MECH0020 Individual Project

2022/23

Student:	Kieran John Anthony Pereira
Student ID:	20072669
Project Title:	MACHINE-LEARNED TRAFFIC SIGN
	NARRATION SYSTEM: A HOLISTIC APPROACH
Supervisor:	Professor Manish Tiwari

Word Count: 7498

Abstract

Increasing driver safety has been a focal point of car design since the invention of the seatbelt in 1959; however, despite safety improvements, there remains a vast imbalance in car-crash fatalities by age demographic; with young, inexperienced and elderly drivers considered higher risk than other categories of drivers. Driver inattention and distractions are the root cause of 56% of such fatalities each year, especially due to poor observation of road signs.

This report aims to address poor driver awareness by developing a novel machine learning algorithm capable of detecting and narrating traffic signs in real-time that can be incorporated into a vehicle dashboard system. A thorough literature review was initially conducted to find the most suitable object detection method for this algorithm to be developed into a working prototype, with Convolutional Neural Networks chosen as the optimal model.

From gaps from the literature review, this project improves current conventional training and testing processes used to develop detection algorithms. It also proposes a methodology for training a Convolutional Neural Network on a diverse globally distributed dataset of highway code traffic signs, using an approach specifically created for this project for dataset processing and image cropping. The resulting model achieved an 84.4% Mean Average Precision and a 0.81 F1 score after training, achieving state-of-the-art performance akin to the literature reviewed, despite training on a more diverse and complex dataset.

The model was tested against a large dataset and specific scenarios aimed at determining its robustness to external factors such as visibility, obstruction and weather tests. The model proved its robustness to external influencing factors, obtaining a testing F1 score of 0.711. The narration part of the algorithm development was created, deployed and tested using real-time video.

Given the state-of-the-art performance of the model, the machine learning algorithm created from this project, through commercialisation of a low-cost product, has the potential to not only increase driver awareness on our roads but save lives.

Student Declaration

I, Kieran John Anthony Pereira, confirm the work presented in this report is my own. Where information has been derived from other sources, I confirm this has been indicated in the report.

The code created to develop the trained Convolutional Neural Network is contained within a Google Colaboratory Jupyter Notebook ("Traffic_Sign_Dataset_Processing_&_Training.ipynb"). The narration algorithm was created in Python (Final_Algorithm.py).

All code and supporting files can be found in the GitHub Repository, which has also been provided as supporting material in a zipped folder titled "*Compressed GitHub Repository.zip*":

(https://github.com/KieranPereira/Traffic-Sign-Detection)

The table below shows the area of code supporting each section of this report.

Acknowledgements

I would like to thank my personal tutor and supervisor, Dr Manish Tiwari for supporting me throughout the year on this project. His insightful advice and constructive criticism were integral in realising this project's full potential and his suggestions played a crucial role in structuring this report, serving as the foundations for its creation.

Furthermore, I would like to extend my sincere gratitude to Dr Rohit Gupta, who helped guide this project throughout the year. I would like to thank Rohit for taking the time to meet with me on a weekly basis which provided productive discussions for brainstorming ideas and overcoming the challenges faced during the project. Without guidance his guidance and genuine interest, this project would not have been possible to complete.

Table of Contents

APPENDICIES

List of Figures

List of Tables

Nomenclature

CV:	Computer Vision
RGB:	Red, Green, Blue
CNN:	Convolutional Neural Network
mAP:	mean Average Precision
YOLO:	You Only Look Once
MTSD:	Mapillary Traffic Sign Dataset
RAM	Random Access Memory
GPU	Graphics Processing Unit
CPU	Central Processing Unit
FPS:	Frames Per Second
IoU:	Intersection over Union
BCE:	Binary Cross Entropy

1.0 Introduction

1.1 Problem Definition

Skills to drive motorised vehicles are acquired through experience and situational learning. Vigilance on the road, surroundings, together with navigating and monitoring a vehicle's cabin instruments can be challenging for all drivers especially, young [1], inexperienced and elderly [2] drivers.

Young drivers in the United Kingdom are the largest group killed or seriously injured in car accidents [1], representing 32% of fatalities as shown Figure 1. Studies indicate injury or death in car accidents increases by over 22% for elderly drivers, where impaired vision and slower reaction times, due to cognitive and physical decline, contribute to this statistic [2]. Research suggests driver inattention and distraction are the primary causes of road accidents [3,4], with the most overlooked risk being highway code signs, exacerbated by 44% of drivers forgetting important traffic sign meanings [5]. This research suggests a key underlying problem common to both demographics is lack of perception. This is concerning, given the increased amount of roadside advertising and other environmental clutter with studies supporting young drivers invest more attention to road advertising than highway code signs [6]

With technological breakthroughs like Tesla's Autopilot, the technology to increase driver awareness is proven and available in the automotive industry. Nevertheless, given the high cost of such vehicles and evidence demonstrating young drivers are substantially likely to buy used vehicles [7], this technology is far from reaching the demographics that need it most.

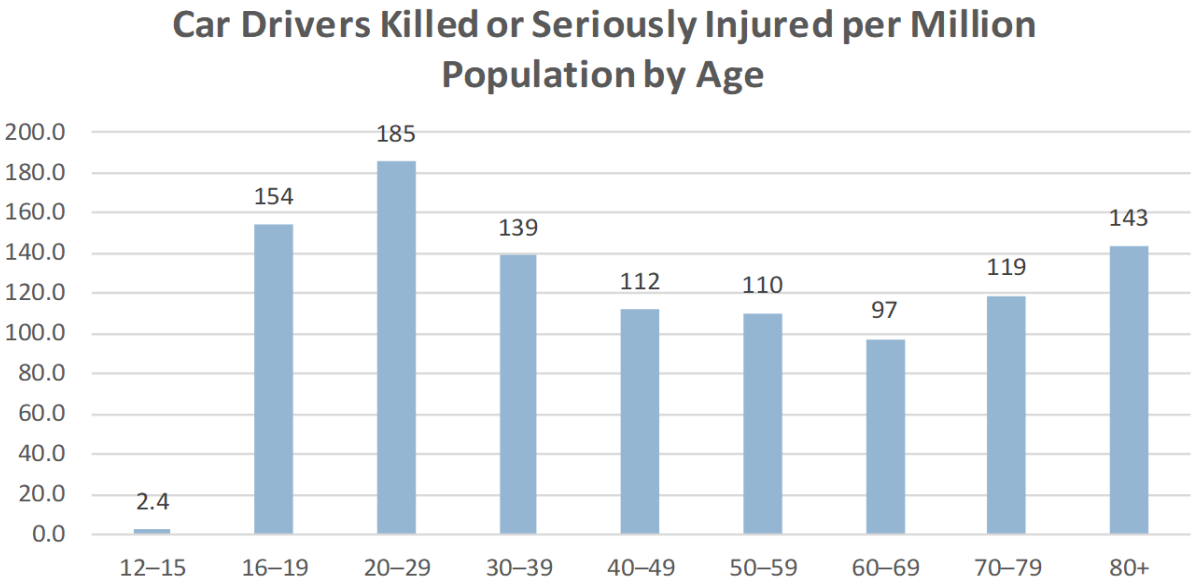


Figure 1: UK Driving Statistics for 2022 showing the number of car drivers killed or seriously injured per million population by age [1].

[1] Drivers less than 30 years old
[2] Drivers more than 69 years old

1.2 Aims & Objectives

The aim of this project is to deliver a low-cost solution to improve awareness and perception by providing early notifications of oncoming traffic signs using audio alerts to aid drivers of road and safety conditions ahead. It is specifically targeted to drivers with poor perception found in the demographics described earlier.

In delivering this, the project's primary objective is to construct a machine learning algorithm that recognises, interprets and narrates in natural language the meaning of essential traffic signs in real time using Computer Vision (CV), a field of artificial intelligence that extracts information from images. The project fulfils a secondary objective borne from the literature review and described further in section 2.2.

2.0 Literature Review

2.1 Current Work

Since the invention of the seatbelt in 1959 [8], driver safety is a key focus in vehicle design. With advancements in computing, efforts have focused on improving driver perception using CV, a technology already used for lane departure warnings and driverless cars. The following CV methods were reviewed for consideration for this project:

2.1.1 Colour Recognition

Colour Recognition is the process of separating a target object from an image's background using its distinct colour. These algorithms are used in the manufacturing industry [9], where products are intentionally coloured distinctly for tracking [10]. This detection method requires low resources and can be optimised for speed [11], maintain high accuracies with many instances [12], and easily tuned for sensitivity [13]. Figure 2 illustrates an example of separating of small objects from an image.

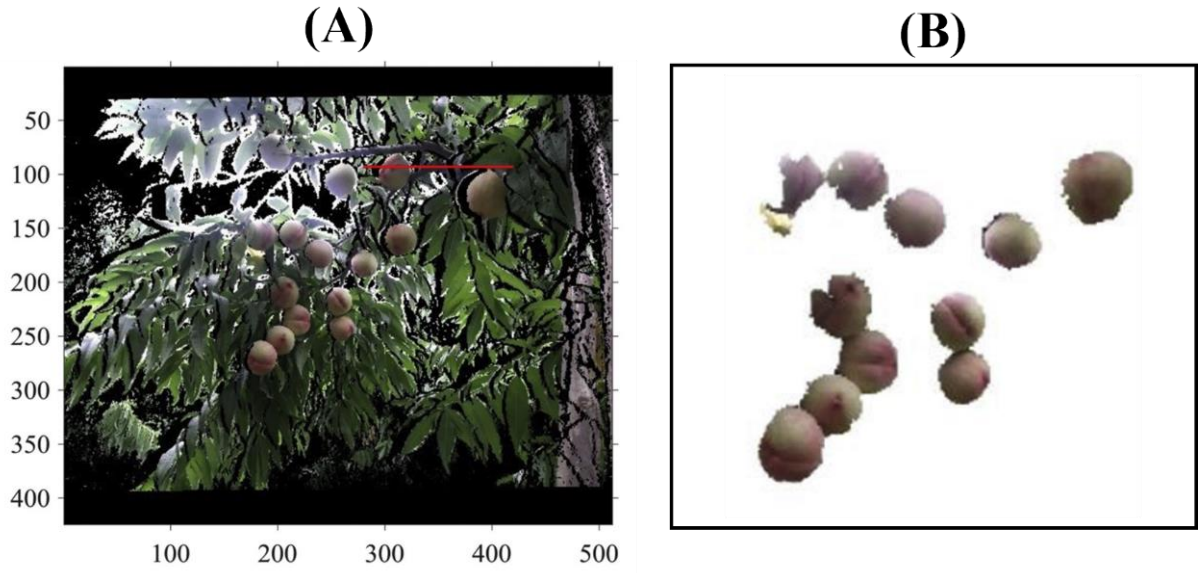


Figure 2: Images highlighting the extraction of relevant objects in an image: (A) Original image, (B) resulting image showing desired objects after colour filtering [12].

As all traffic signs are distinctly coloured, Colour Recognition can be applied to recognise traffic signs by setting appropriate thresholds to detect specific colours of such signs for further interpretation. Table 1 illustrates accuracy levels of Colour Recognition algorithms from three commonly used methods.

Table 1: A Comparison of Different Algorithms Using Colour Detection to Detect and Classify Traffic Signs.

Ref.	Year	Method	Colours Detected	Accuracy
[14]	2010	Colour (RGB) Thresholding	Red, Blue, Yellow	85.3%
[15]	2007	Hue and colour Thresholding	Red	86%
[16]	2017	Support Vector Machine Learning for pixel classification	Red, Blue, Yellow	96%

Whilst state-of-the-art Colour Recognition algorithms can boast high accuracies, these are negatively impacted in the surroundings where:

a) There is poor lighting or visibility due to weather and dirt. Such limitations have been addressed in other research [17], [18], showing how recognition can be improved by changes in brightness, contrast, colour constancy and colour intensity. However positive detections remain low in adverse conditions impacting visibility (e.g. fog and snow) and excessive headlight glare.

b) There are traffic signs and road advertisements of similar colours near each other. Colour Recognition algorithms cannot distinguish between such signs and may blend the two.

2.1.2 Shape Recognition

Shape detection is a CV technique used to identify geometric shapes in photos or videos. The recognition algorithm pre-processes an image to calculate the number of edges, curves and corners of the target object contained within, allowing its shape to be predicted. This is achieved using techniques like the Hough Transform to extract lines, circles and ellipses from the image [19]. Shape detection is also used in the automotive industry, for example; to detect occupied spaces in car parks, shown Figure 3 [19] where a Shape Recognition algorithm uses the Hough Transform method to find small rectangles which it classifies as cars, thus determining locations of empty parking spaces.

Figure 3: Image of a parking lot. Vehicles are outlined as rectangles by via Shape Recognition methods [19].

Traffic signs use pre-defined shapes to provide meaning where outer circular shapes signify Orders, triangular shapes, “Warnings” and rectangular shapes, “Information”. Table 2 highlights the accuracy levels from relevant research in the field of traffic sign shape detection from four commonly used methods.

Table 2: A Comparison of Different Algorithms Using Shape Detection to Detect and Classify Traffic Signs.

Ref.	Year	Method	Shapes Detected	Accuracy
[20]	2010	Hough Transform	Triangles, Diamonds, Squares, Circles	83.67%
[21]	2014	Optimal Corner Detection [22]	Triangles, Circles	87%
[23]	2007	Optimal Corner Detection, Colour Recognition	Circles, Triangles, Rectangles	95.5%
[24]	2015	Hough Transform	Circles, Triangles	97.3%

With reference to the models in Table 2, the method for interpreting the meaning of traffic signs is derived from a two-step process, illustrated in Figure 4. First, the sign is determined by identifying the recognisable shapes within the image, giving the sign's category. The inner symbol of the identified shape is then searched for within, which gives the classification of the sign.

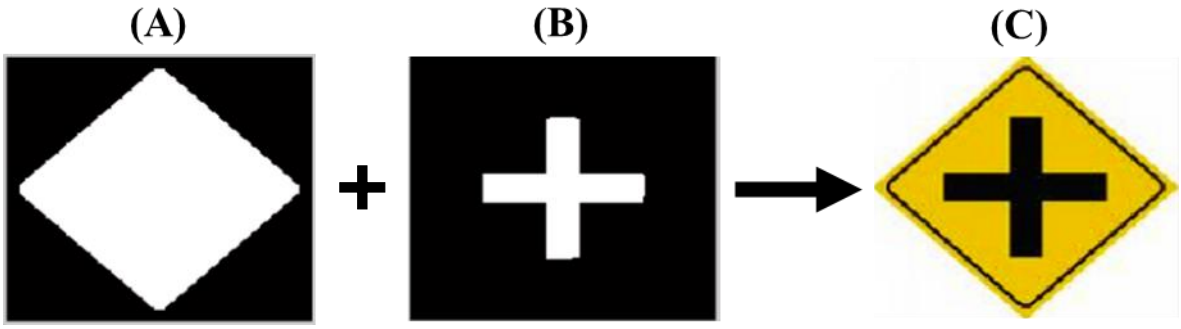


Figure 4: The application of shape detection for traffic signs as modelled by *Sallah, et al* [21]. Combining the first two images, (A), (B) gives the correct prediction shown in (C).

The effectiveness of these algorithms for traffic use is subject to partial obstructions and the degree of degradation of both the shape of the sign and symbols within [25]. These methods scale poorly where the interpretation of a sign uses slightly different symbols in different countries. Algorithms relying on the number of edges or corners are easily confused when the number of predicted edges does not match the dataset, leading to misclassification, an effect worsened with distance, as edges appear as curves the further the sign is from the road [26].

2.1.3 Deep Learning

Object detection using Deep Learning methods is one of the most popular methods for real-time detection [27]. These methods use Convolutional Neural Networks (CNN) to detect desired objects from an image. Before being able to detect target objects, CNNs must be trained and validated, allowing it to learn the features that differentiate the target object from other objects. Following this, CNNs makes detections by extracting important features such as the colours, lines and shapes, combining these to form geometric features and colour profiles of objects. The features for the target object are recognised by patterns learnt by the CNN, allowing detections to be made.

The training and validation process of developing a CNN is illustrated in Figure 5. First, the untrained CNN is given annotated images from a training dataset from which it learns the features of target objects by updating its internal parameters (such as weights and biases). For each image in the training set, the CNN makes a prediction of the class (category of object) and location (drawn as a box) of the target object within the image. This prediction is compared against the actual class and location of the object and the difference between the two is quantified by a loss function, the lower this difference the closer the predicted image represents the actual. State-of-the-art performance is accepted when this difference is below 0.02 [28].

For each iteration (epoch) of training, the loss function is re-calculated and the aforementioned parameters are self-tuned to reduce the loss function to the desired level achieving state of the art performance. Between each epoch, the CNN is also tested on new images separate from the training data know as validation data. This step evaluates the CNN's accuracy on unseen images, giving a rough indication of how well it performs on new images which allows performance metrices to be calculated, such as the Mean Average Precision and F1 scores. The validation stage gives an indication of how quickly CNN learns the features of the target objects. Based on this indication, adjustments to the training process can be made, by tuning the hyperparameters which helps improve the training process. Finally, once a final detection model has been made, it is tested on a large testing dataset which gives an unbiased indication of how well the CNN detects new objects and if necessary, the model can be retrained based on testing results.

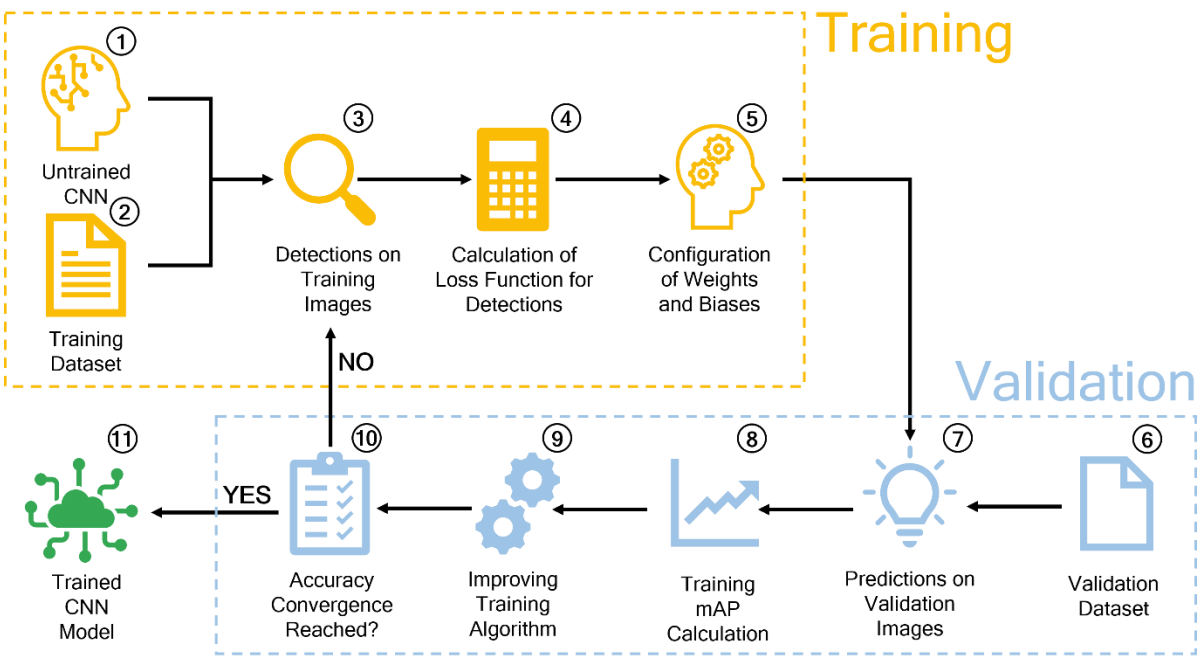


Figure 5: A flowchart representing the process of training and validation to produce a successful CNN model capable of detection target objects

Advancements in computational processing have led to increased use of CNNs in complex applications such as detecting early signs of brain tumours [29], identifying quality control faults in manufacturing [30]and detecting pedestrians and vehicles to enable autonomous navigation [31]. Firms like Nissan, Toyota and Tesla [32]use CNNs due to its robustness potential as the model's accuracy is tied to the quality of the training dataset, which can be easily diversified and expanded. Although a number of companies have employed Deep Learning algorithms for speed detection cued from speed signs, as depicted in Figure 6 [33], they have yet to fully utilise the capabilities of interpreting other traffic signs for road safety altering.

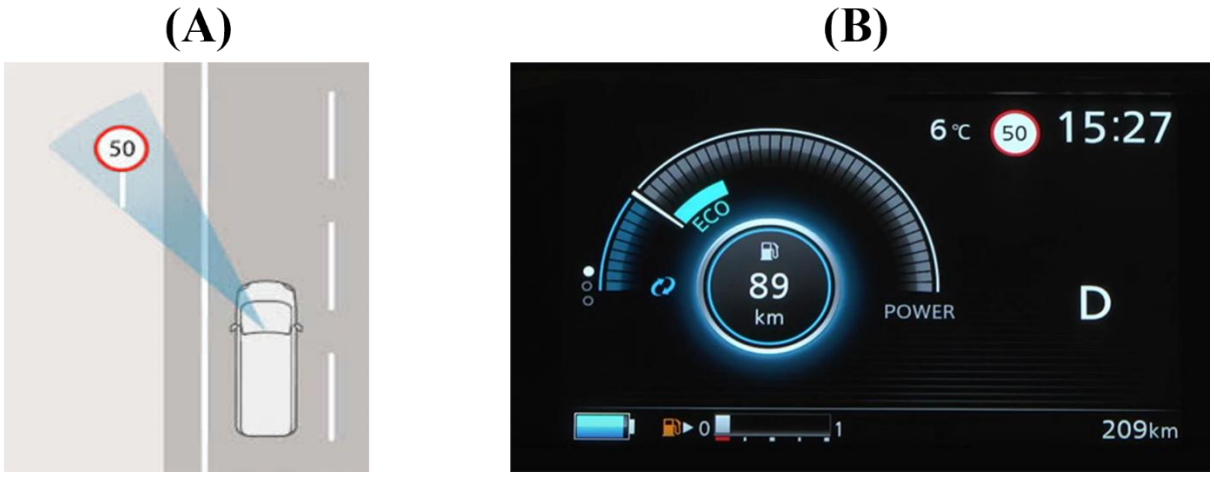


Figure 6: (A) A diagram illustrating how an on-board camera scans a traffic sign ahead (B) Picture of dashboard information shown the speed limit from the sign detected [33].

Recent works in traffic sign detection is given in Table 3. The 'You Only Look Once (YOLO)' algorithm is a popular method optimised for its fast detections, as predicting object classes and bounding boxes is done in one step. This open-source algorithm is used by industry leaders in autonomous navigation, such as Tesla [34].

Table 3: A Comparison of Different Algorithms Using Deep Learning Methods to Detect and Classify Traffic Signs.

Ref.	Year	Algorithm	Training dataset size	Number of classes	Accuracy	Location
[35]	2021	Faster-RCNN	10,842	54	76%	Turkey
[36]	2020	TensorFlow Object Detection API	2000	56	94.63%	Germany, Belgium
[37]	2022	YOLOv3	900	43	97.22%	Germany
[38]	2022	Custom CNN	11,074	23	98.45%	Taiwan
[39]	2021	YOLOv4	11,335	11	64.71%	Global
[40]	2016	Custom built	2000	10	97.69%	Sweden
[41]	2022	Yolov5	2,182	8	97.70%	Custom

Despite recent interest and investment, Deep Learning algorithms do have limitations which can inhibit performance. A machine learning model's accuracy strongly depends on its dataset quality. Having a dataset that depicts the diversity of items in the real world is critical for achieving reliable results in real-life scenarios. It is crucial that the dataset is balanced and unbiased to prevent the model overfitting, occurring when a model becomes too closely associated to the original dataset, leading to excellent training accuracy but poor testing accuracy. Bias arises from flaws within the dataset where the distribution of images are not representative to the population being observed. An example of the effects of high bias was IBM's gender classifier model, where performance worsened for darker-skinned females due lower instances in training data [42]

.

2.2 Literature Gap and Objectives

2.2.1 Literature Gap

The literature review identified three CV methods for traffic sign detection. Deep Learning using CNNs offered superior advantages over Colour and Shape Recognition, including greater robustness, error correction and its ability to detect multiple classes simultaneously. Consequently, these methods were chosen as the preferred CV method to prototype the traffic signs detection algorithm for this project.

The literature review also highlighted three areas in which the results of research studies were subject to debate. Firstly, the high Mean Average Precision (mAP) and F1 scores from the research documented in several papers, such as [36,37], is debateable, as they use simplified datasets with limited variance in images and a low number of classes [40,41]. In addition, the models were not tested on separate testing data, giving no indication whether these high scores were a consequence of overfitting.

Additional areas of opportunity included testing a model's performance on realistic scenarios, such as weather or brightness, which had not been explored in current research. Testing these scenarios would ensure the detection model is suitable for deployment.

Finally, considering geographic variety is crucial in ensuring the algorithm's applicability in other regions. From the literature considered, few papers discussed the potential scope of traffic sign detection models, those that do, its training is often localised to specific regions. However, with global signs, this is a significant limitation which this project attempts to address.

2.2.2 Success Criteria and Novelty

In addition to the aims and objectives outlined in Section 1.1, the project expanded its objective scope to address the aforementioned gaps in research for the field of traffic sign detection with the following success criteria:

A. For training and testing, the project must use a dataset containing a variance of images in different conditions, captured in different weather and brightness settings.
B. An assessment of the model's performance on different conditions including weather, light and obstructions must be carried out to assess suitability for deployment.
C. The project must test on a large dataset to ensure the algorithm is robust to variance. Since there is no universal definition of a large dataset, 5000 images were used as a benchmark taking reference that the dataset size of the research reviewed ranges from 900- 41,909 images.
D. The training dataset for this project must be globally distributed to ensure the model can be used in different countries to increase the utility of the algorithm.
E. The dataset for this project must include a minimum of 59 traffic classes categorised as regulatory or warning signs. This makes up 60% of these signs as defined in the UK highway code [43]. Forty percent of these signs were disregarded as it is unfeasible to have sufficient data for all these signs.
F. The final trained CNN must achieve state-of-the-art performance after training, achieved when all loss functions are below 0.02 [44] and a mAP score above 81.1%, given as the benchmark for global datasets [45].

