. 20, no. 6, pp. 70-77, Nov.-Dec. 2018.

[52] M. Hirz and B. Walzel, "Sensor and object recognition technologies for self-driving cars," *Computer-Aided Design and Applications,* pp. 501-508, January 2018.

[53] A. Finn, J. Adam, M. Del Rose, B. Kania, J. Overholt, U. Silva, J. Bornstein, A. Hsieh and S. Lacroix, "Evaluating autonomous ground-robots," *Journal of Field Robotics,* vol. 29, no. 5, pp. 689-706, September 2012.

[54] P. Dewan, R. Vig, N. Shukla and B. K. Das, "An Overview of Traffic Signs Recognition Methods," *Internation Journal of Computer Applications,* vol. 168, no. 11, pp. 7-11, June 2017.

[55] T. Bräunl, "EyeBot7," 2019. [Online]. Available: http://robotics.ee.uwa.edu.au/eyebot/. [Accessed June 2019].

[56] T. Bräunl, M. Pham, F. Hidalgo, R. Keat and H. Wahyu, "EyeBot 7 User Guide," 29 August 2018. [Online]. Available: http://robotics.ee.uwa.edu.au/eyebot/EyeBot7-UserGuide.pdf. [Accessed June 2019].

[57] T. Bräunl, Embedded Robotics - Mobile Robot Design and Applications with Embedded Systems, Third Edition, Heidelberg Berlin: Springer-Verlag, 2008, pp. (XIV, 541).

[58] J. Billington, "Audi and Volkswagen to use advanced lidar technology in autonomous mobility fleet," Autonomous Vehicle International, 21 December 2018. [Online]. Available: https://www.autonomousvehicleinternational.com/news/sensors/audi-and-volkswagen-to-use-advanced-lidar-technology-in-its-autonomous-mobility-fleet.html. [Accessed April 2019].

[59] S. Albawi, T. A. Mohammed and S. Alzawi, "Understanding of a Convolutional Neural Network," in *The International Conference on Engineering and Technology*, At Antalya, 2017.

[60] National Highway Traffic Safety Administration, "Preliminary Statement of Policy Concerning Automated Vehicles," [Online]. Available: https://www.nhtsa.gov/staticfiles/rulemaking/pdf/Automated_Vehicles_Policy.pdf. [Accessed June 2019].

[61] "FPGA-Based Traffic Sign Recognition for Advanced Driver Assistance Systems," *Journal of Transportation Technologies,* vol. 3, no. 1, pp. 1-16, 2013.

[62] FPL, "FlatBuffers," [Online]. Available: https://google.github.io/flatbuffers/. [Accessed June 2019].

[63] EyeSim VR Team, "EyeSim VR User's Manual," 13 November 2018. [Online]. Available: http://robotics.ee.uwa.edu.au/eyesim/ftp/EyeSim-UserManual.pdf. [Accessed March 2019].

[64] Audi, "Audi Autonomous Driving Cup 2018," Audi AI, 2018. [Online]. Available: https://www.audi-autonomous-driving-cup.com/competition/overview/. [Accessed June 2019].

[65] S. Maldonado-Bascon, S. Lafuente-Arroyo, P. Gil-Jimenez, H. Gomez-Moreno and F. Lopez-Ferreras, "Road-Sign Detection and Recognition Based on Support Vector Machines," *IEEE Transactions on Intelligent Transportation Systems,* vol. 8, no. 2, pp. 264-278, 2007.

[66] U.S. Department of Transportation, "Preparing for the Future of Transportation," October 2018. [Online]. Available: https://www.transportation.gov/av/3/preparing-future-transportation-automated-vehicles-3. [Accessed June 2019].

[67] J. F. Khan, S. M. A. Bhuiyan and R. R. Adhami, "Image Segmentation and Shape Analysis for Road-Sign Detection," *IEEE Transactions on Intelligent Transportation Systems,* vol. 12, no. 1, pp. 83-96, 2011.