3.0 Methodology

Drawing inference from methodologies discussed from the Literature Review, this report follows a framework for developing the final narration algorithm as shown in Figure 7.

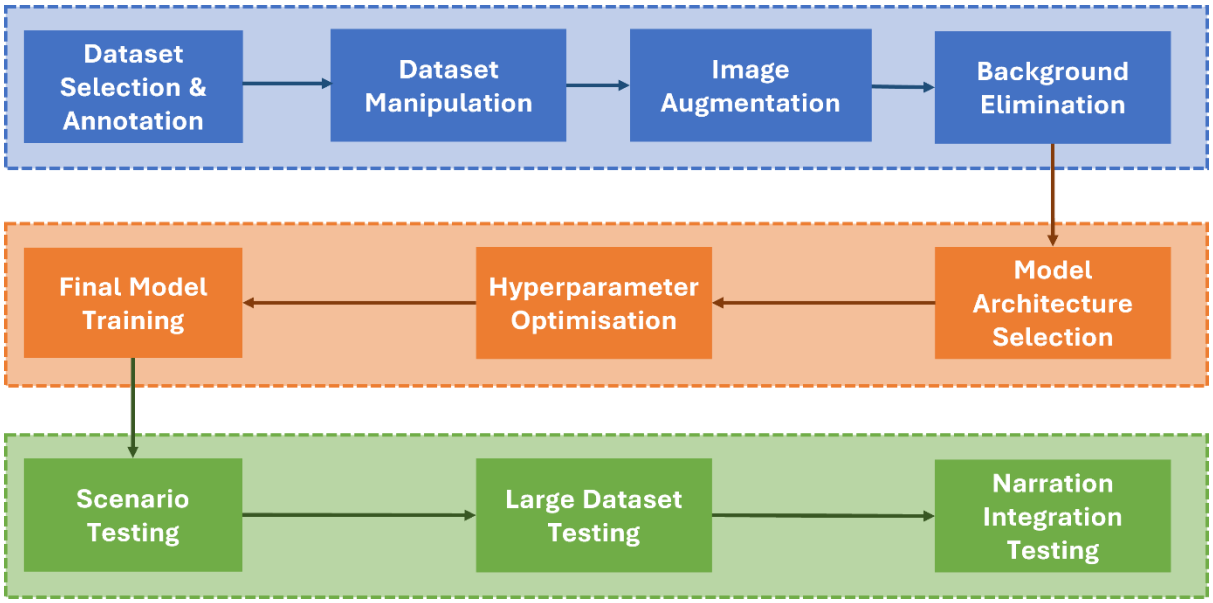


Figure 7: Design flowchart for developing the final narration algorithm.

3.1 Dataset Selection

The dataset chosen must have a sufficient number of images, be globally distributed and include at least 59 regulatory/warning traffic sign classes to fit criteria C and D in Section 2.2.2. This project was constrained in using open-source traffic sign datasets, a list of these is given in Table 4.

Table 4: Publicly Available Traffic Sign Datasets with All Relevant Attributes

Name	Number of images	Location	Number of classes
Kaggle Traffic Sign Detection Dataset	877	Global	4
Chinese Traffic Sign Detection	6,164	China	58
Mapillary Traffic Sign Dataset	100,000	Global	401
DFG Traffic Sign Dataset	7,000	Slovenia	200
Swedish Traffic Sign Dataset	4,000	Sweden	6
German Traffic Sign Recognition Benchmark	50,000	Germany	43
Russian Traffic Sign Images Dataset	100,000	Russia	156

The Mapillary Traffic Sign Dataset (MTSD) was most suited to the success criteria C and D given in Section 2.2.2 and was chosen as the dataset for development of the detection model. With enough classes and images, the MTSD is globally distributed as shown in Figure 8, showing good representation from different regions.

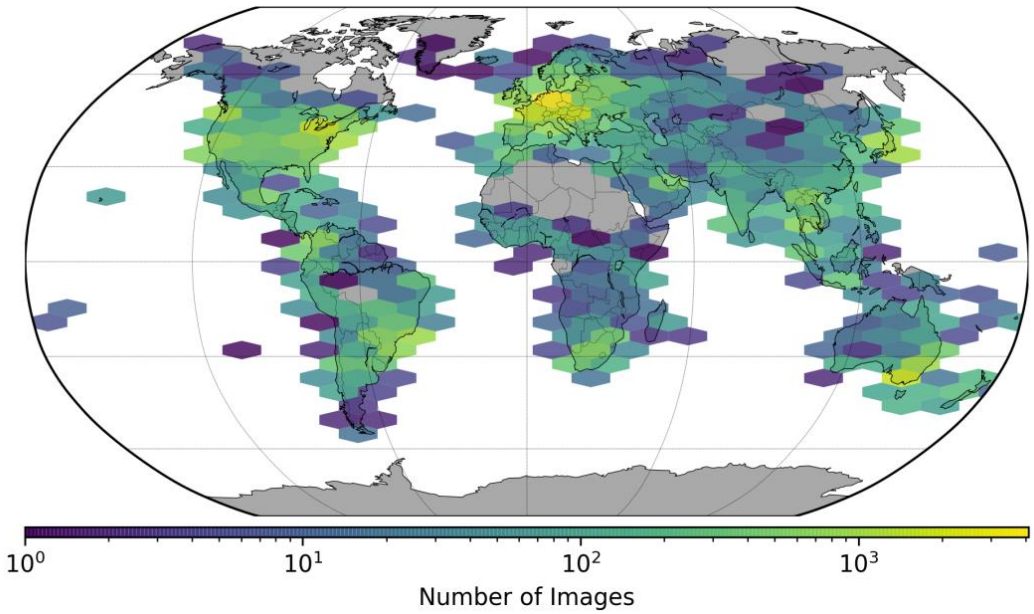


Figure 8: Distribution of images for the Mapillary Traffic Sign Dataset [45]

In addition, the MTSD comprises of 320,000 labelled traffic signs from all images [45], of these, 52,000 are verified and fully annotated and 48,000 partially annotated, with only the traffic sign classes verified for each image. Although the dataset has 401 classes, the distribution of classes is unequal, with over 65% of signs in the dataset labelled as "other", a miscellaneous category that describes signs not within the global highway code. This class was removed from the dataset for this project and a new distribution was calculated with results shown in Figure 9, also illustrated as a bar-chart in Appendix 1, both showing class imbalances.

A limitation of the MTSD is the inclusion of partially verified images, mostly submitted by the public, which are classified as partially annotated. This set of images was removed from the training dataset, to improve model accuracy as these signs are not fully verified.

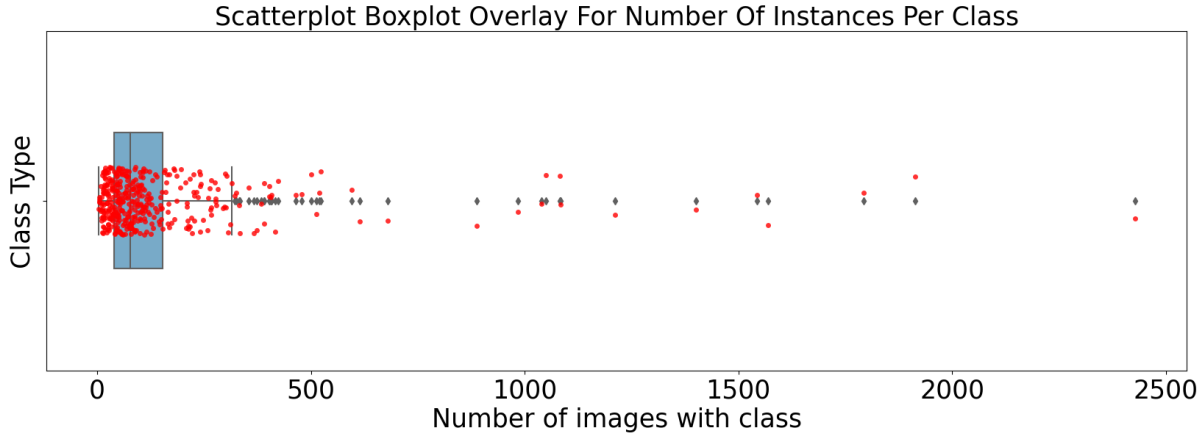


Figure 9: Scatter plot and box plot overlay for number of instances per class after removing traffic signs classified as "other".

3.2 Baseline Performance

YOLOv5 was chosen as the preferred training algorithm due to its fast prediction times for real-time detection. To use this framework the annotation formats describing the location of each sign were converted to be YOLOv5 compatible. The MTSD was then split into three subsets comprising of 36,590 images (65%) for training, 5,321 (10%) for validation and 13,346 images (25%) for testing, although testing was not required at this stage.

As discussed in Section 2.1.3, the aim of training is to minimise the loss functions to achieve state-of-art performance. For the detection of traffic signs detections, there are three loss functions that measure the following areas:

- Box loss measures the difference between the predicted and actual bounding box coordinates of objects within the image, helping train the CNN to accurately locate the position of objects.
- Object Loss measures how well the model correctly predicts the number of traffic signs in an image.
- Class Loss measures how well the model can predict the class of each sign within the image.

These functions were given weighted importance, by prioritising the Class and Object loss functions, important for narration tasks. Section 4.1 shows the results from baseline training.

3.3 Class Reduction

The results found in Section 4.1 suggest the class imbalance within the dataset is causing bias in the model's predictions which is reducing accuracy. Modifying the dataset to smoothen the distribution can be used to reduce bias against underrepresented classes. The following techniques were considered in sequence to smoothen this distribution.

3.3.1 Removing Unnecessary Classes to Problem

This project's aim was to notify drivers of essential information, thus, signs not deemed essential, e.g. "informational" and "complementary" categorised signs, were removed from the dataset. To achieve this, the MTSD class names were referenced, which have the convention '*Sign Category-Meaning-Location Variant'*. Where a "keep-right" sign is given as *'regulatory--keep-right--g1'*. Thus, the unnecessary categories were searched and removed from the dataset.

After removal, a manual review was conducted to determine if other signs could be removed. The exclusion criteria assumed the algorithm's primary users would not be confident drivers, therefore signs relevant only to truck, bus, and taxi drivers were deleted from the dataset. After reduction, 237 classes were removed from the dataset, leaving 164 classes with a distribution shown in Figure 10.

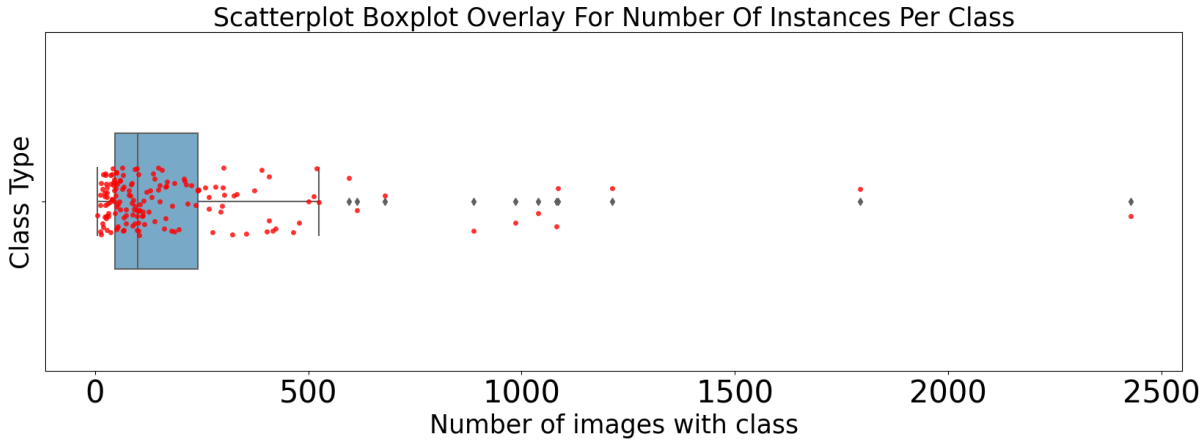


Figure 10: Scatter plot and box plot overlay for number of instances per class after removing unnecessary classes.

3.3.2 Grouping Regional Labels

As outlined in criterion D in Section 2.2.2, the object detection model should not be constrained to have regional limitations. A potential factor influencing accuracy maybe the algorithm's confusion with differentiating different variants, as they look similar. To reduce this error, classes with global variants were grouped, reducing the number of classes in the dataset from 164 to 93, with a distribution shown in Figure 11, showing the reduction of outliers in the dataset signifying a more balanced class distribution.

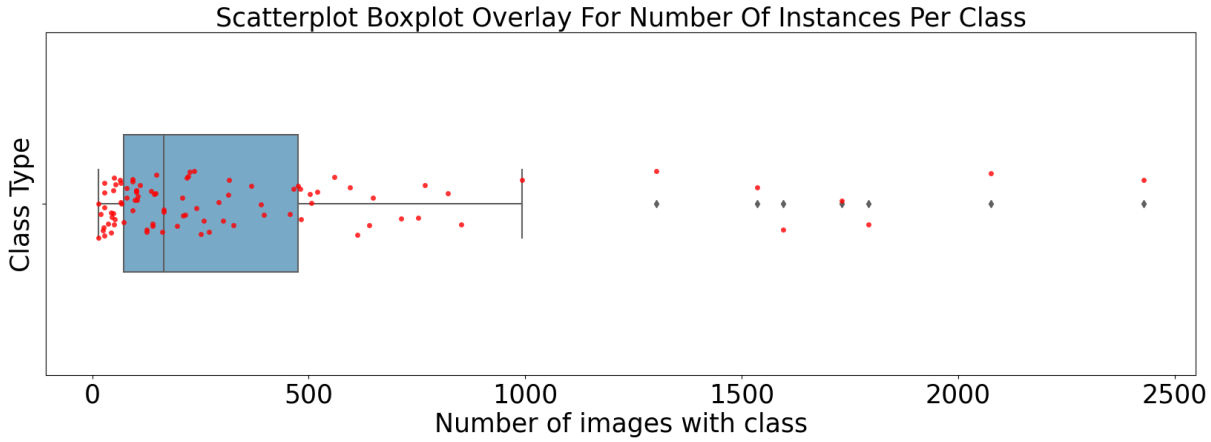


Figure 11: Scatter plot and box plot overlay for number of instances per class after grouping regional labels.

3.3.3 Removal of Low Instance Classes

This approach addresses imbalanced class issues by eliminating underrepresented classes to balance the dataset and reduce computational load during training, however, removing these classes means the model will not consider them when deployed. Therefore, it is important to balance the benefits of class removal with the potential drawbacks of reducing the utility of the model. To achieve this balance whilst

meeting criterion E in Section 2.2.2, signs with fewer than 100 instances were removed, reducing the dataset to 63 classes. The final distribution of images in the dataset can be seen in Figure 12, which when compared to Figures 10 and 11 show the total reduction of classes from the original dataset. The impact of all Class Reduction techniques can be fully comprehended by comparing Appendix 2 (a bar chart of the final distribution of classes) to Appendix 1 (a bar chart of initial distributions of the MTSD). After this final reduction, an untrained CNN was trained using the new dataset, with results in Section 4.2.

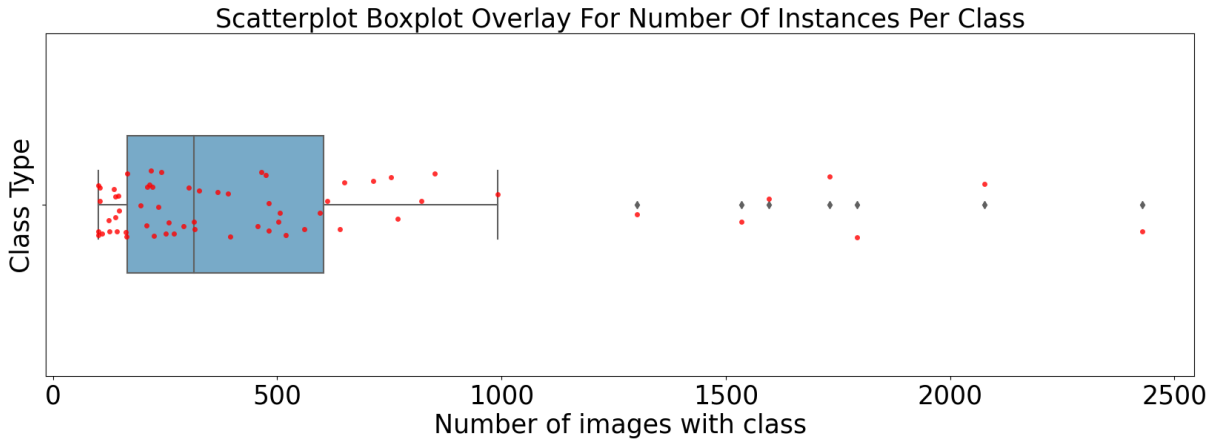


Figure 12: Scatter plot and box plot overlay for number of instances per class after frequency thresholding at 100 instances.

3.4 Image Resizing and Cropping

Further reduction of the dataset cannot be used to improve the accuracy as this will violate criterion E in Section 2.2.2. Therefore, other methods were explored to simplify the data, such as image processing. YOLOv5's resizes all images to a 640 × 640-pixel square as the CNN requires square images for training. Images in the MTSD had an average width and height of 3497 and 2442 respectively, with standard deviations of width and height of 1055 and 749 respectively, shown in Figure 13. This meant images were scaled down by an average factor of 5.5, making labels too small for training. Increasing the image sized used by YOLOv5 was attempted however due to the higher memory usage required for processing, this was not an option for training due to the resource constraints to this project. Cropping methods were explored to reduce all image sizes, make the images' aspect ratios closer to 1:1 (square) reduce the standard deviation of image sizes in the dataset.

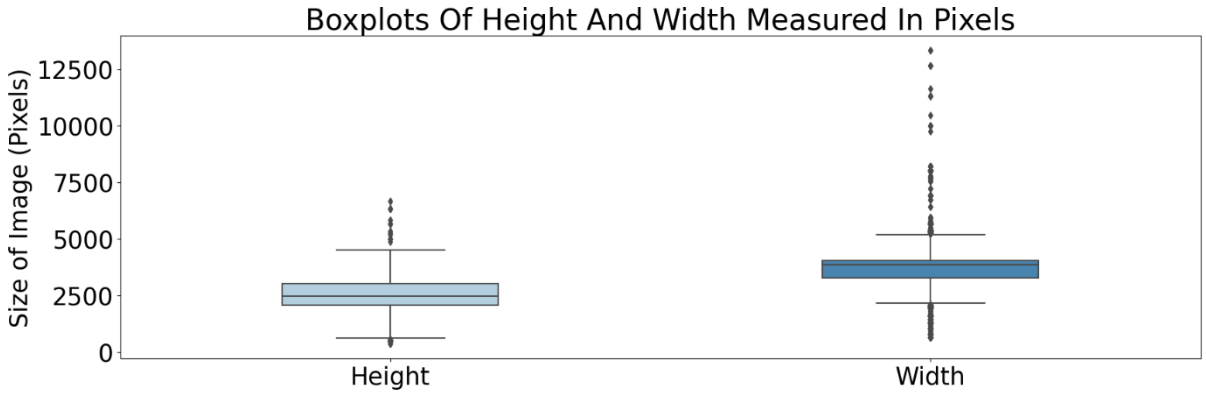


Figure 13: Boxplots of height and width before cropping.

There are currently two main methods for cropping images for object detection applications. The first uses a generalised cropping technique to reduce the image dimensions by a fixed percentage, reducing the standard deviation of image dimensions and drawing the aspect ratio closer to 1:1. This method risks the unintentional cropping of key information from an image, diminishing its effectiveness and purpose. The second method mitigates this risk by drawing a cropping box using the locations of target objects within the image. However, this leads to irregular sized images which will increase the standard deviations of image sizes.

To address this, a hybrid cropping algorithm was created that used bounding boxes to define the initial cropping box areas. To eliminate irregular shapes, 20% of the original image size was restored if the width or height of the image was less than 20% of the original image size. Finally, a 50-pixel tolerance was added to the edges of the images to ensure none of the edges of the objects were omitted from poor labelling. A comparison of these methods is shown in Figure 14, showing the hybrid technique captured both signs whilst retaining a regular shape. The hybrid cropping method was applied to the entire dataset, giving a distribution shown in Figure 15, showing the aspect ratio was closer to 1:1 than in Figure 13. The average width and height after cropping was 931 and 1123 pixels respectively and the standard deviation for these was 620 and 288 pixels respectively. A reason for the standard deviation of width being significantly higher than height is attributed to the fact that signs are typically located on the outer edges of the image. The algorithm was retrained using the same arguments and hyperparameters as in Section 3.3 with results and discussion in Section 4.3.

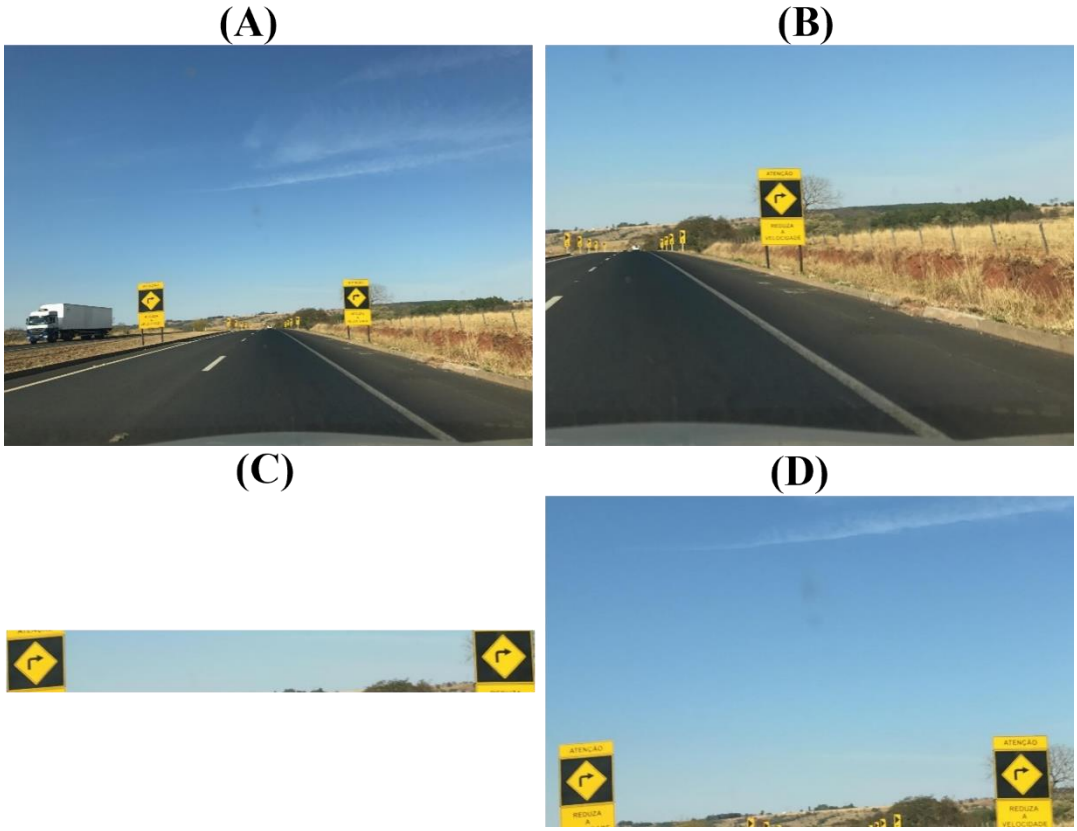


Figure 14: Four images demonstrating the differences in cropping techniques: Original Image (A), Generalised cropping with 40% cut-off, one of the traffic signs was cropped from the image (B), Label Cropping, giving an irregular image shape (C), Hybrid Cropping (D).

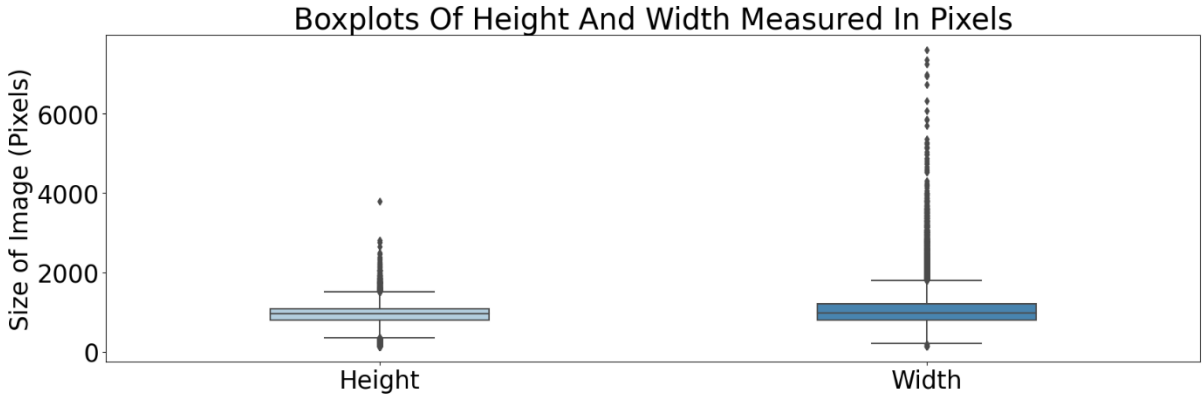


Figure 15: Boxplots of height and width after cropping

3.5 Background Elimination

Inspection of images after cropping showed unimportant information was contained within these images, such as the sky, trees, and roads. To reduce noise, background elimination methods using RGB thresholds were explored. This process involved first; eroding, dilating and Gaussian blurring all images to enhance line boundaries. Secondly, colour thresholds for red, blue, yellow, black, and white traffic signs were defined and tuned by manually evaluating the algorithm's performance on 200 randomly chosen images before training. Figure 16 shows the processes and their effect on an example image, illustrating how thresholding filters out noise. Although some parts of unwanted background remain in Figure 16 (C), the colour thresholds used had to compromise between reducing the clarity of traffic signs in the image and reducing noise in the image. Results from training using RBG thresholding are given in Section 4.4.

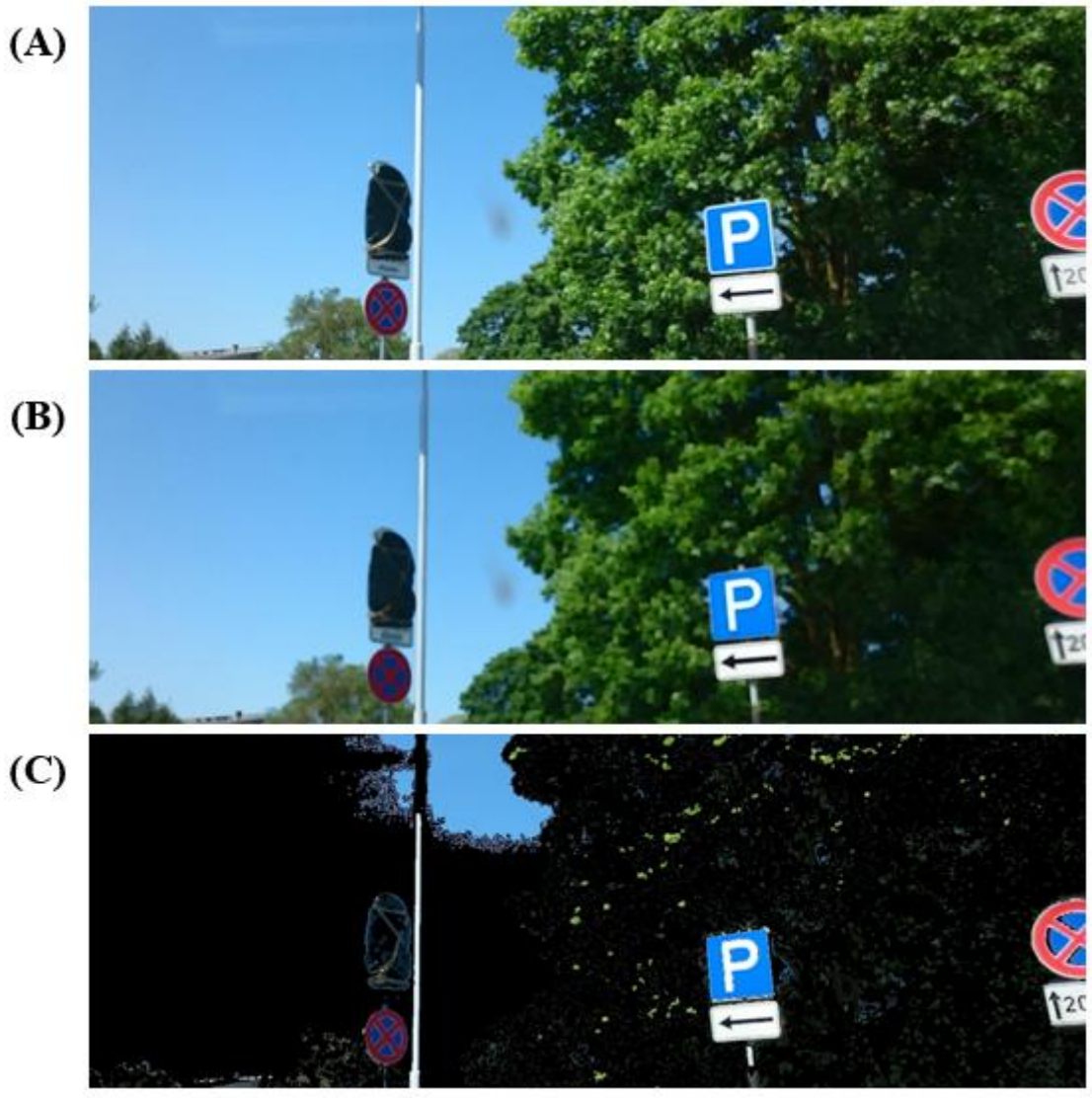


Figure 16: (A) Original Image and (B) Image after the first stage of processing (C) Image after RBG thresholding.

3.6 Hyperparameter Optimisation

Hyperparameters are configurations used to control the training process which can be optimised to maximise accuracy and prevent overfitting. YOLOv5's twenty-seven tuneable hyperparameters were used as the basis of this project's hyperparameter optimisation study, a description of these is given in Appendix 3, with Table 5 listing the most important hyperparameters for this project.

Table 5: Hyperparameters Deemed More Relevant for Traffic Sign Detection. These Parameters Were Changed Appropriately By 20% from Default Values.

Hyperparameter Name	Function Description
Learning rate	Sets the step size after each epoch. A higher learning rate leads to a faster convergence but runs a risk if overshooting from the optimal solution.
Momentum	Sets the amount of epoch history to include for the learning equation to improve gradient decent.
Box loss gain	Places emphasis on the accuracy of the bounding box.
Class Loss gain	Places emphasis on the accuracy of the predicted class
Object Loss gain	Places emphasis on the accuracy of the number of objects in an image
Anchor-multiple threshold	Adjusts how sensitive the model is to variations in the aspect ratio of the predicted bounding boxes. This is important as most traffic signs have a bounding box of 1:1.
Scaling	Resizes all images to a standard size. A larger size requires higher resolution images and more RAM but leads to better results.

Whilst training time was substantially reduced since baseline training, finding the optimal hyperparameters through manual iterative tuning would be too time consuming, estimated to take 270 GPU hours. Therefore, YOLOv5's hyperparameter optimisation algorithm was used which uses a genetic algorithm for optimisation by evaluating the impact each hyperparameter has on accuracy. The process, shown as a flowchart in Figure 17, takes the following steps to seek the optimal combination of hyperparameters:

1. A pretrained detection model is chosen to start the hyperparameter evolution process. The model developed after image cropping described in Section 4.1 was used to initialise the evolution process.
2. The initial hyperparameters are chosen as the first parent set
3. The parent set hyperparameters are randomly changed to form 10 variations of new hyperparameters.
4. The model is then trained for 10 epochs (a generation), each with a different set from the population.
5. The Hyperparameters responsible for the 5 best epochs of the generation are taken, averaged, and chosen for the next generation's parent set.
6. This process is repeated with the new parent set until convergence of the mAP score was reached.

The results from hyperparameter optimisation can be found in Section 4.5.

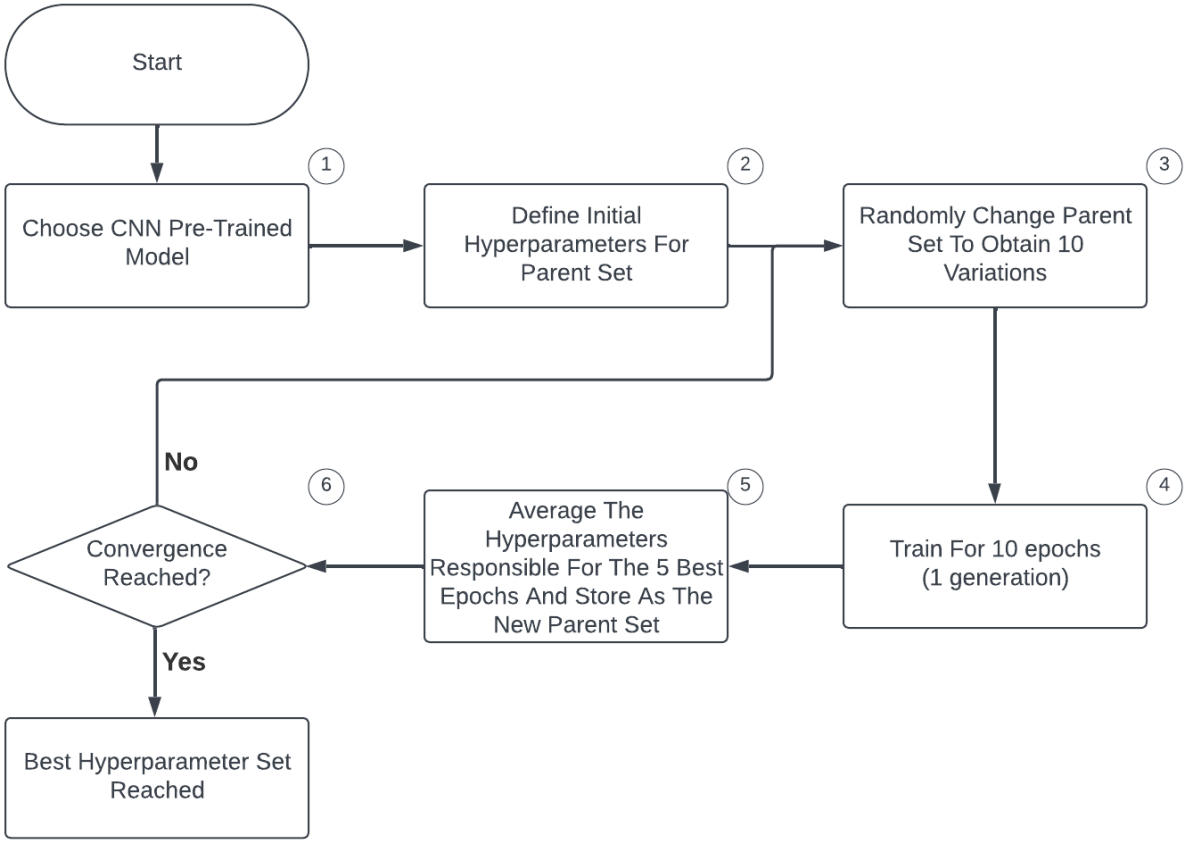


Figure 17: A flowchart of the hyperparameter evolution process

4.0 Results and Discussion

4.1 Baseline Results

Before results could be analysed, the F1 score had to be calculated, to compare the training and testing accuracies. This was calculated by finding the harmonic mean of the precision and recall, metrics determined during validation which represent the accuracy of detected objects and the percentage of actual objects correctly identified respectively.

Baseline training was run for 1400 epochs, lasting 168 hours, before it was concluded the accuracy of the model would not increase, although convergence had not been reached. Figure 18 shows the mAP and F1 scores had maximum values of 23.2% and 0.32 respectively.

The loss function training curves, shown in Appendix 4, were minimised after the first runtime session with values for box, object and class loss obtaining minimum values of 0.0617, 0.0161 and 0.02094 respectively. However, due to experiencing timeouts every 500 epochs, these values progressively worsened, signifying the dataset must be simplified to prevent this from occurring again.

Appendix 4 also shows the final precision higher than recall, indicating the model was more likely to miss a traffic sign than misclassify it, a tendency indicative of a bias in the dataset, which occurs when some classes are under-represented and will negatively impact accuracy.

With reference to criterion F in Section 2.2.2, the goal of state-of-the-art performance is met when the mAP is greater than 81.1% and all loss functions are below 0.02, which in comparison to baseline results, show the model is far from this objective.

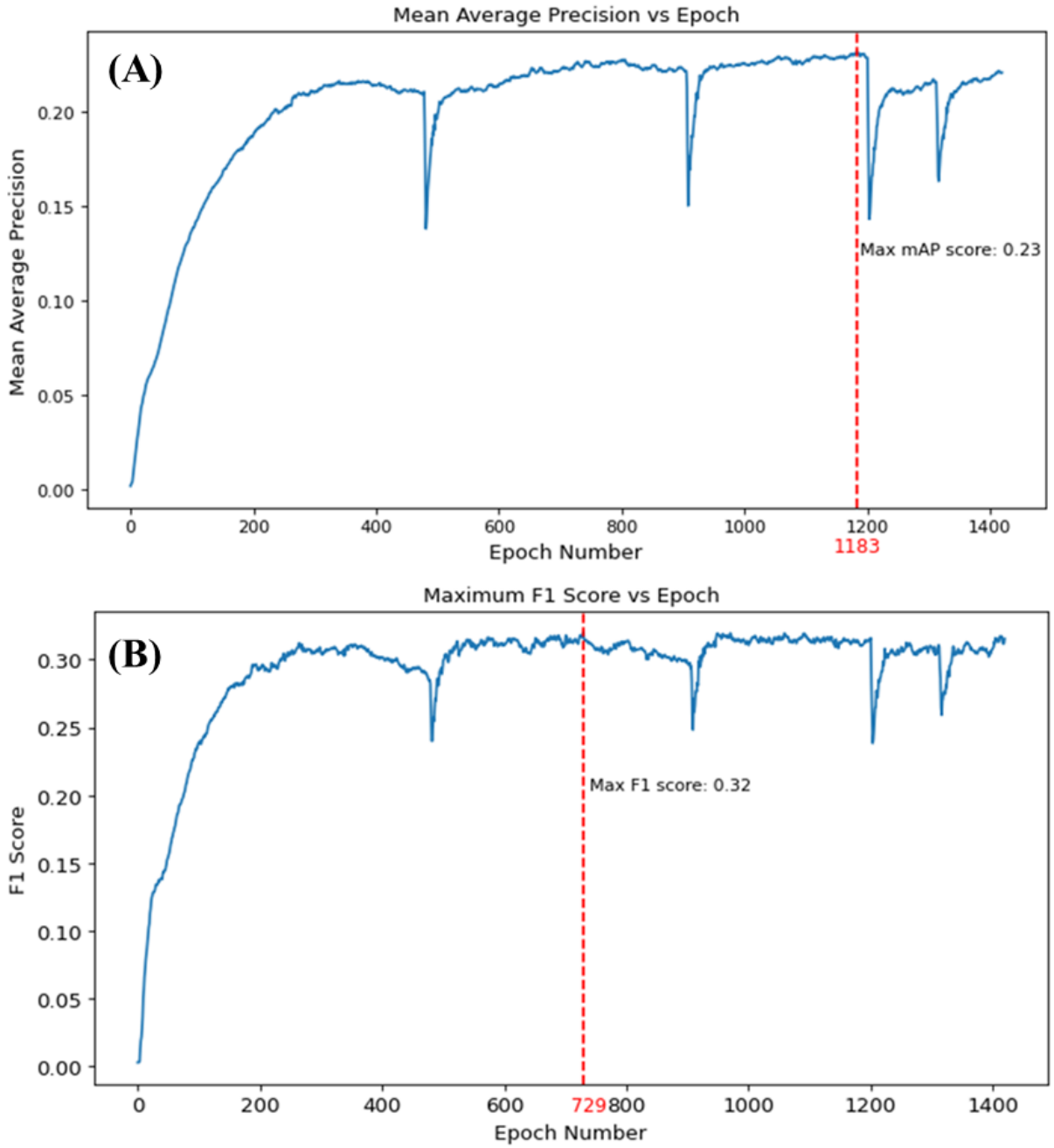


Figure 18: Graphs of mAP (A) and F1 score (B) against epoch number for baseline training. The red line shows the epoch with the optimal value. The sharp downwards peaks result from resuming training after a runtime timeout in Google Colaboratory.

4.2 Class Reduction

The results after reducing the dataset's classes are shown in Figure 19, with the mAP and F1 scores reaching maximum values of 34.07% and 0.412 respectively. As predicted, class reduction significantly reduced the training time, with model convergence after 147 epochs, after 70 hours (~3 days), a 58.33% decrease from baseline training. Appendix 5 shows the loss functions, with minimum values for box, object and class losses of 0.0447, 0.0159 and 0.00805 respectively, showing in reference to criterion F in section 2.2.2, the model had not achieved state-of-the-art performance in either the mAP or loss functions. Hence, further pre-processing was required before the model could be re-trained and deployed.

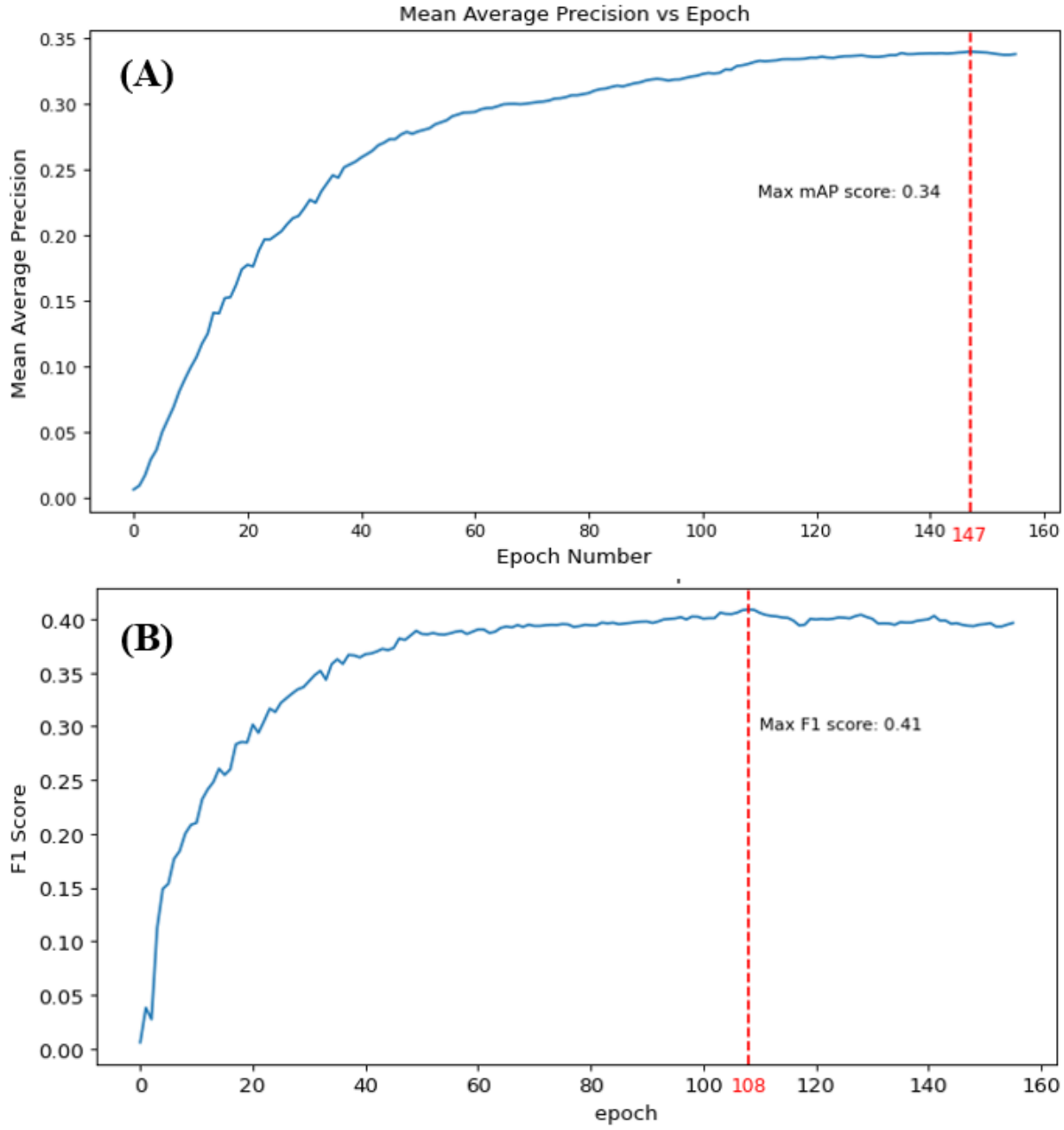


Figure 19: Graphs of mAP (A) and F1 score (B) against epoch number after Class Reduction. The red line shows the epoch with the optimal value.

4.3 Image Cropping

The Hybrid Cropping method created specifically for this project had a significant positive impact on the training accuracy of the model. The model reached convergence in 164 epochs, after 22.5 hours, 32% quicker than Class Reduction, reaching training mAP and F1 scores of 76.1% and 0.75 respectively, shown in Figure 20. This surpasses some models of other research reviewed in Section 2, including ones trained on global datasets [39]. Appendix 6 shows the loss functions took minimum values for box, object and class losses of 0.01756, 0.00567 and 0.00531 respectively, meaning these were all within the state-of-the-art threshold defined in criterion F in Section 2.2.2. However, the model had not surpassed the state-of-the-art threshold for mAP, meaning further optimisation was required.

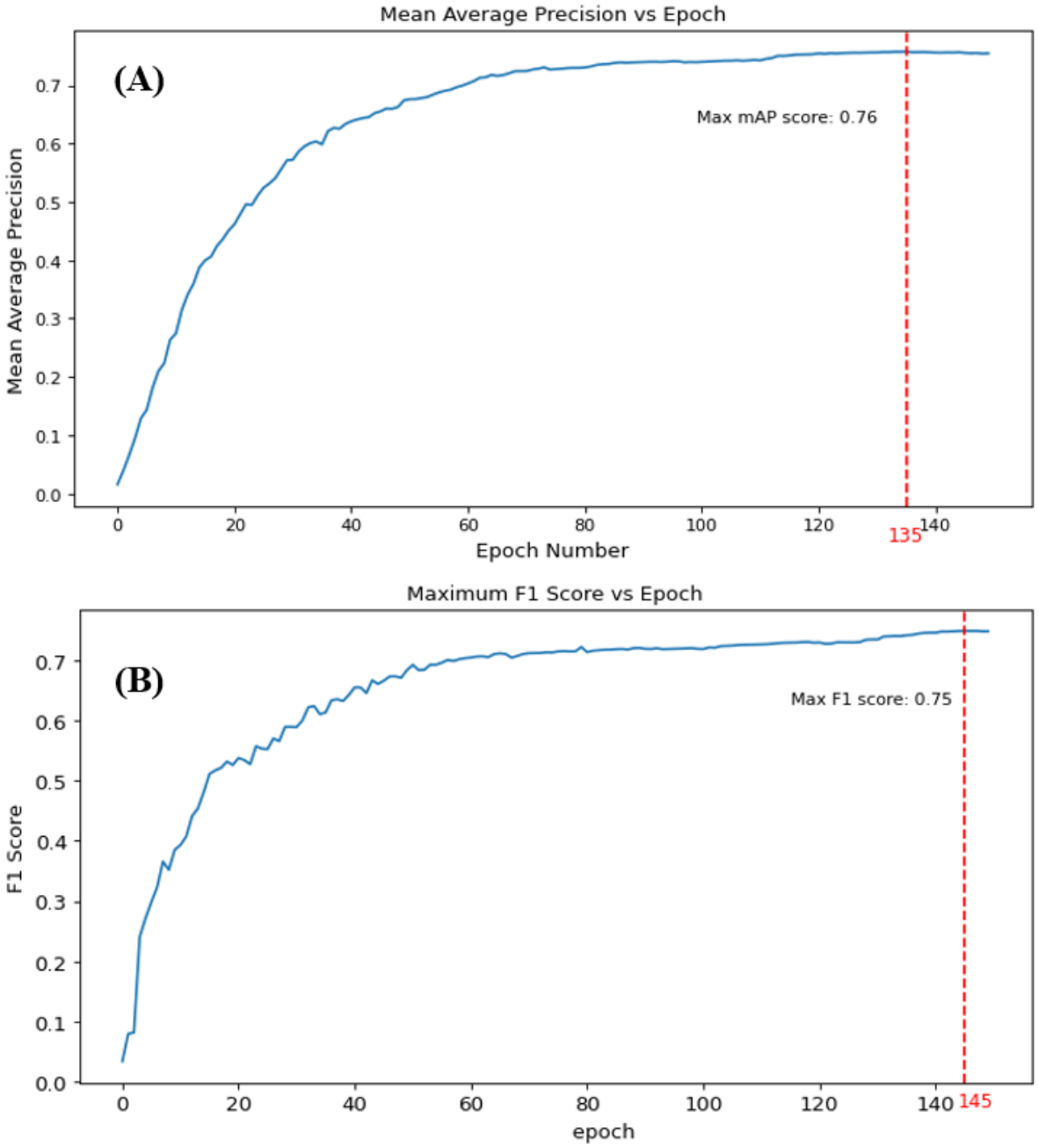


Figure 20: Graphs of mAP (A) and F1 score (B) against epoch number for training after Image Cropping. The red line shows the epoch with the optimal value.

4.4 Background Elimination

Training using background elimination gave a maximum mAP and F1 score of 66.0% and 0.67 respectively, as shown in Figure 21, showing background elimination using RGB thresholding was not as effective as expected, instead negatively impacting the model's accuracy.

Counterintuatively, the loss functions shown in Appendix 7, with values for box, object and class loss of 0.0143, 0.00449 and 0.00319 respectively, all improved with background elimination. This shows that the model overfitted its data, as the accuracy, measured by the mAP, decreased despite this improvement.

The degredation in accuracy could be attributed to the variable lighting conditions of road images that cannot easily be captured through RGB thresholding, meaning some traffic signs may have been partially removed by the background elimination process. Therefore, going forward, RGB thresholding will not be used. However, the first stage of the process, which sharpened line boundaries of the image, was still used for preprocessing.

For future work, exploring methods previously discussed such as colour segmentation (Section 2.1.1) or edge detection (Section 2.1.2) could be used over RGB thresholding as these methods use more advanced techniques to reduce image noise.

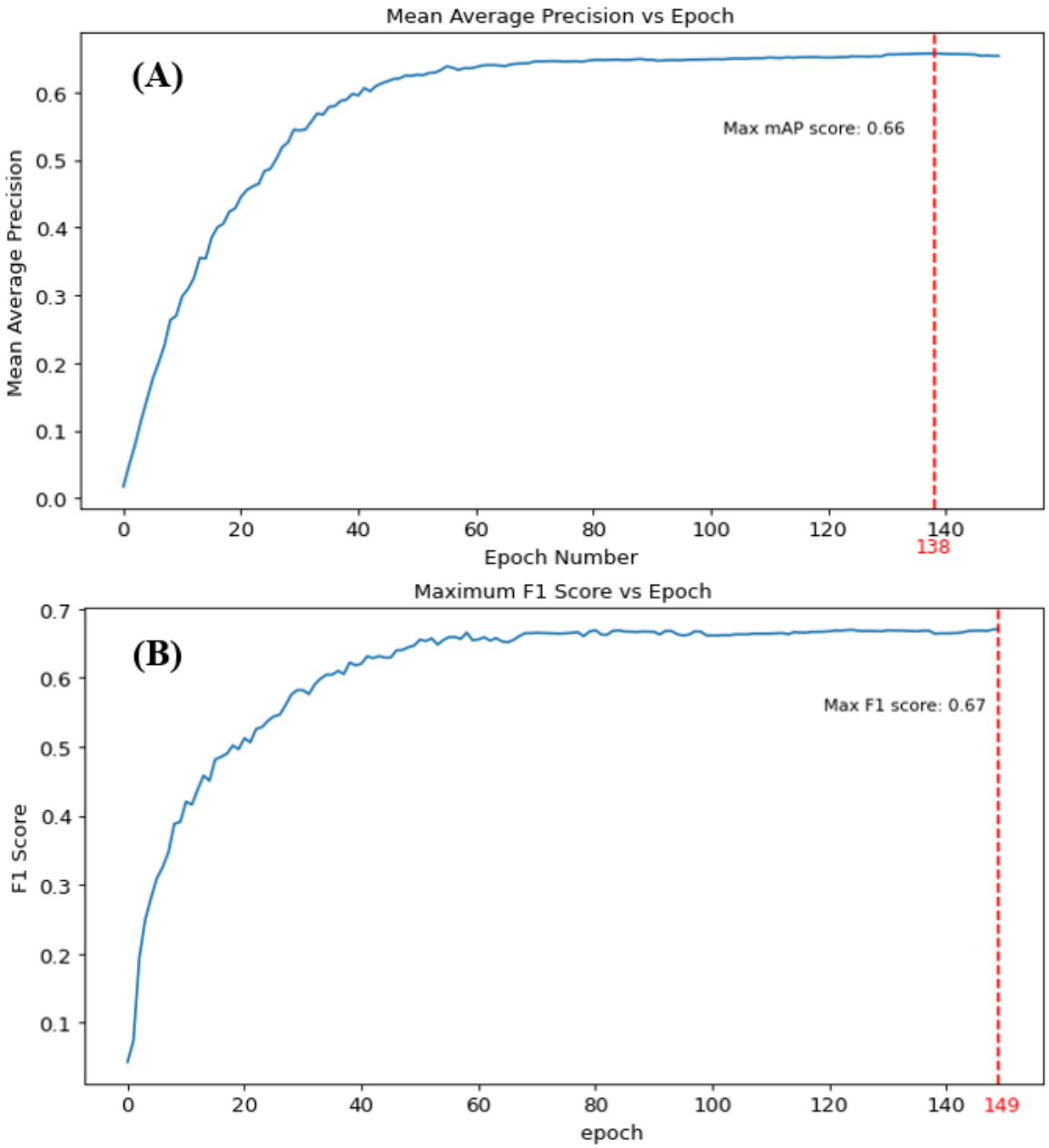


Figure 21: Graphs of mAP (A) and F1 Score (B) against epoch number after Background Elimination. The red line shows the epoch with the optimal value.

4.5 Hyperparameter Optimisation

The evolution algorithm was run for seventy-three generations (730 epochs), lasting 75 hours. Figure 22 shows the F1 and mAP training curves produced. For initial generations, the adjustment of hyperparameters was significant, as the optimization algorithm explored a wide range of different hyperparameters. However, as the accuracy began to stabilize, the variations in hyperparameters between each generation became increasingly minimal, meaning optimisation had converged to an optimal set of hyperparameters. The overall evolution process improved the model's mAP by 7.28%, to 83.43%, meeting the accuracy requirements for criterion F defined in Section 2.2.2.

Appendix 8 illustrates the movements of each hyperparameter from its initial value, with the absolute changes shown in Figure 23, showing the "Anchor Threshold" parameter had the largest influence on accuracy improvement.

Unlike training, the hyperparameter optimisation algorithm seeks to maximise the accuracy and not minimise the loss functions. As a result, the loss functions, shown in Appendix 9, had high values for box, object and class loss of 0.0759, 0.00252 and 0.00159 respectively, showing the state-of-the-art performance had not been met. Therefore, the model had to be trained from scratch, using the hyperparameters found from optimisation, to minimise this loss function.

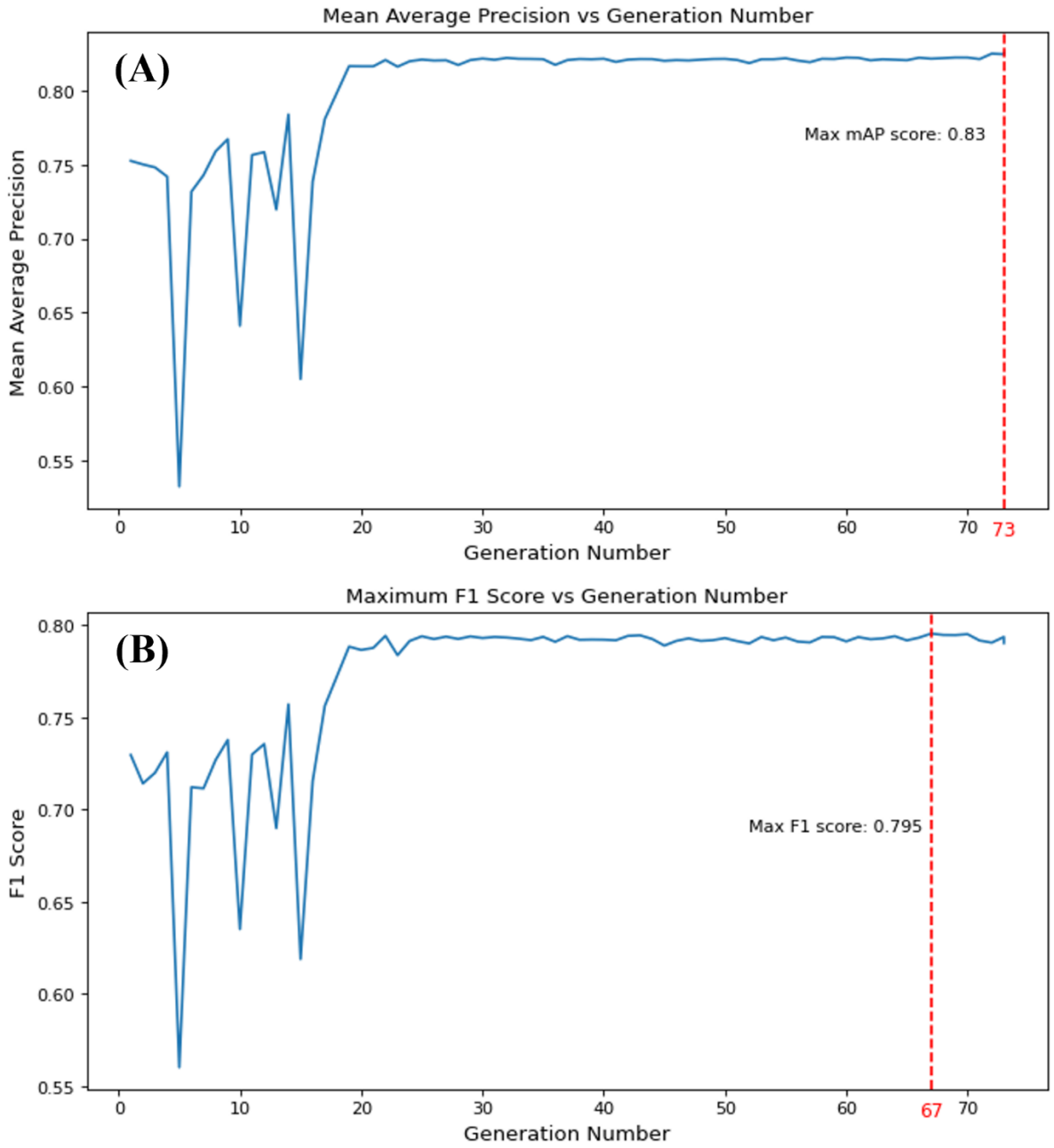


Figure 22: Graphs of mAP and F1 Score against Generation Number for hyperparameter evolution.

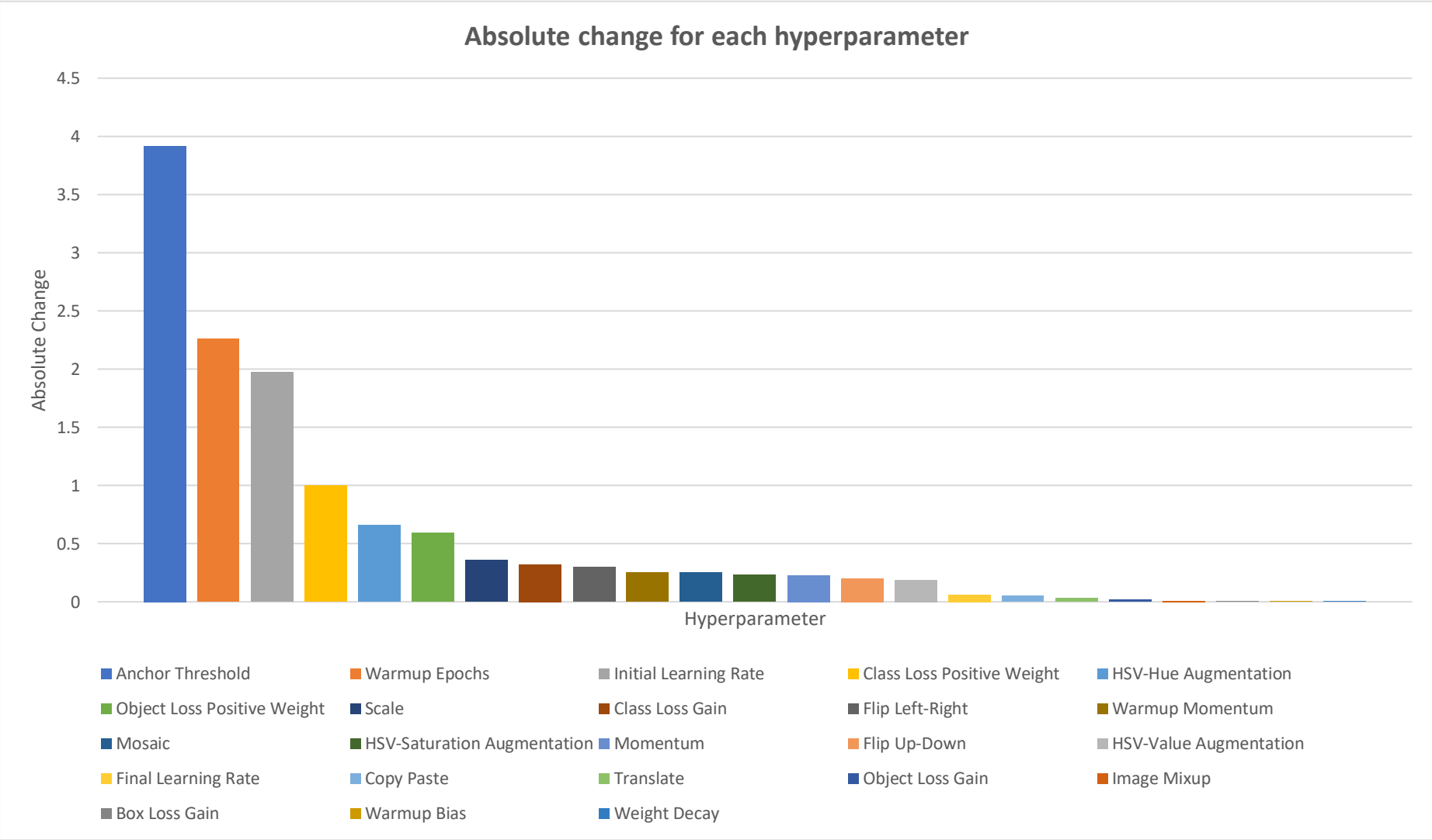


Figure 23: A bar chart illustrating the change of hyperparameters from the start of evolution to the end

4.6 Final Training

A final detection model was trained from scratch using the optimal hyperparameters found through the evolution algorithm. Figure 24 shows the final mAP and F1 scores of the model were 84.4% and 0.81 respectively, with box, object and class losses of 0.0121, 0.00387, 0.00251 respectively (shown in Appendix 10), showing the model can be considered state-of-the-art as defined in criterion F in Section 2.2.2. Appendix 10 shows the gap between the precision and recall has decreased compared to results shown in Section 4.1, indicating the reduced bias in the dataset and training process. Finally, convergence was reached after 82 epochs, lasting 12.5 hours, meaning the overall training process was reduced by 155.5 hours.

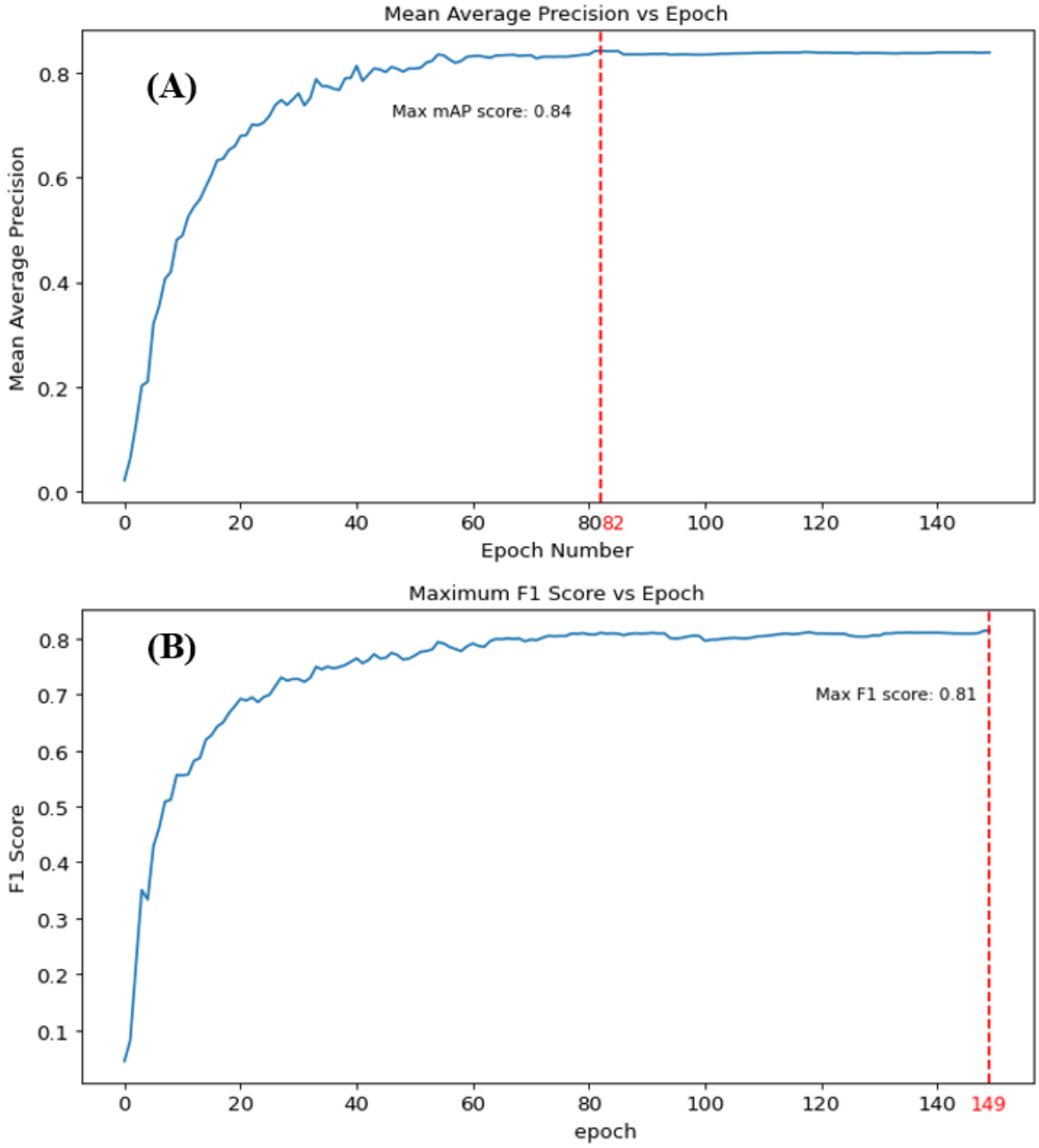


Figure 24: mAP (A) and F1 Score (B) training curves for final training. The epoch with the optimal value is shown by a red line.

5.0 Model Testing

Model testing is the process of verifying training results by testing the detection model against independent images. This was carried out to meet criteria A and C in section 2.2.2.

5.1 Robustness Testing

To determine the model's robustness to external influencing factors, 3 main tests were carried out under this category.

5.1.1 Obstruction Test

Obstruction testing was carried out to evaluate the detection model's accuracy against traffic signs partially obstructed by foliage, dirt and other vehicles. With reference to Figure 25, the model could detect objects with 50% or less obstruction. This success rate surpasses shape recognition algorithms which are not robust to obstructions of this extent.

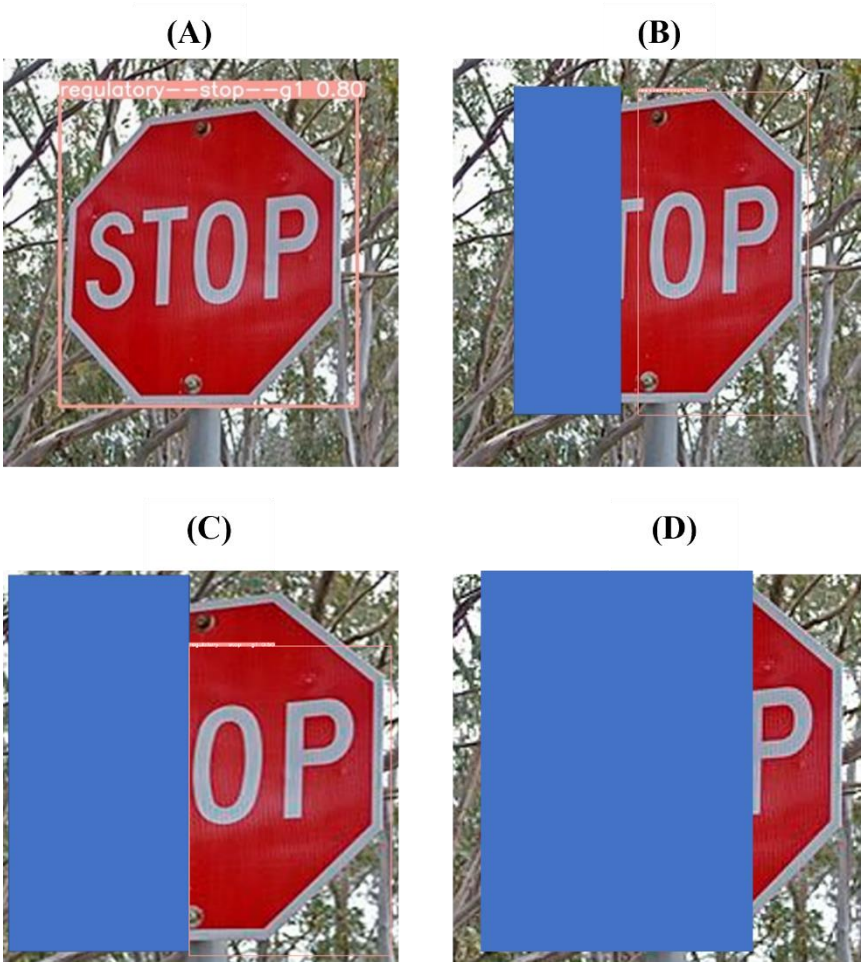


Figure 25: Predictions as the traffic sign is incrementally obstructed. (A) The algorithm identifies the stop sign with 80% confidence. (B) The sign is obstructed by 33%, the algorithm identifies the sign with 72% confidence. (C) The sign is obstructed by 50%, the algorithm identifies the sign with 50% confidence. (D) The sign is obstructed by 75%, the algorithm fails.

5.1.2 Brightness Test

Brightness testing was carried out to evaluate the detection model's accuracy against traffic signs in low-light settings, an area which other CV methods such as colour detection struggle in. With reference to Figure 26, the model achieved successful detections when brightness was lowered by 90% of natural light. Unexpectedly, the accuracy of detections increased as brightness decreased. This led to the conclusion that reducing the brightness helped decrease noise and background interference in the image, making it easier for the model to identify the traffic signs. This hypothesis was verified when lowering the brightness for other images, with the same trend shown. Hence, the model has displayed resilience to brightness changes, making it suited for implementation on roads where headlights are employed.

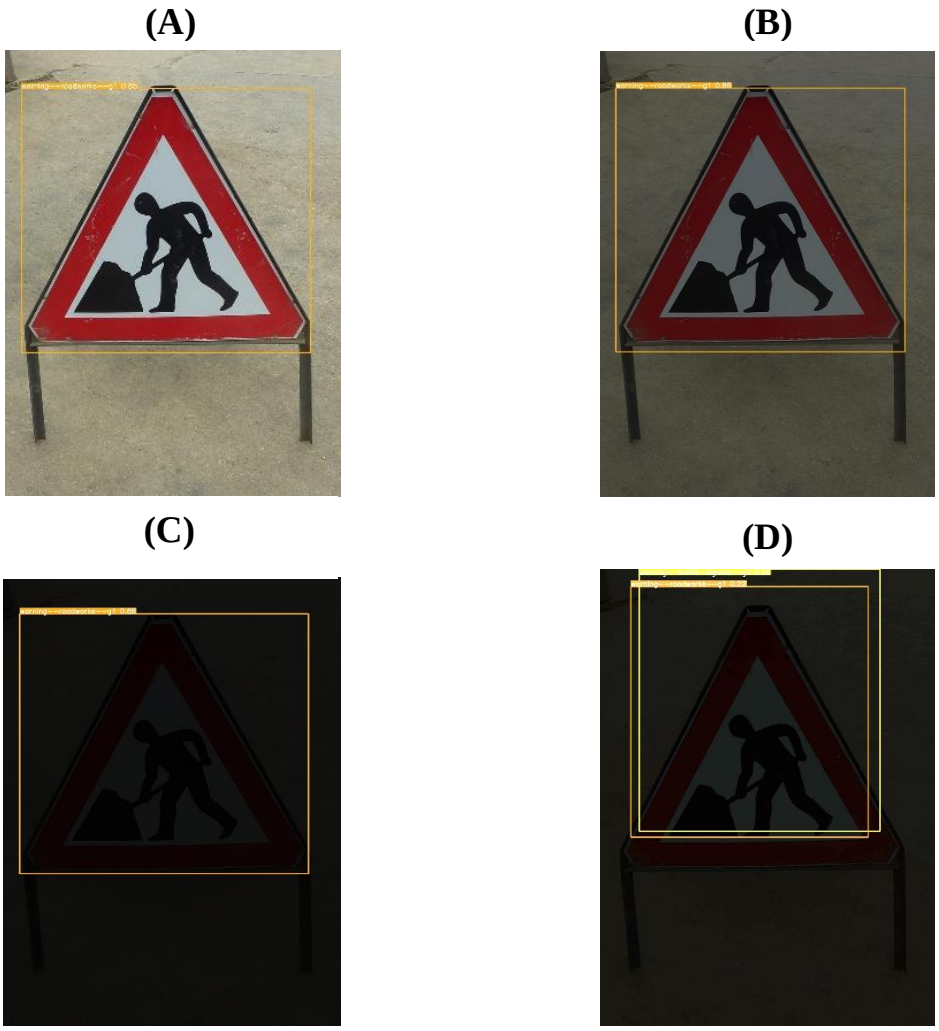


Figure 26: Comparison of sign identification performance at different brightness levels. (A) Sign identified as road-works with 80% confidence. (B) 50% brightness reduction; sign identified correctly with 86% confidence. (C) 90% brightness reduction; sign identified with 88% confidence. (D) 95% brightness reduction (brightness increased for visual purposes); sign misidentified as a traffic sign (20% confidence) and a roadworks sign (22% confidence).

5.1.3 Weather Test

Images of snow, rain and fog were tested to determine the model's performance in different weather conditions, with results shown in Figures 27 to 29. Results shown that the model made correct predictions in settings of fog, snow and rain. With reference to Figure 28, the model's confidence decreased significantly, which is attributed to the increased noise, making the sign blend in more with its environment. Nevertheless, it to acknowledge that correct predictions were always given, which is important as incorrect predictions could pose a risk to the driver. However, considering how snow significantly impacted the model's performance, future work could focus on adding images in snowy settings to the dataset to increase confidence.

(A) **(B)**

Figure 27: Performance of model in foggy conditions. Original image (A) and image with prediction (B) are shown. The model correctly identified the Stop sign with 55% confidence.

Figure 28: Correct prediction on a stop sign with a moderate amount of snow with 26% confidence.

(A) **(B)**

Figure 29: Performance of model in different levels of rain. The model performed well in light rain (A), correctly predicting the sign with a 90% confidence. The model correctly predicted the sign in heavy rain with a blurry image at lower confidence 23% (B).

5.2 Large Dataset Testing

To satisfy criterion C in Section 2.2.2, the model required testing against a large dataset independent from its validation and training datasets. It was found that the testing set supplied in the MTSD's fully annotated set did not have labels and therefore could not be used for testing as the model's predictions had to be verified. Furthermore, there were no alternative open-source datasets that had global distribution. Consequently, this project had to use the partially annotated set from the MTSD, which contained a higher prevalence of deficiencies than the training data, as the labelled images could be used to evaluate the model's prediction accuracy. The partially annotated dataset exhibited a high prevalence of deficiencies meaning the quality is not the same as the training dataset. In hindsight, testing with verified data is preferred as this would have ensured that the quality of the data is at least comparable to the training dataset.

To test the model on the large dataset, 13,350 images from the partially annotated set were separated for testing purposes. To increase the validity of the test, the same class reduction, image cropping and image processing steps applied to the training data were applied to the testing set, reducing the final set to 7,783 images.

The model was tested on each image and a corresponding predicted annotation was given to the image if any traffic signs were found, which was compared to actual annotations. A full confusion matrix from these results is shown in Appendix 11, with a binary confusion matrix shown in Figure 30.

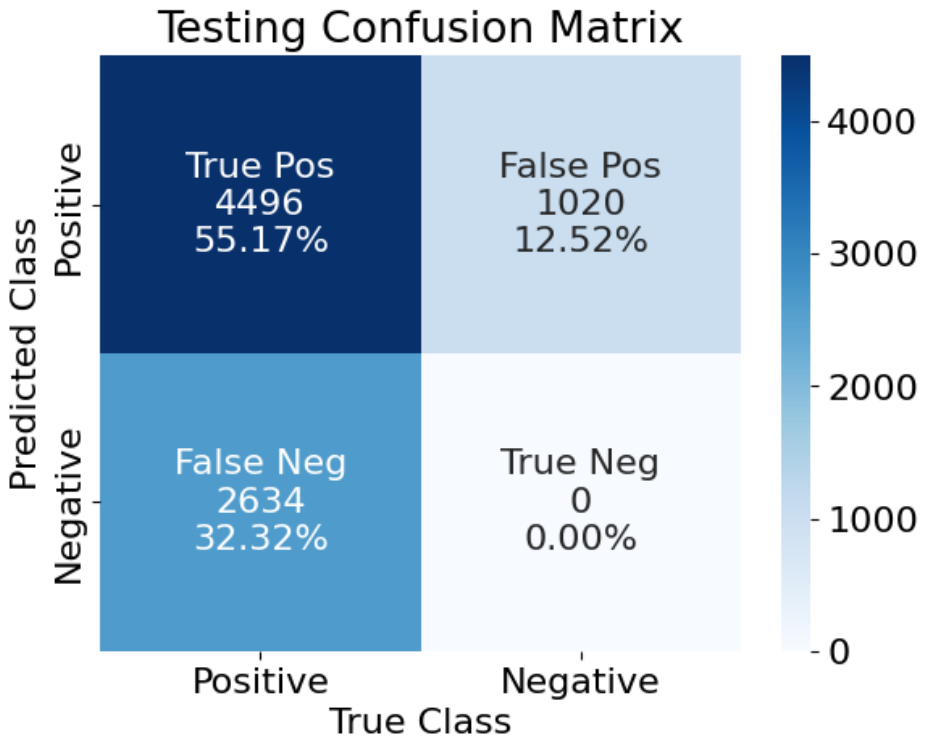


Figure 30: Binary Confusion Matrix for Large Dataset Testing. The non-percentage numbers within the matrix represent the number of instances of traffic signs.

To give a direct comparison between the training and testing accuracies of training and testing, the testing F1 score had to be calculated using the following equation:

$$\text{F1 Score} = \frac{(\text{precision} \times \text{recall})}{\text{precision} + \text{recall}} = \frac{\text{True Positives}}{\text{True Positives} + \frac{1}{2}(\text{False Positives} + \text{False Negatives})}$$

$$\text{F1 Score} = \frac{4496}{4496 + \frac{1}{2}(1020 + 2634)} = \mathbf{0.711} \quad (1)$$

Comparing this F1 score from final training (0.81), there is a decrease in performance. As the difference is only 12.2%, this difference is acceptable and can be attributed to the final slightly overfitting the data.

6.0 Deployment of Prototype

The final stage of development was integrating the object detection model with a narration algorithm to create a working prototype. In addition to narrating signs, segmenting each frame and storing predictions, the prototype carried out the following steps to minimise distractions and false predictions:

Step 1. Remove predictions below a confidence threshold of 80%, filtering out incorrect predictions and irrelevant signs to the driver.

Step 2. If signs are found, the one with the highest confidence score is narrated first. For future work, signs could be weighted to prioritise against levels of cautions, hazards and warnings depicted in these signs. For example, a warning sign will have higher weighting than a regulatory sign and thus will be narrated first.

Step 3. The algorithm does not narrate signs detected more than once in a 10 second interval to avoid narrating the same sign. In a practical setting, this will require calculations to optimise near real time responses based on the distance of the sign and speed of the vehicle.

6.1 Assessment and Discussion for Deployment

As the final algorithm was constrained to a laptop, final testing was limited to objects captured by laptop's camera, thus dashcam testing was not possible. Due to specifications of the laptop, the average framerate was 2 Frames-Per-Second rate (FPS).

The prototype can make faster detections with GPU support due to its advantages in parallel processing, however, the final algorithm couldn't use GPU support using cloud computing applications as webcam usage is not supported. To estimate the framerate with GPU support, a working assumption that the time taken for the detection model to make predictions on each frame is the main time component between frames. With this assumption, the time taken for the detection model is calculated, averaged and converted into a hypothetical framerate. Using Google Colaboratory's GPU support, a random image from the testing dataset was detected with the model, taking 64.4 milliseconds to detect. The hypothetical framerate can be calculated as:

$$FPS = \frac{1000}{64.4} = 15.53\ FPS$$

15.53 FPS is comparable to low-end security cameras. However, car dashboard cameras record 30 FPS minimum to reduce motion blur when capturing fast-moving scenes. Thus the prototype can be adapted for integration into a dashboard camera by processing every second frame front the video.

A video supplementing this report ("*MECH0020_VideoTest_20072669.mp4*") shows the testing of the algorithm on 5 different signs. Relative sizing shown in Figure 31 for this test. The test showed that all narrations were correct and there were not repeated predictions.

Figure 31: A Screenshot during video testing. The entire image was input into the detection model and the correct detection was given. The sign occupied 0.904% of the total image area, which, upon comparison of testing images, is a reasonable and realistic size.

7.0 Conclusion

This project's aimed to deliver a low-cost notification system, providing audio alerts of oncoming traffic signs, to improve road safety. This was achieved by delivering a machine learning algorithm that recognises, interprets and narrates the meaning of traffic signs, improving driver awareness and was demonstrated in a working prototype. This project successfully developed a state-of-the-art Deep Learning recognition model trained on a globally distributed dataset containing traffic sign images, capable of detecting 65% of warning or regulatory classed traffic signs. The remaining 35% were intentionally omitted from the training dataset because their low occurrences do not provide sufficient data to train a CNN.

This report also lays out a methodology to obtain a model capable of reaching state-of-the-art performance using bespoke image processing techniques that can be applied to other fields, such as a hybrid cropping process.

The Convolutional Neural Network model produced from this project gave a mean Average Precision of 84.1% and an F1 score of 0.81, impressive as training was conducted using a more diverse, complex and challenging dataset with more classes to detect than those used in other research.

This report addresses drawbacks of prior research by performing and demonstrating the robustness of the model in extreme scenarios and testing the performance of the model on 8150 images, obtaining a high testing F1 score of 0.711, demonstrating sufficient performance for commercial use.

The final narration algorithm was tested, to ensure the model's performance had not deteriorated from deployment, with safety considerations implemented to ensure the algorithm does not unnecessarily distract the driver such as setting confidence thresholds for narration.

Before any consideration of commercialising the final detection algorithm, comparing the testing outcomes of the project against results from road tests under different weather conditions, locations and speeds, is needed.

This report has laid the foundations to develop a product that will not only increase drivers' confidence on the road, but potentially save lives.

Appendix 1. Initial Dataset Distribution

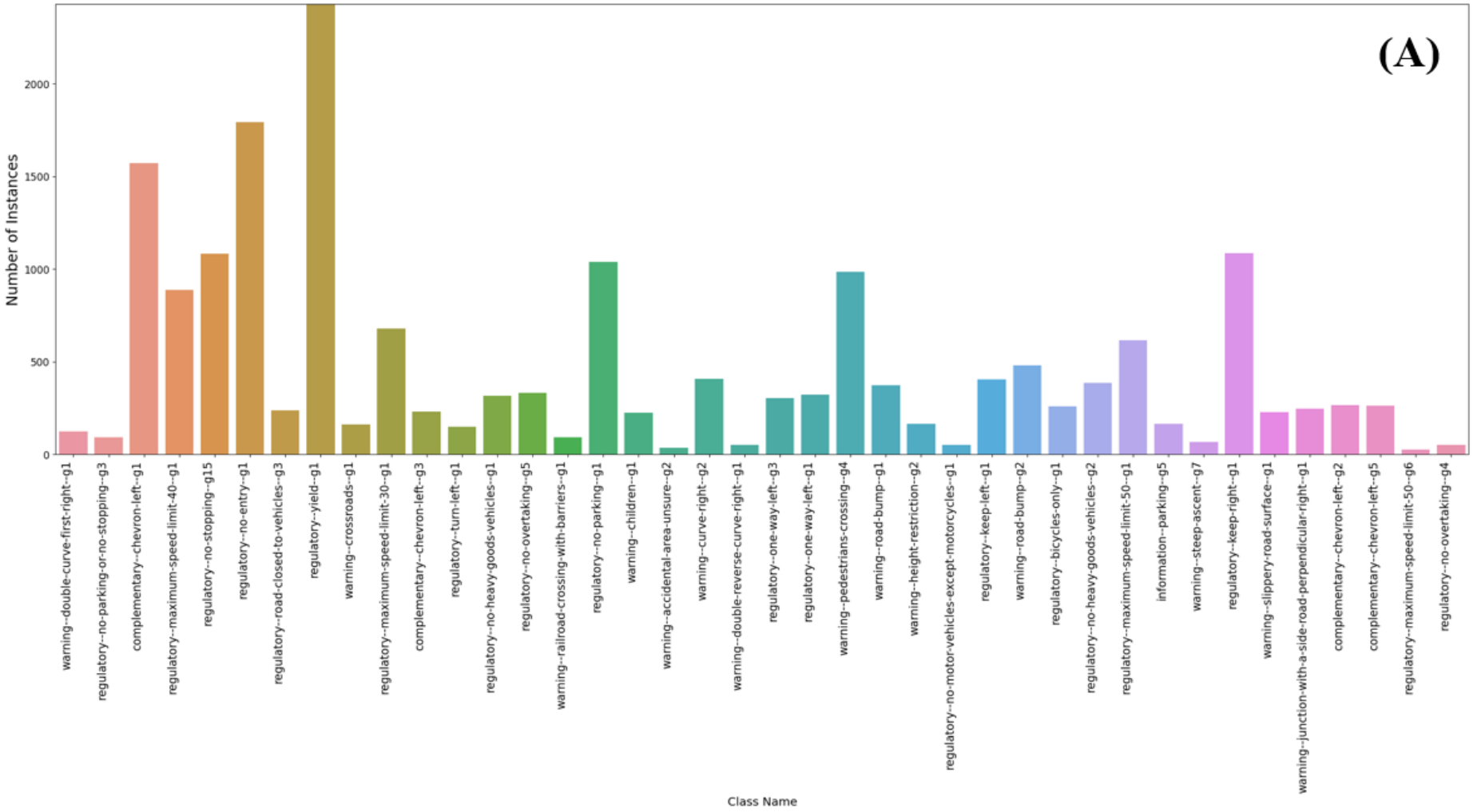


Figure A1.A: Bar charts of number of instances per class for the MTSD, without "other" class, for the first 40 classes in the dataset. The class names and subplots are arranged in class numeric order.

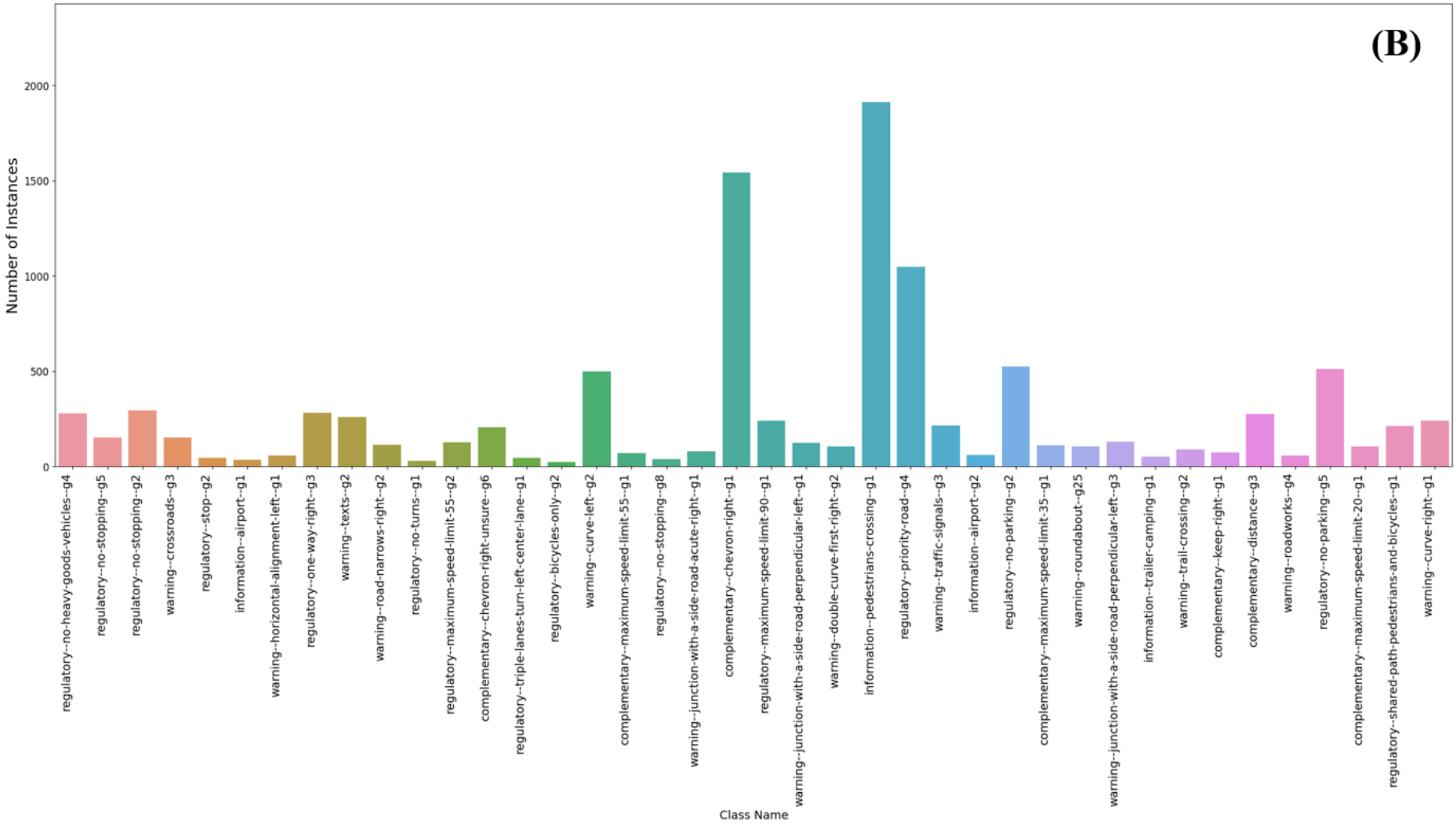


Figure A1.B: Bar charts of number of instances per class for the MTSD, without “other” class, for classes 41-80 in the dataset. The class names and subplots are arranged in class numeric order.

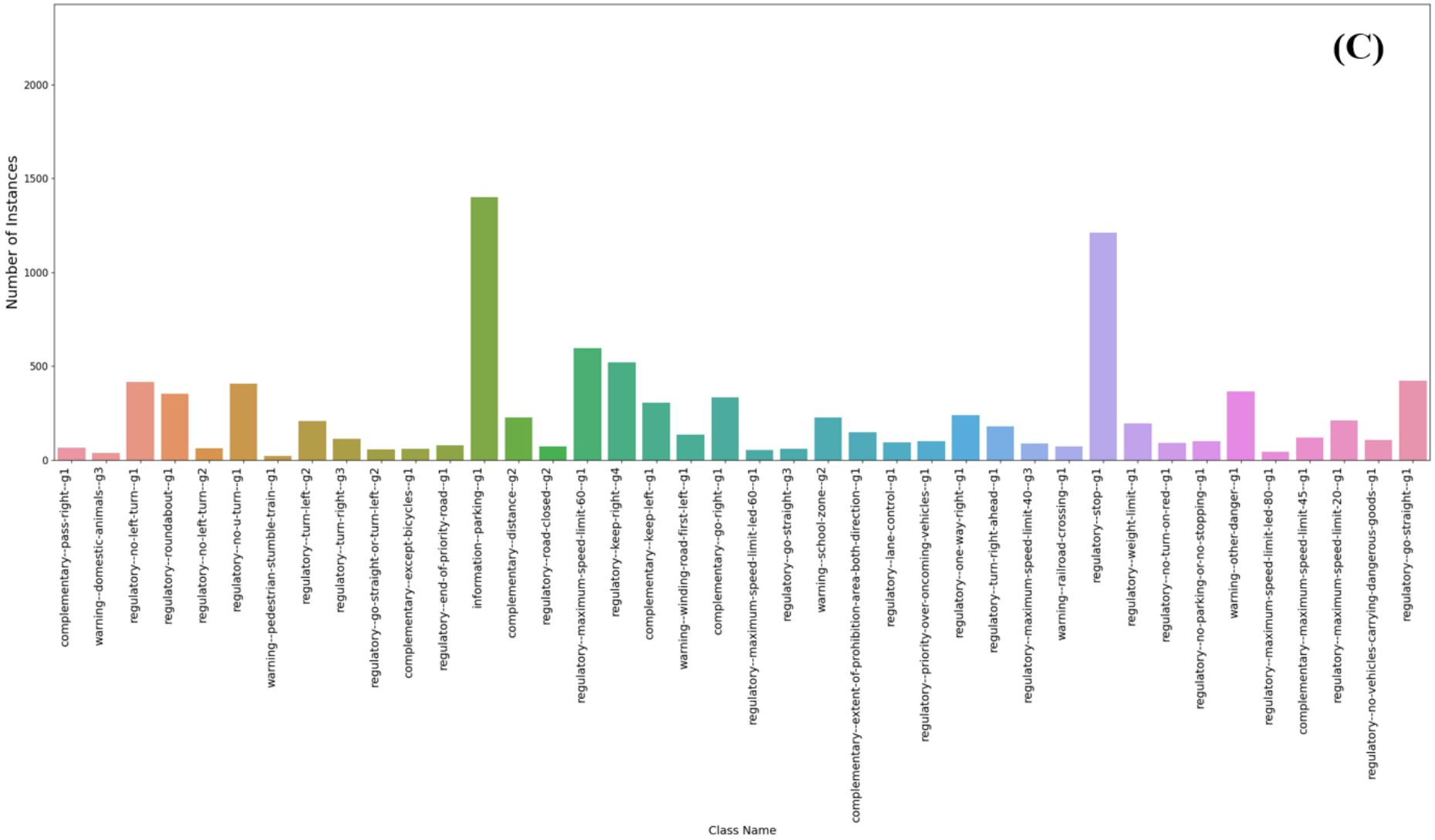


Figure A1.C: Bar charts of number of instances per class for the MTSD, without "other" class, for classes 81-120 in the dataset. The class names and subplots are arranged in class numeric order.

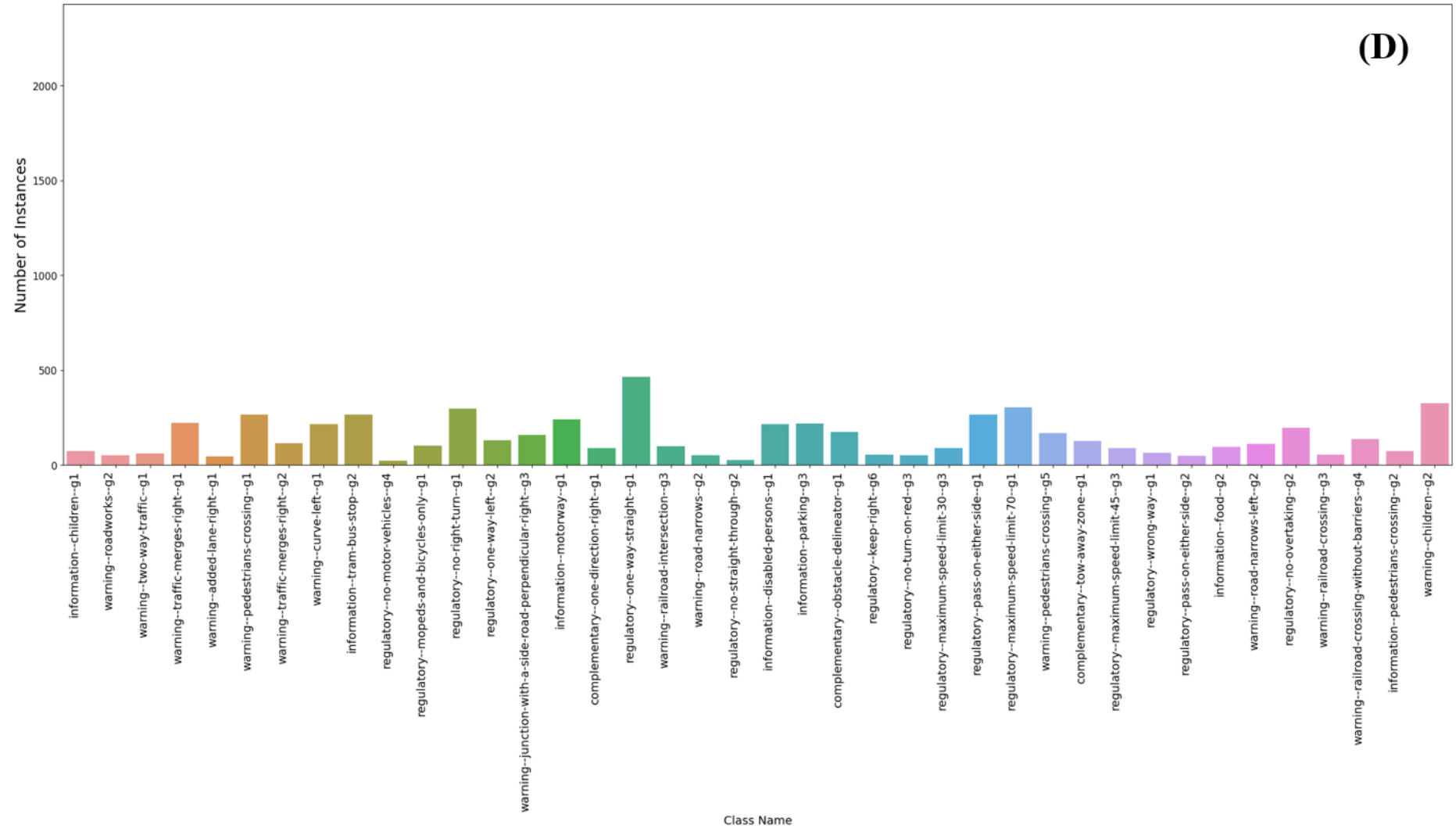


Figure A1.D: Bar charts of number of instances per class for the MTSD, without "other" class, for classes 121-160 in the dataset. The class names and subplots are arranged in class numeric order.

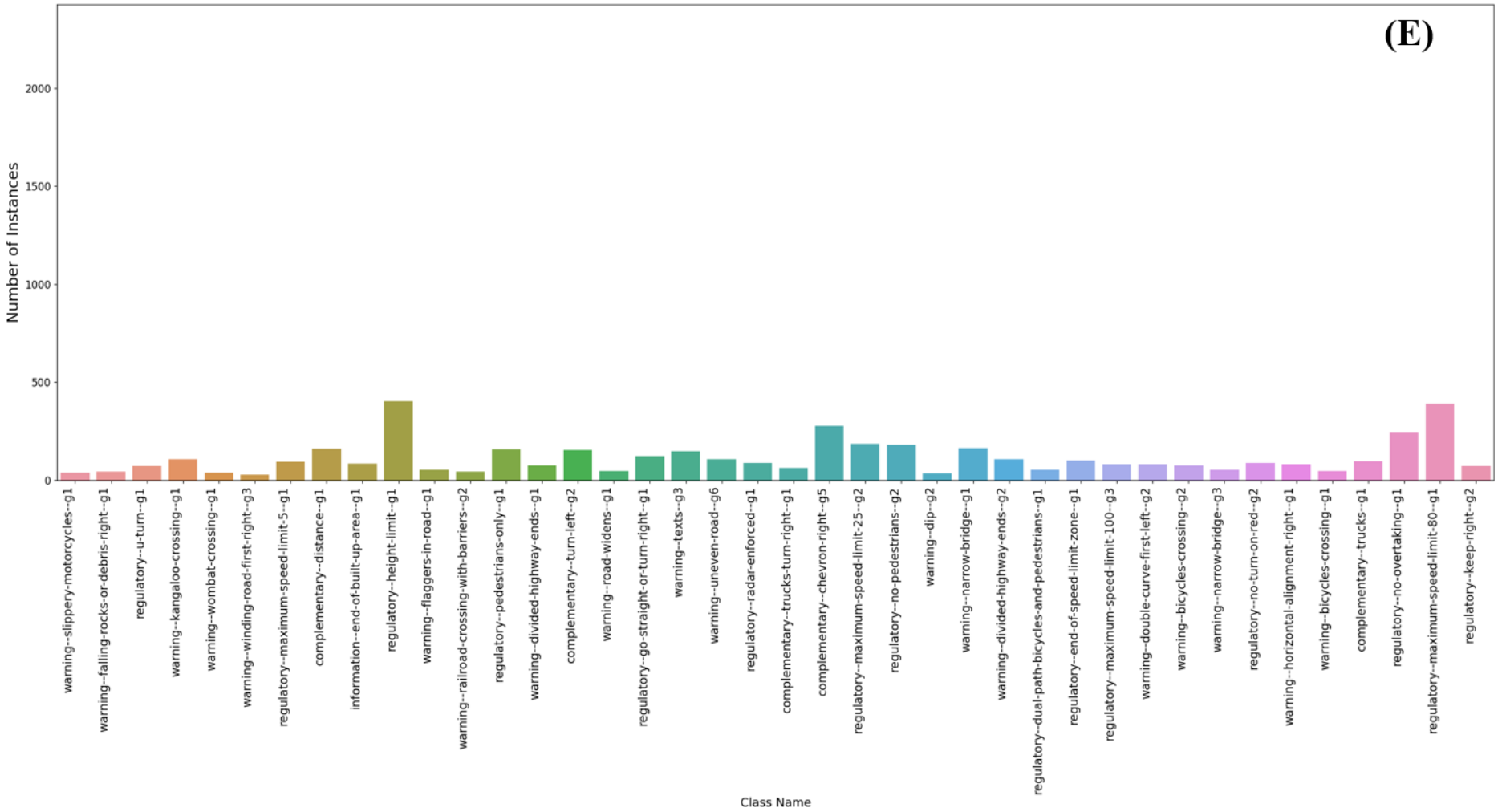


Figure A1.E: Bar charts of number of instances per class for the MTSD, without "other" class, for classes 161-200 in the dataset. The class names and subplots are arranged in class numeric order.

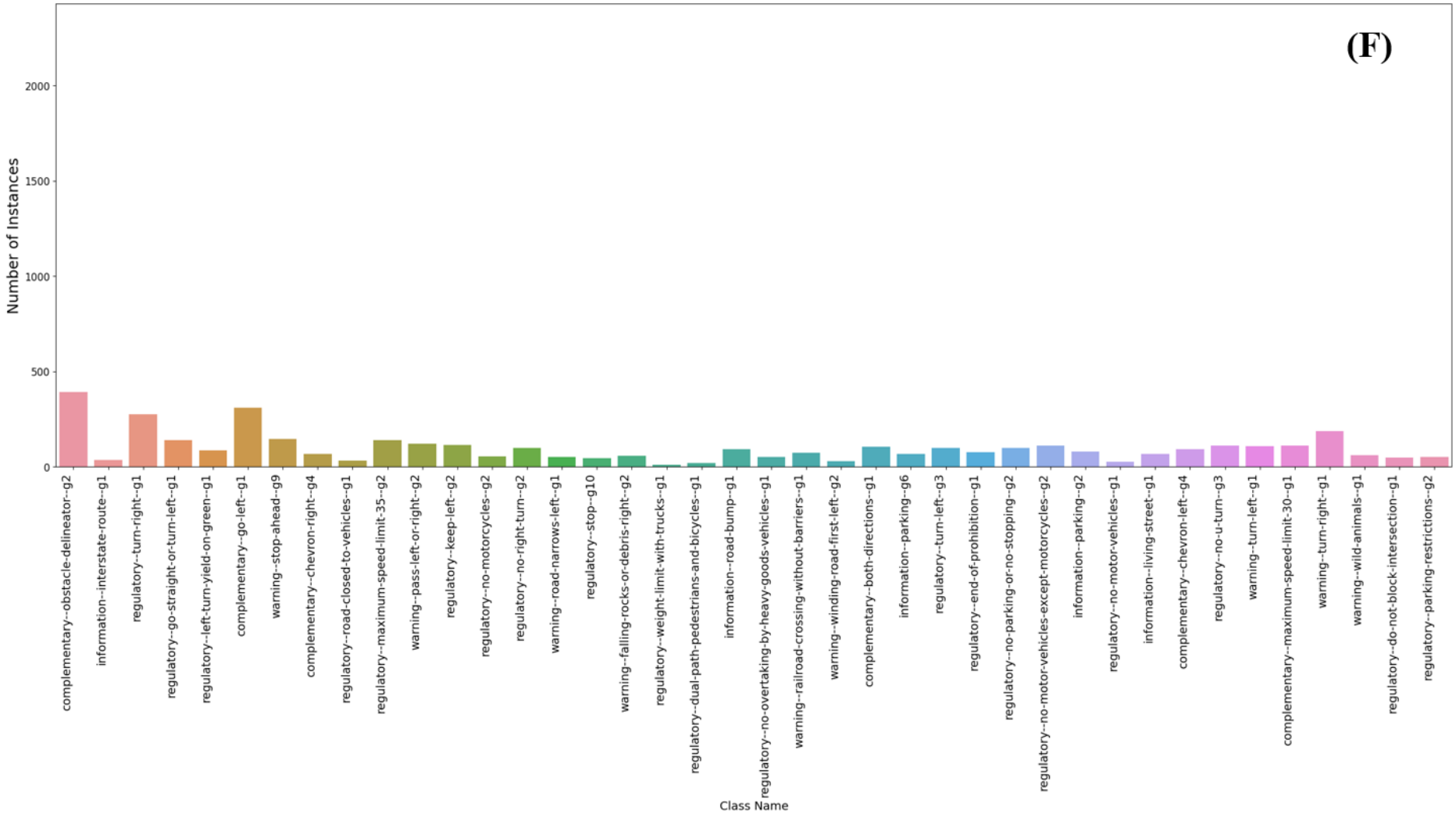


Figure A1.F: Bar charts of number of instances per class for the MTSD, without "other" class, for classes 201-240 in the dataset. The class names and subplots are arranged in class numeric order.

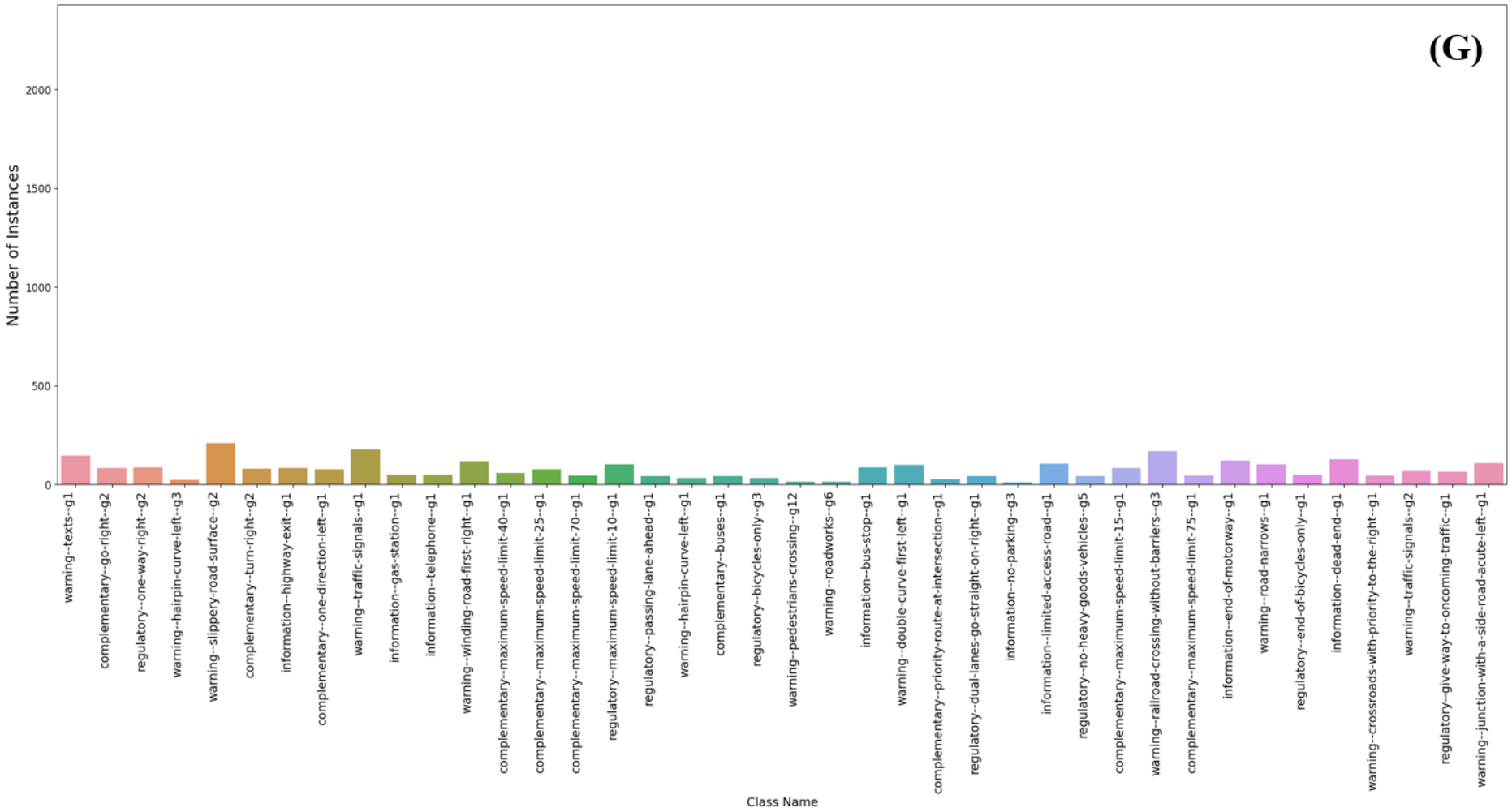


Figure A1.G: Bar charts of number of instances per class for the MTSD, without "other" class, for classes 241-280 in the dataset. The class names and subplots are arranged in class numeric order.

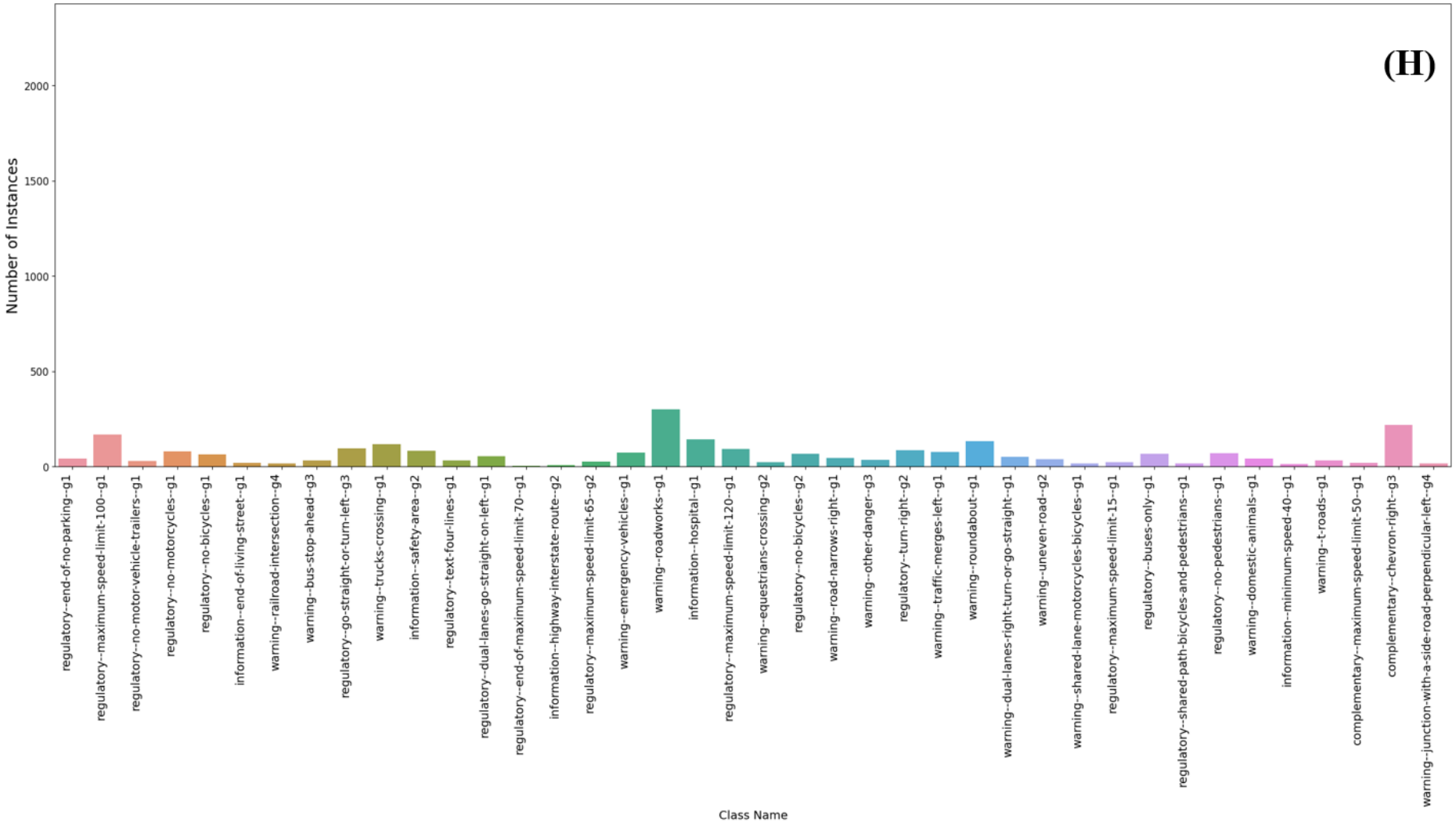


Figure A1.H: Bar charts of number of instances per class for the MTSD, without "other" class, for classes 281-320 in the dataset. The class names and subplots are arranged in class numeric order.

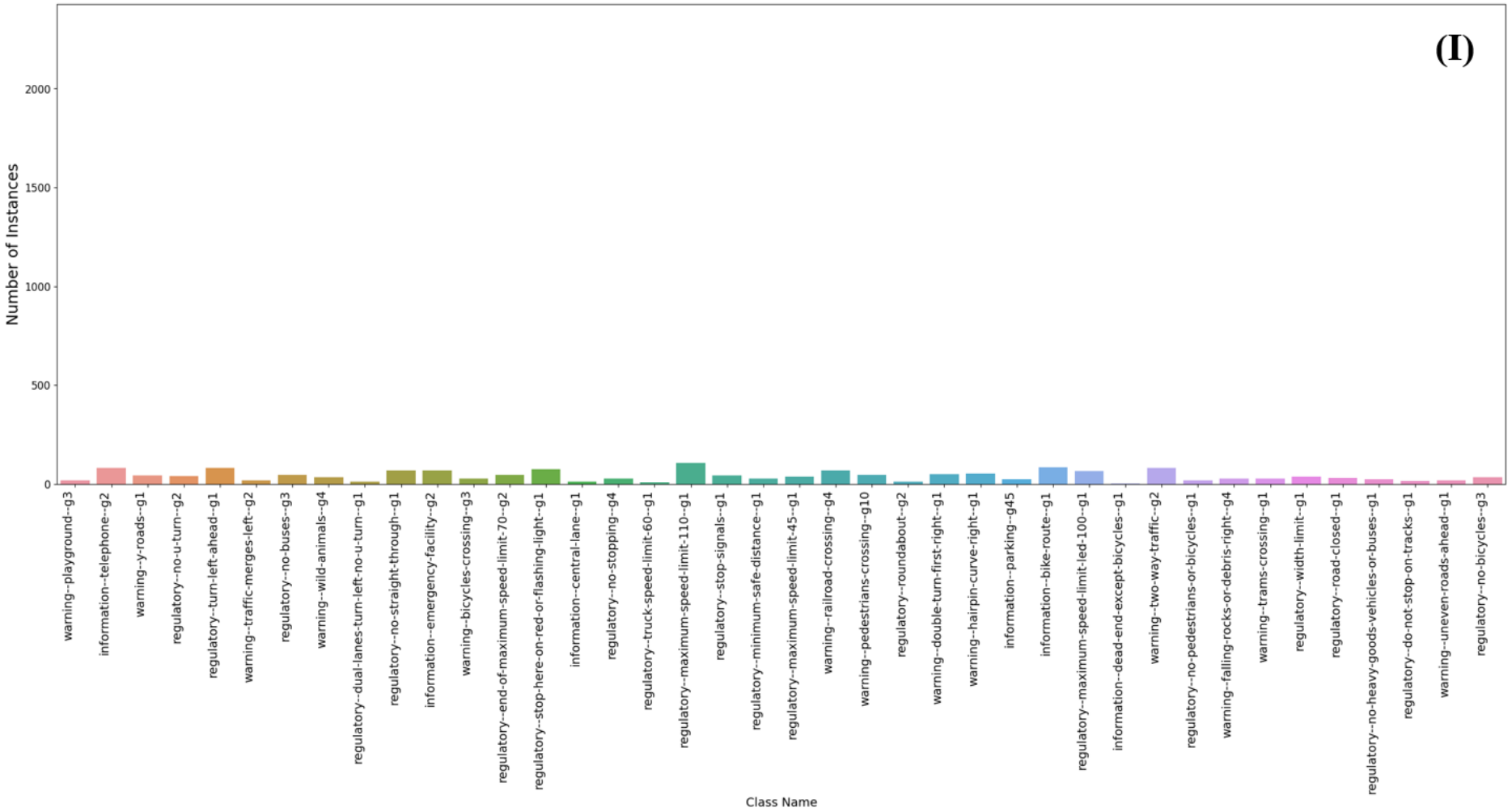


Figure A1.I: Bar charts of number of instances per class for the MTSD, without "other" class, for classes 321-360 in the dataset. The class names and subplots are arranged in class numeric order.

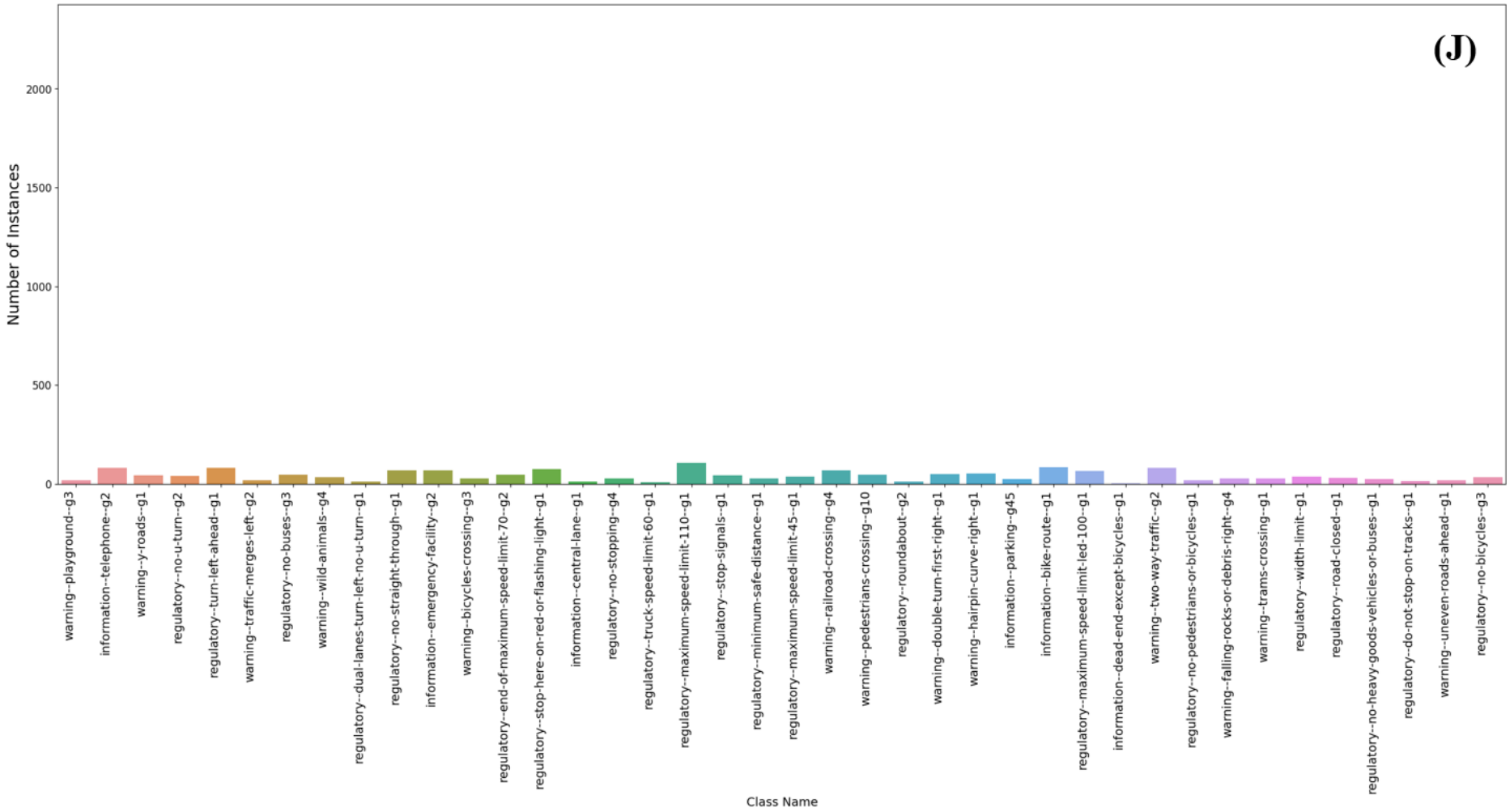


Figure A1.J: Bar charts of number of instances per class for the MTSD, without "other" class, for classes 361-400 in the dataset. The class names and subplots are arranged in class numeric order.

Appendix 2. Final Dataset Distribution

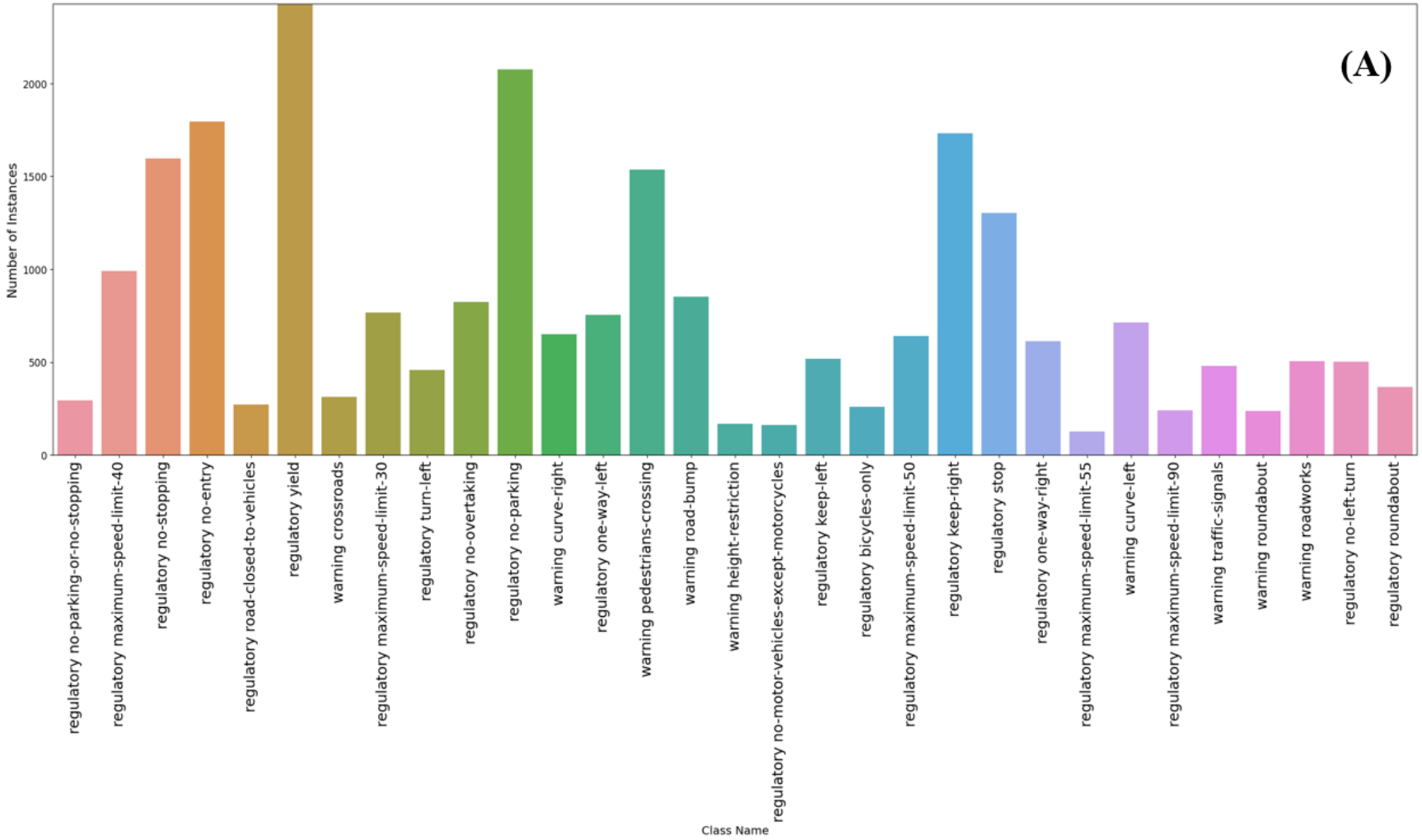


Figure A2.A: Bar chart for the final distribution of classes depicting the number of instances per class for the first 31 classes. The class names and subplots are arranged in class numeric order.

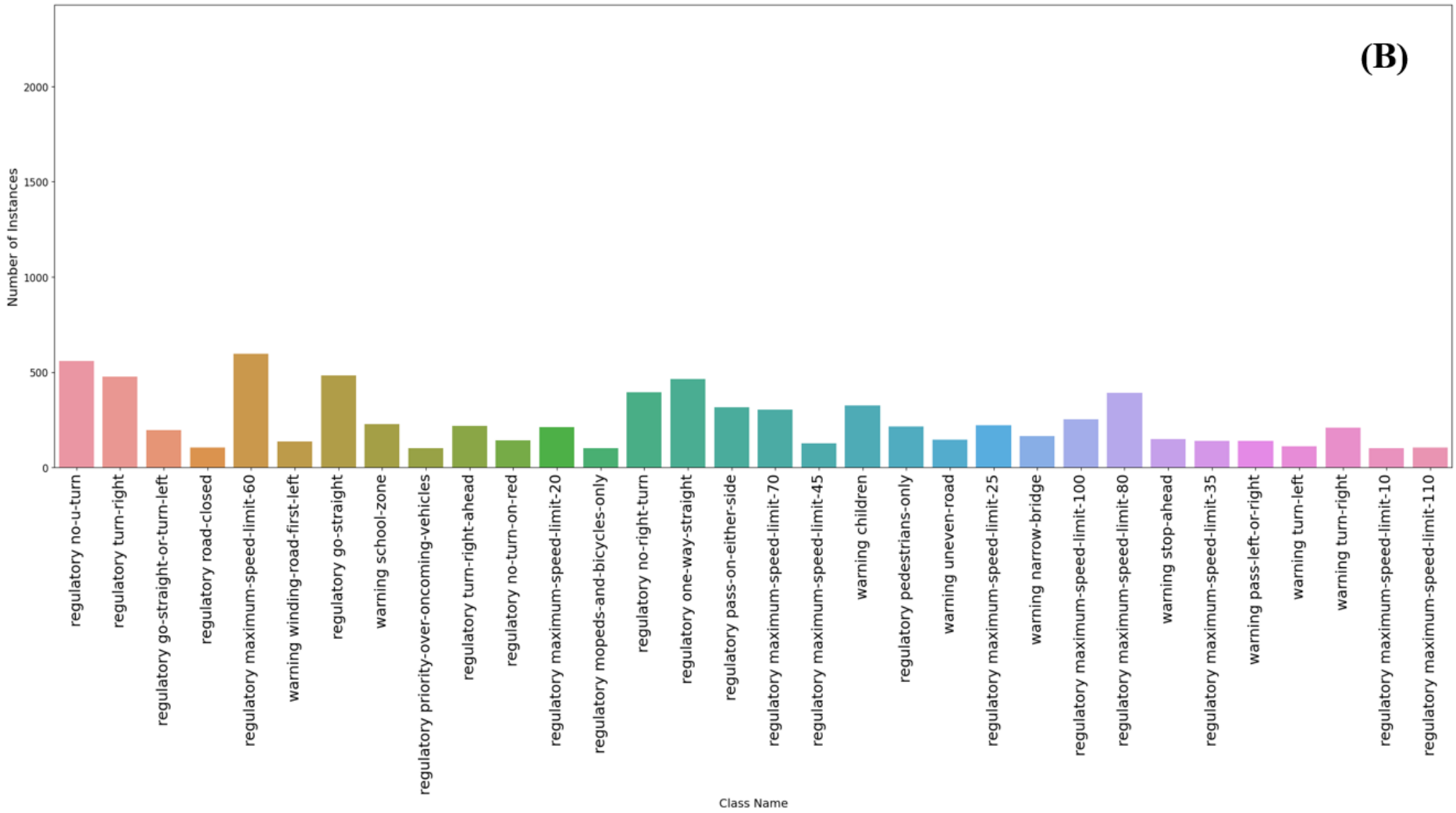


Figure A2.B: Bar chart for the final distribution of classes depicting the number of instances per class for the remaining 32 classes. The class names and subplots are arranged in class numeric order.

Appendix 3. Hyperparameter Definitions

Table A3: A list of all yolov5 and custom hyperparameters. Rows highlighted in bold were determined to be more relevant to the model or important for training and therefore these hyperparameters were changed appropriately by 20% from the default values.

	Hyperparameter name	Function Description
Model Learning Hyperparameters	**Learning rate**	Sets the step size after each epoch. A higher learning rate leads to a faster convergence however there is a risk of overshooting from the optimal solution.
	Batch Size	Determines the number of samples for each epoch of training.
	Momentum	Sets the amount of epoch history to include for the learning equation to improve gradient decent.
	Weight Decay	Adds a penalty term to the cost function of the learning process which can prevent overfitting.
	Warmup Epochs	Specifies initial epochs with low learning rate at the beginning of training, allowing the CNN to adapt to the data slowly.
	Box loss gain	Sets a multiplier which places more emphasis on the accuracy of the bounding box.
	Class Loss gain	Sets a multiplier value which places more emphasis on the accuracy of the predicted class
	Class BCE Loss	Controls the loss function binary cross entropy loss (BCE) used for binary classification.
	Object Loss gain	Sets a multiplier value which places more emphasis on the accuracy of the number of objects in an image
	Object BCE Loss	Controls the loss function binary cross entropy loss (BCE) used for binary classification.
	IOU training threshold	Controls the Intersection Over Union (IOU) threshold for acceptable classifications which dictates the model's sensitivity to slight changes between the predicted and ground truth bounding boxes.
	Anchor-multiple threshold	Adjusts how sensitive the model is to variations in the aspect ratio of the predicted bounding boxes.
	Anchors per output layer	Specifies how many anchor boxes are used to generate a bounding box for each cell in the grid. For larger image sizes a greater value should be used.
	Focal Loss gamma	This gives a weight to each sample based on its difficulty to classify (with rare objects usually being ranked highly) to improve the model's ability to detect rare objects.

	Hyperparameter name	Function Description
Dataset Augmentation Hyperparameters	**Scaling**	Resizes all images to a standard size. A larger size requires higher resolution images and more RAM but leads to better results.
	Mosaic	Creates a large new image containing a combination of multiple images, useful when detecting multiple objects in one image.
	Degree	Rotates all images by a given degree range randomly.
	flipud:	Flips images up and down randomly.
	fliplr:	Flips images left and right randomly.
	Hsv_h	Applies image HSV-Hue augmentations.
	Hsv_s	Applies image HSV-Saturation augmentations.
	Hsv_v	Applies image HSV-Value augmentations.
	Translate	Translates all images by a percentage.
	Shear	Distorts the image by shifting pixels in a fixed direction based on given range.
	Perspective	Simulates changes in camera angle or viewpoint.
	Mix-up	Creates new images by taking a weighted average of two image's pixel values.
	Copy_paste	Copies small sections of an image to create a new image
Custom Hyperparameters	Resizing value	Controls the amount of extra image space an image is assigned when too small.
	Frequency threshold	Sets the minimum number of images per class
	Quantile resizing number	Specifies the quantile that is taken to choose a resize value for the image

Appendix 4. Baseline Results of Precision, Recall, Loss functions

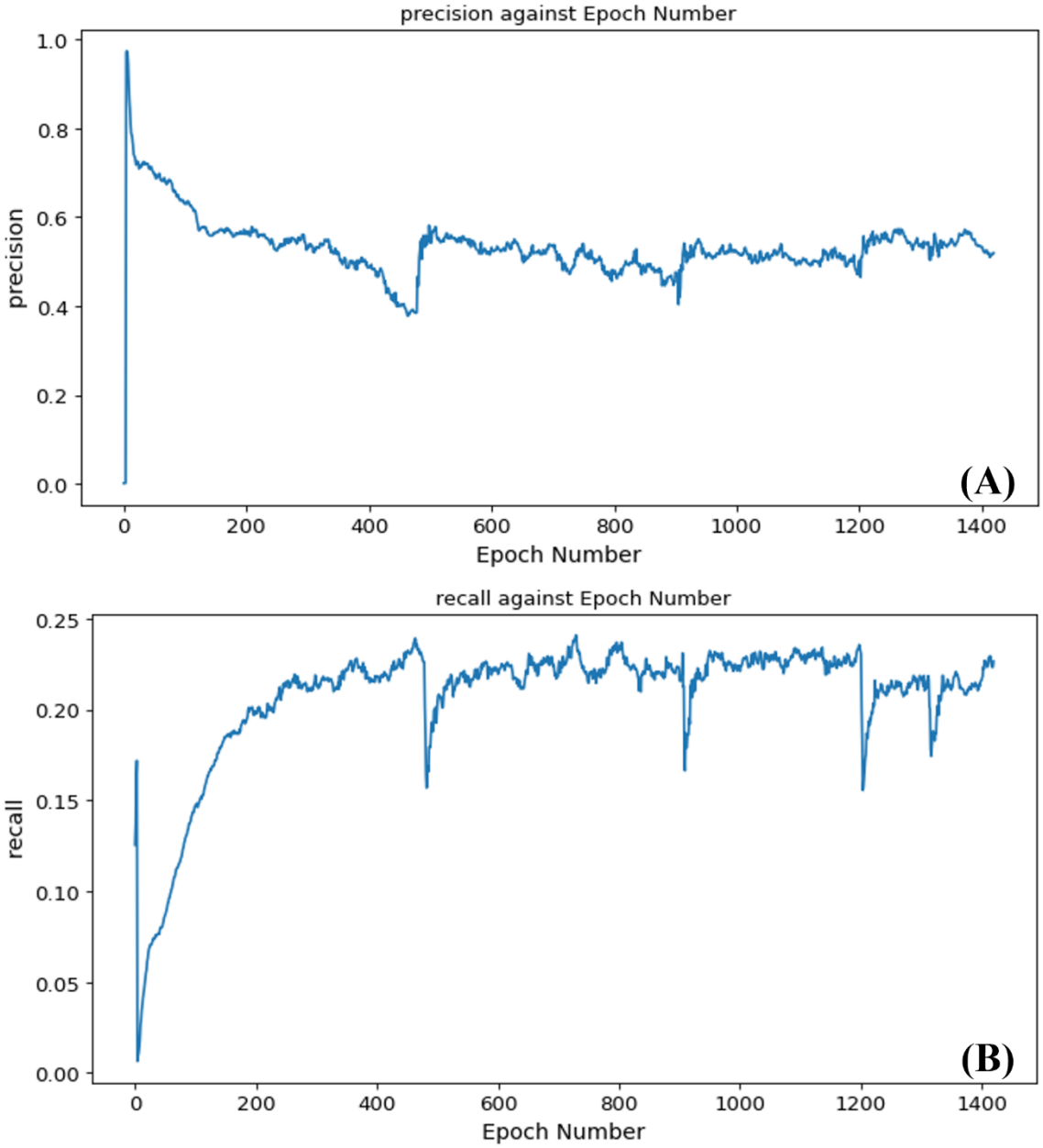


Figure A4.1: Graphs of Precision (A) and Recall (B) against the number of epochs for baseline training

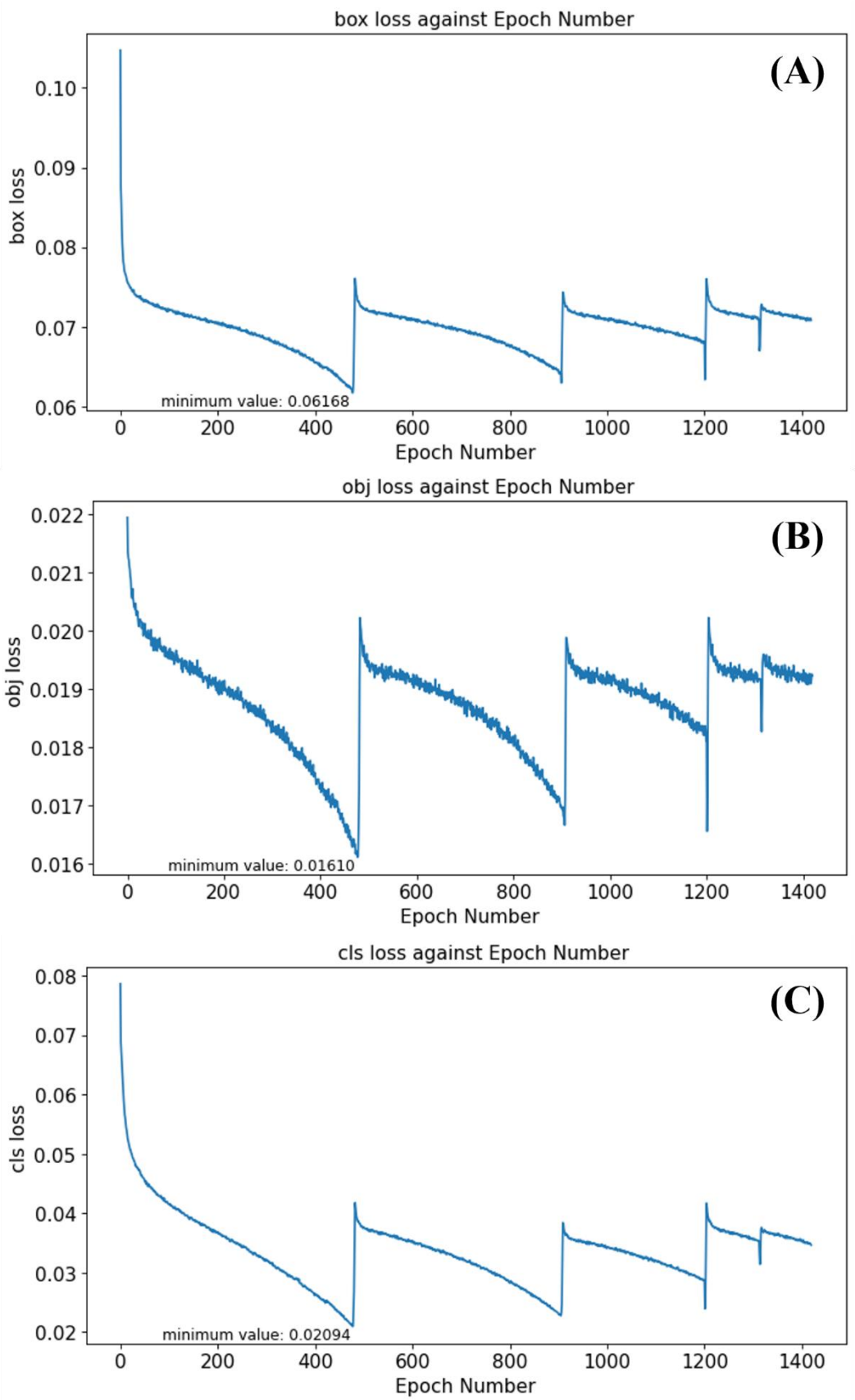


Figure A4.2: Graphs of different loss functions vs epoch numbers. (A) Box loss, (B) Object Loss, (C) Class Loss for baseline training.

Appendix 5. Class Reduction Training Results of Precision, Recall, Loss Functions

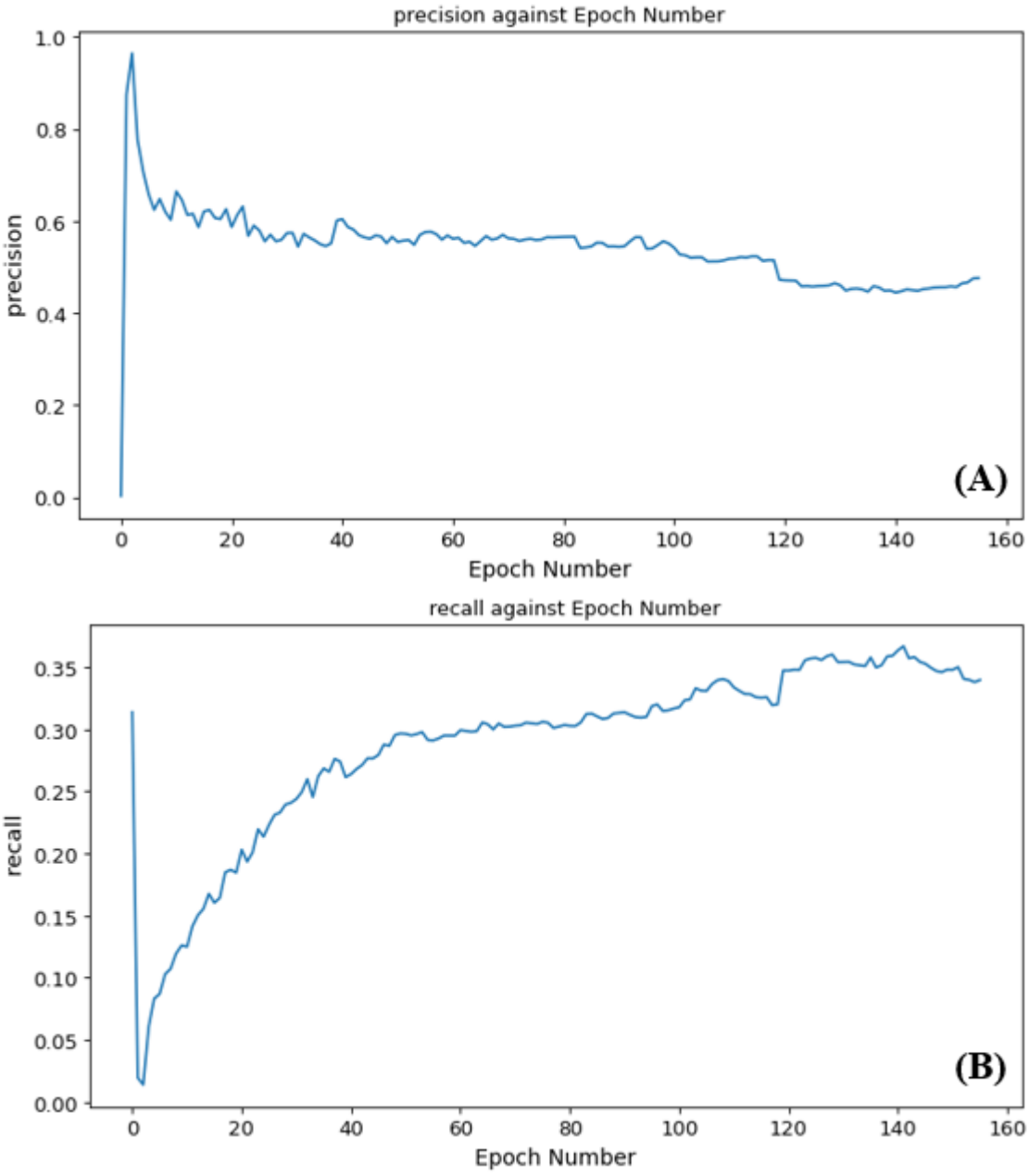


Figure A5.1: Graphs of Precision (A) and Recall (B) against the number of epochs for training after class reduction

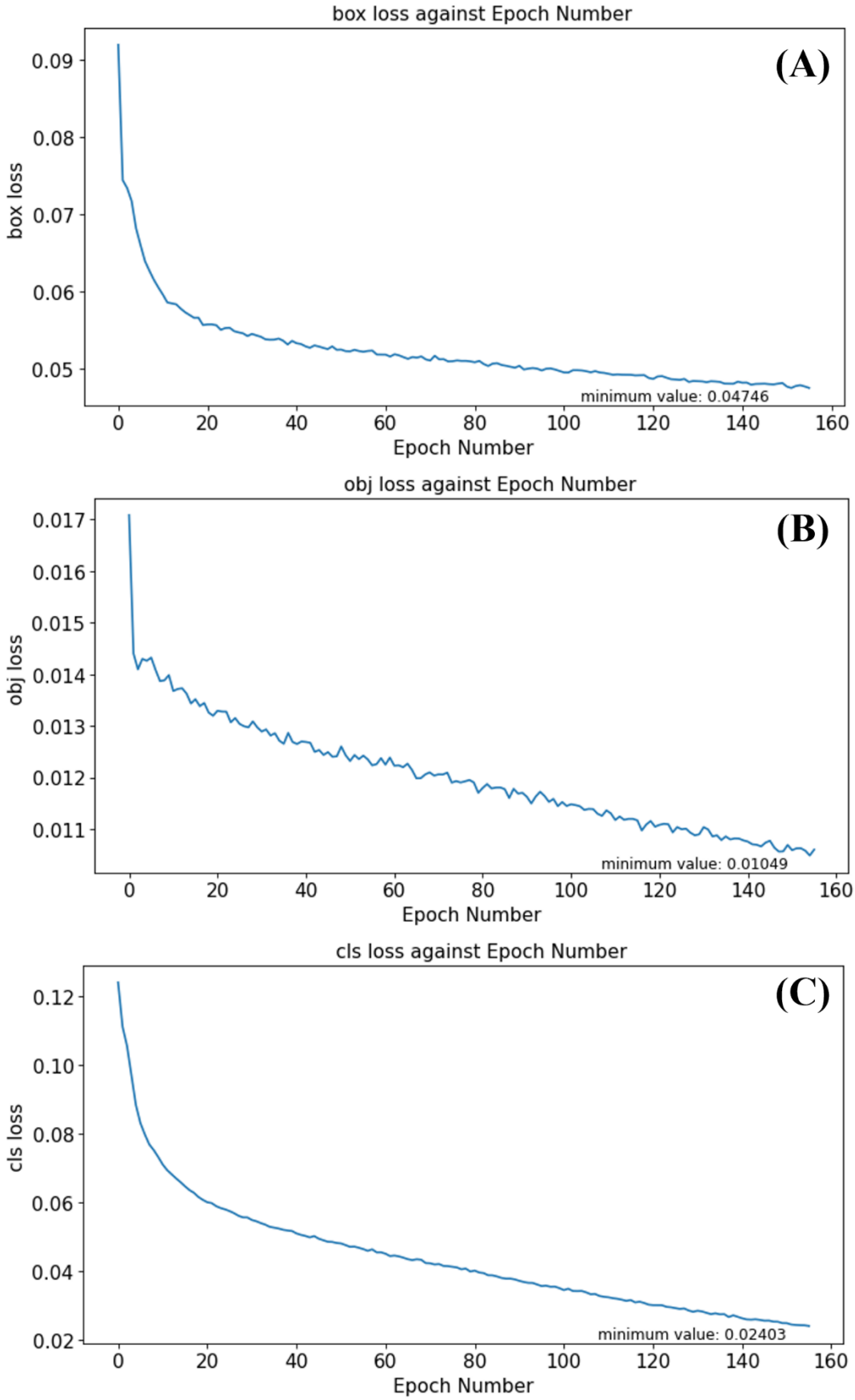


Figure A5.2: Graphs of different loss functions vs epoch numbers. (A) Box Loss (B) Object Loss (C) Class Loss for training after class reduction

Appendix 6. Training Results of Precision, Recall, Loss Functions After Image Cropping

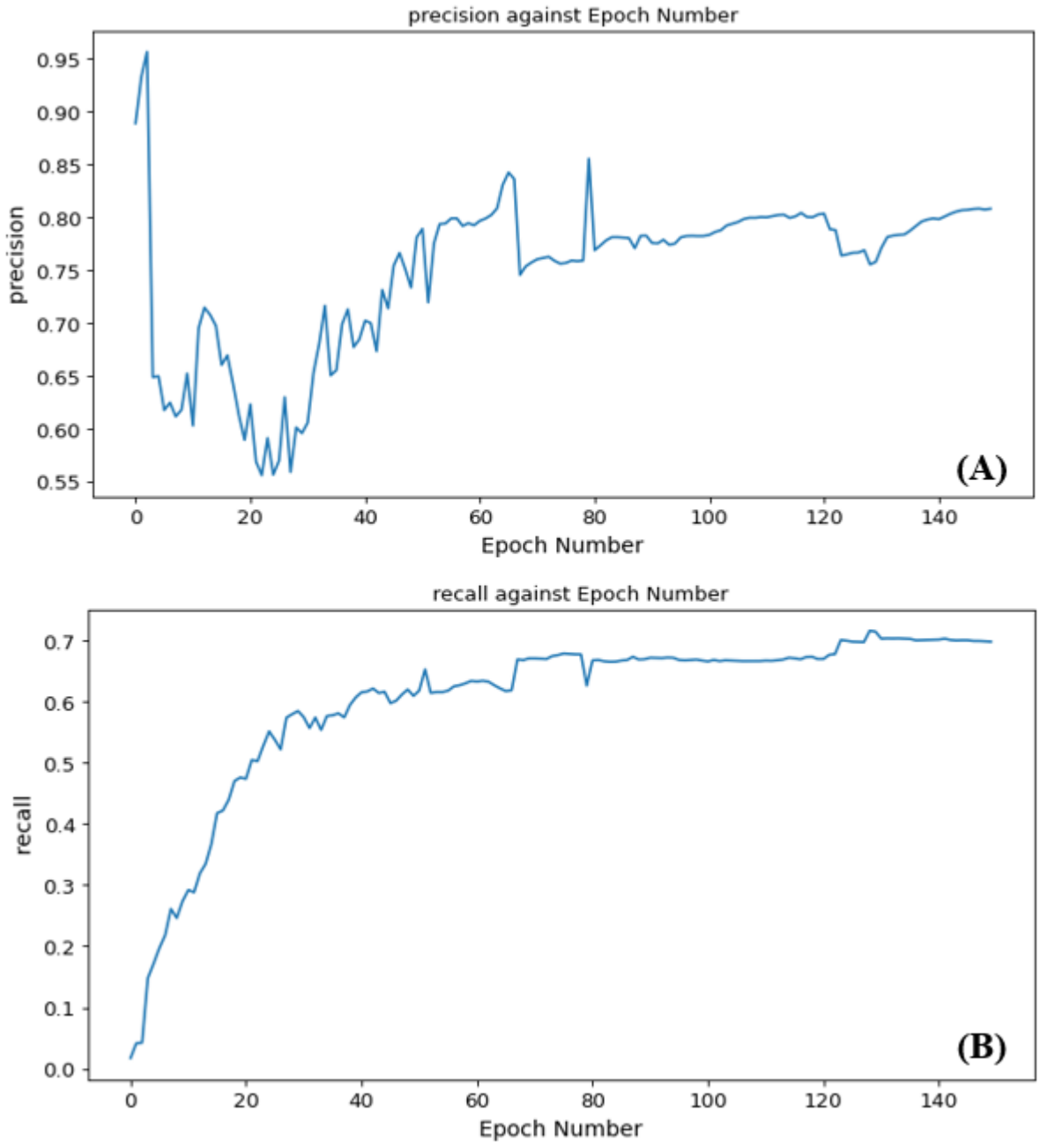


Figure A6.1: Graphs of Precision (A) and Recall (B) against the number of epochs for training using image cropping.

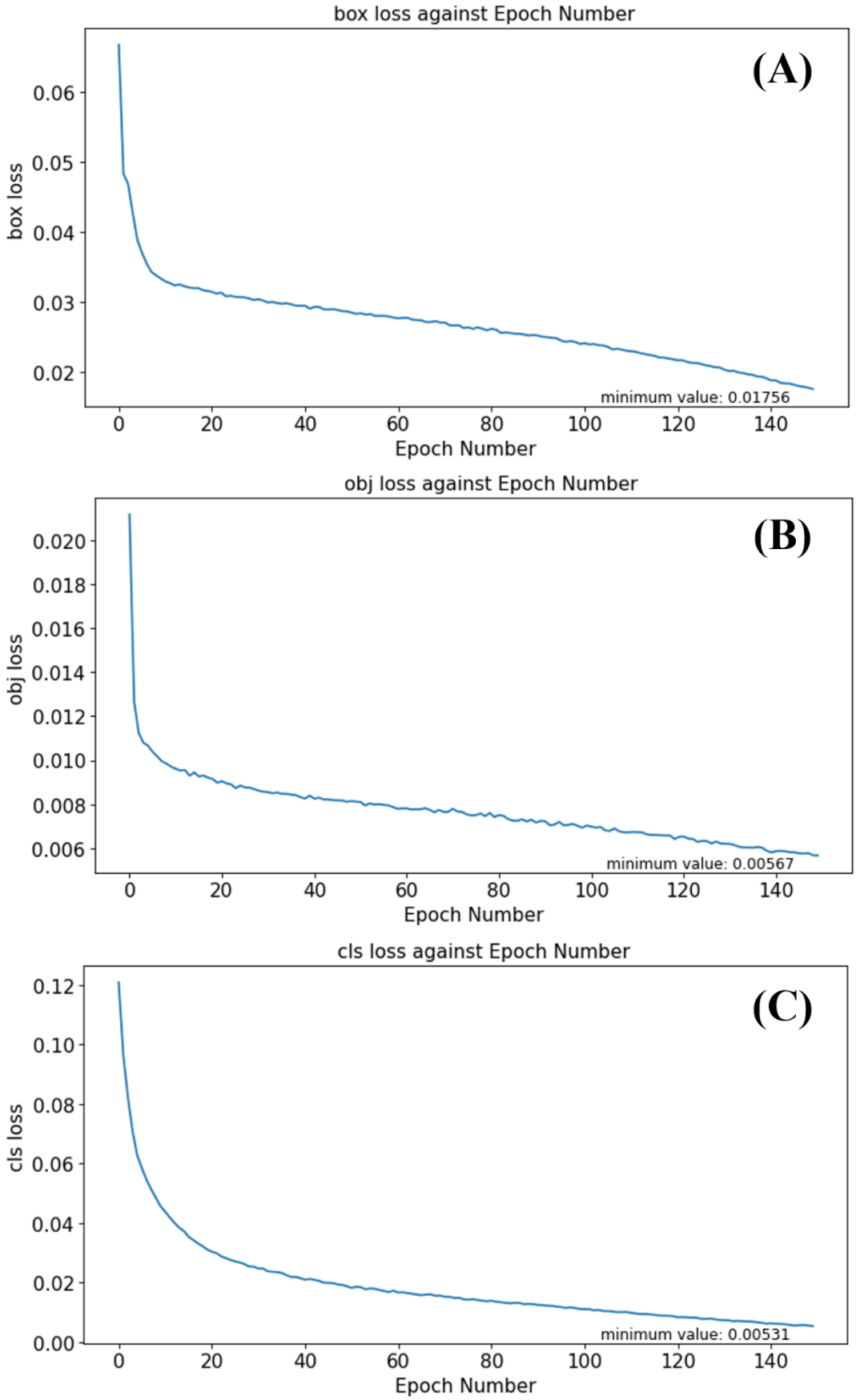


Figure A6.2: Graphs of different loss functions vs epoch numbers. (A) Box Loss (B) Object Loss (C) Class Loss for training with image cropping.

Appendix 7. Background Elimination Training Results of Precision, Recall, Loss functions

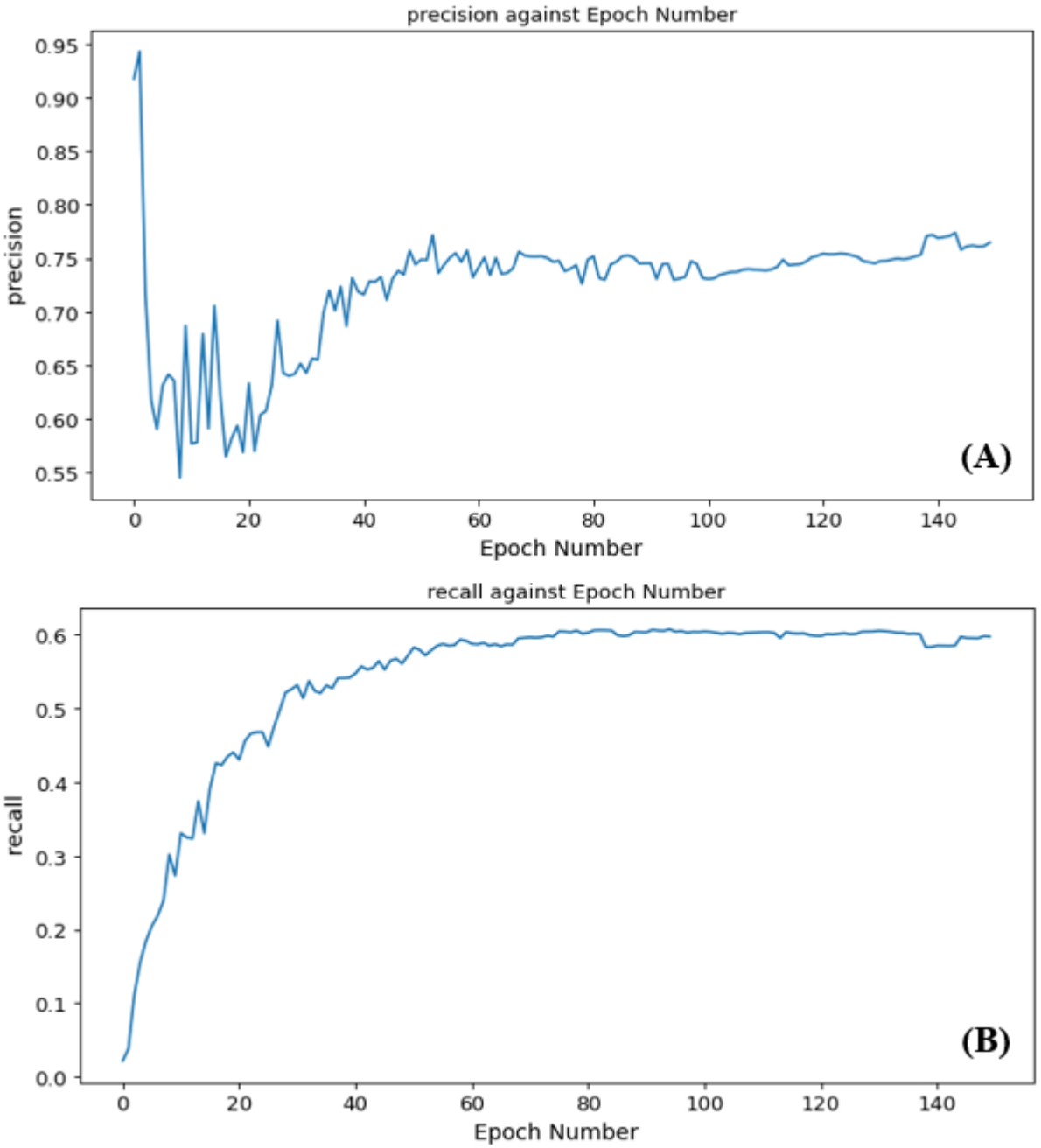


Figure A7.1: Graphs of Precision (A) and Recall (B) against the number of epochs for training using background elimination.

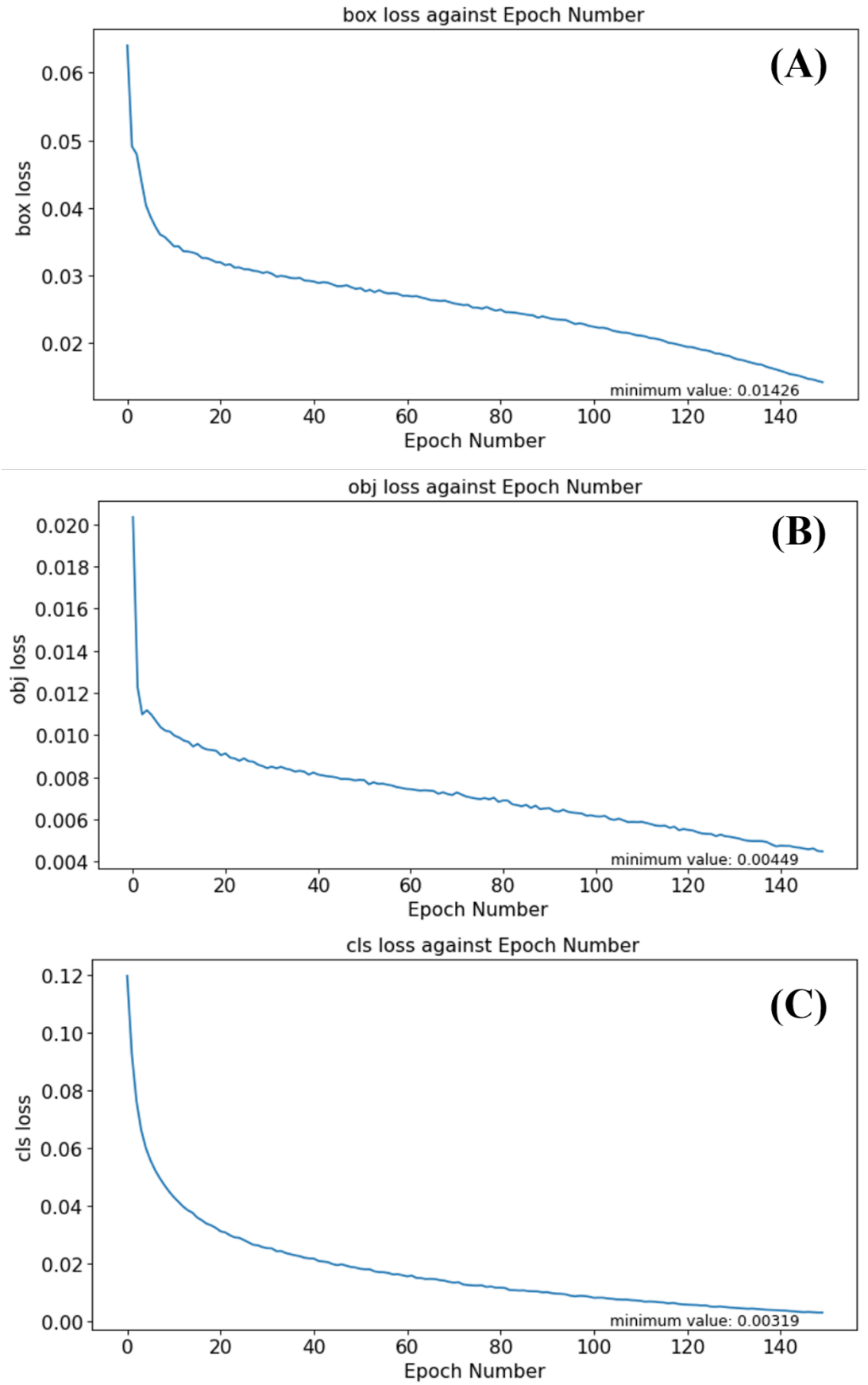
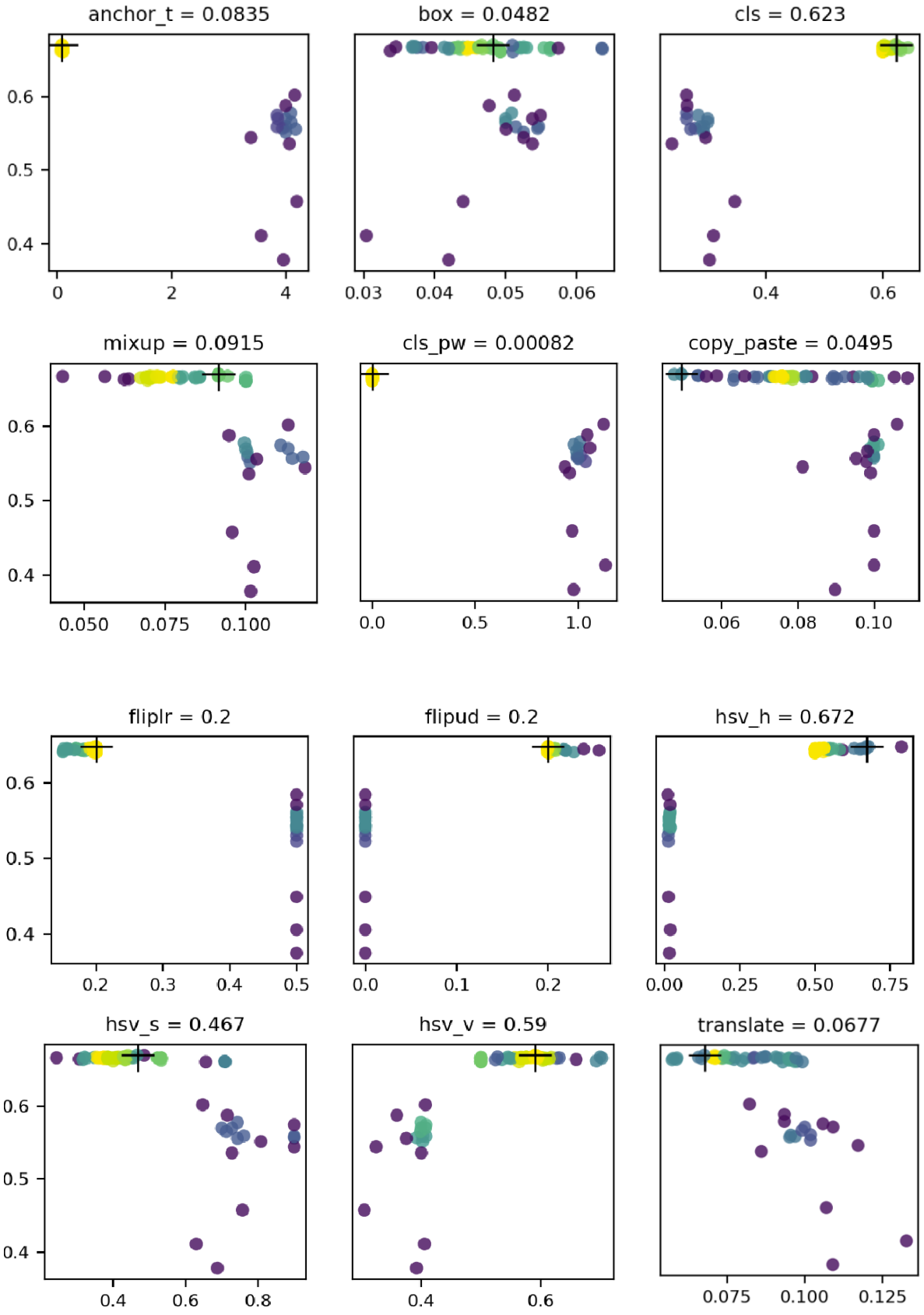


Figure A7.2: Graphs of different loss functions vs epoch numbers. (A) Box Loss (B) Object Loss (C) Class Loss for training with background elimination.

Appendix 8. Graphs of Hyperparameter Movements



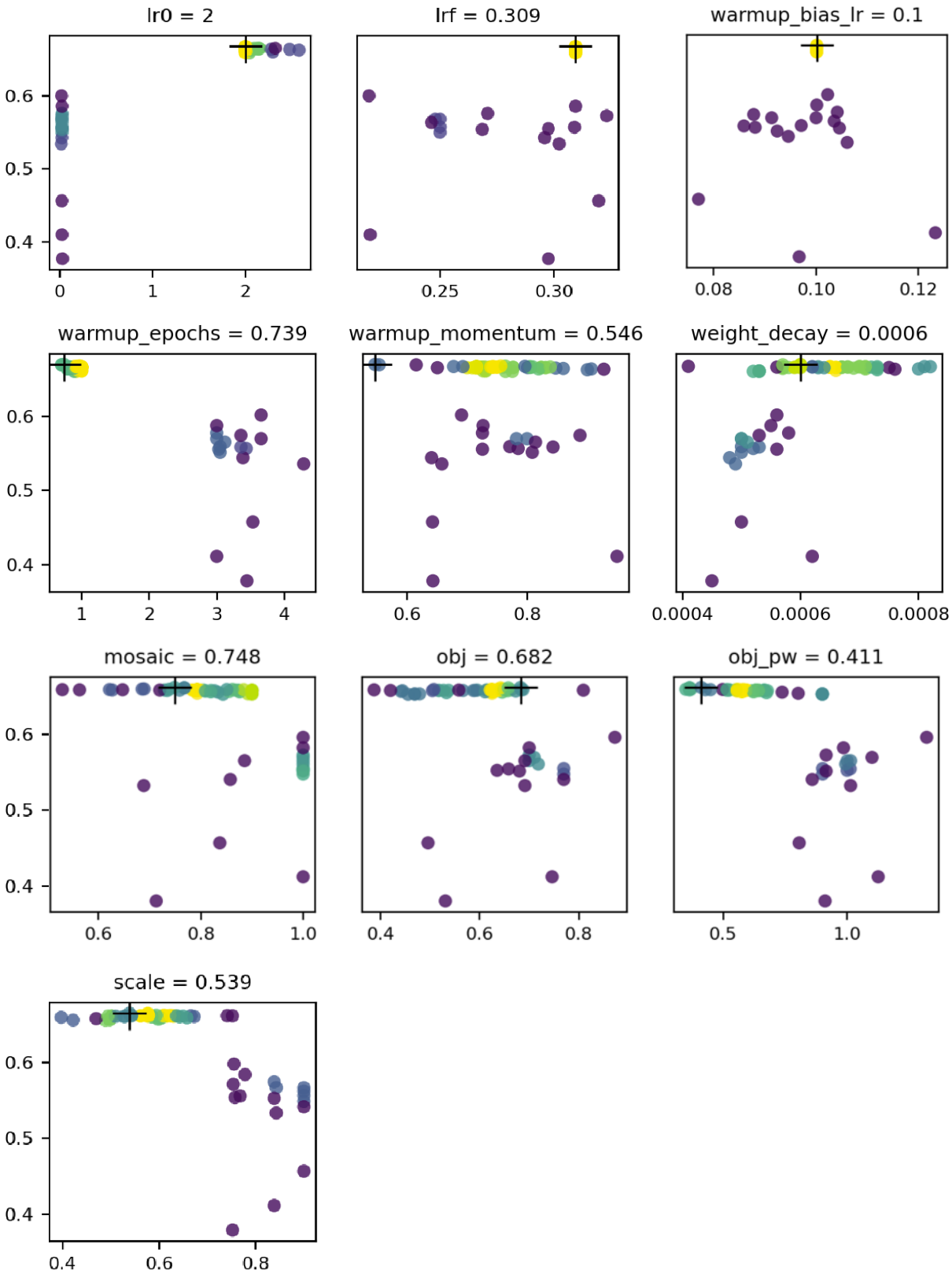


Figure A10: Plots of individual hyperparameter movements for each generation. Only plots for hyperparameters that changed value during the evolution process are shown. Each plot is given in terms of mAP50:90 (y-axis) against hyperparameter value (x-axis). Where mAP50:90 represents the mean average precision across an IoU threshold from 0.5 to 0.9. The best value chosen for each parameter is shown at the top of each plot and marked with a black cross on the diagram. Each circle represents one generation, with the colours within representing the area's density, with blue representing low density and yellow representing high density.

Appendix 9. Hyperparameter Training Results of Precision, Recall, Loss functions

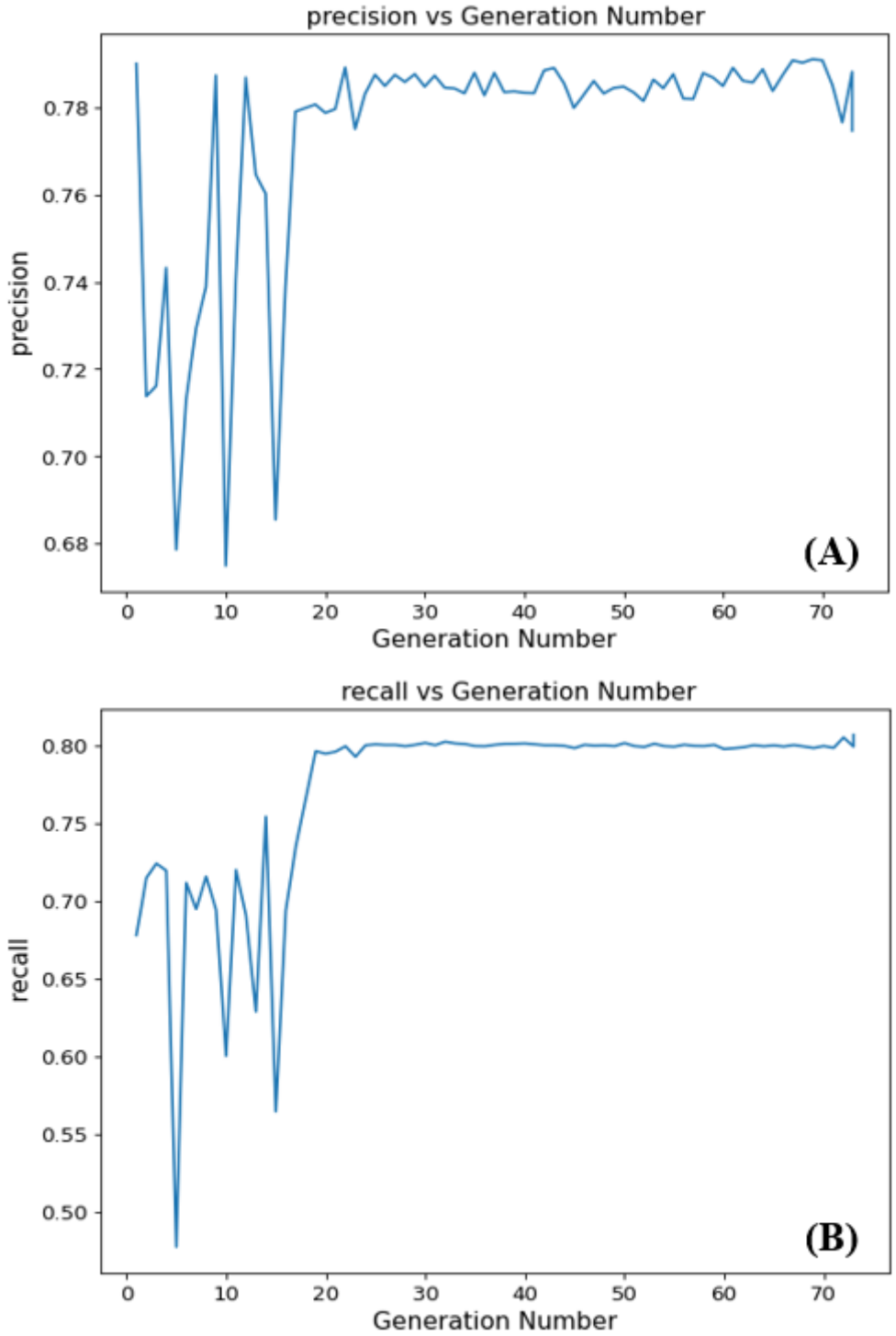


Figure A8.1: Precision (A) and recall (B) curves against generation number for hyperparameter evolution training. The high variance in precision is caused by an abundance of false negatives (missing traffic signs). This could be attributed to the poor class and object losses.

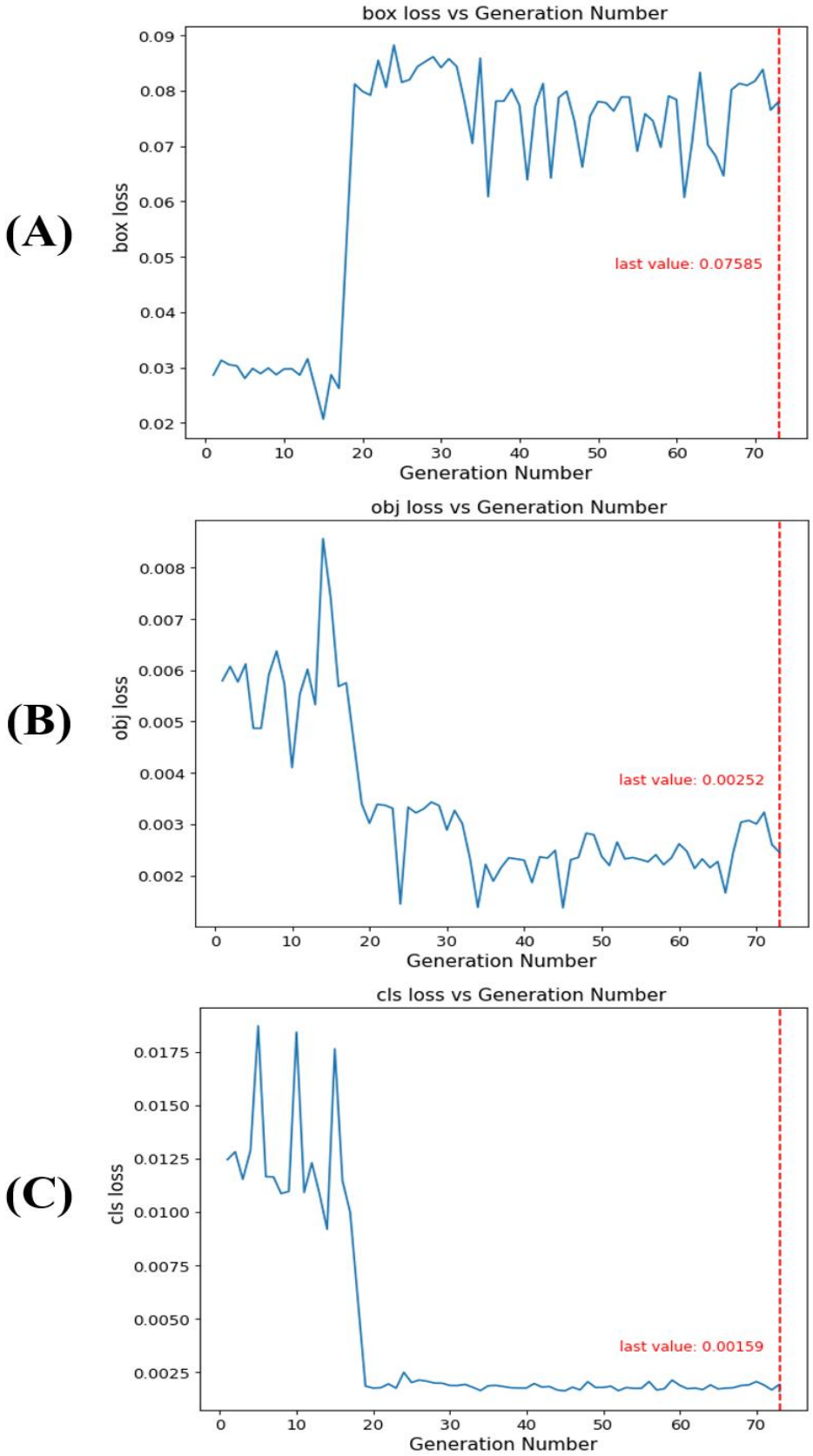


Figure A8.2: Box (A), Object (B) and Class (C) loss functions against generation number for hyperparameter evolution. Due to the iterative improvement of hyperparameters, convergence of the loss functions is not clear, motivating the need to train a model from scratch using these sets of hyperparameters.

Appendix 10. Final Training Results of Precision, Recall, Loss functions

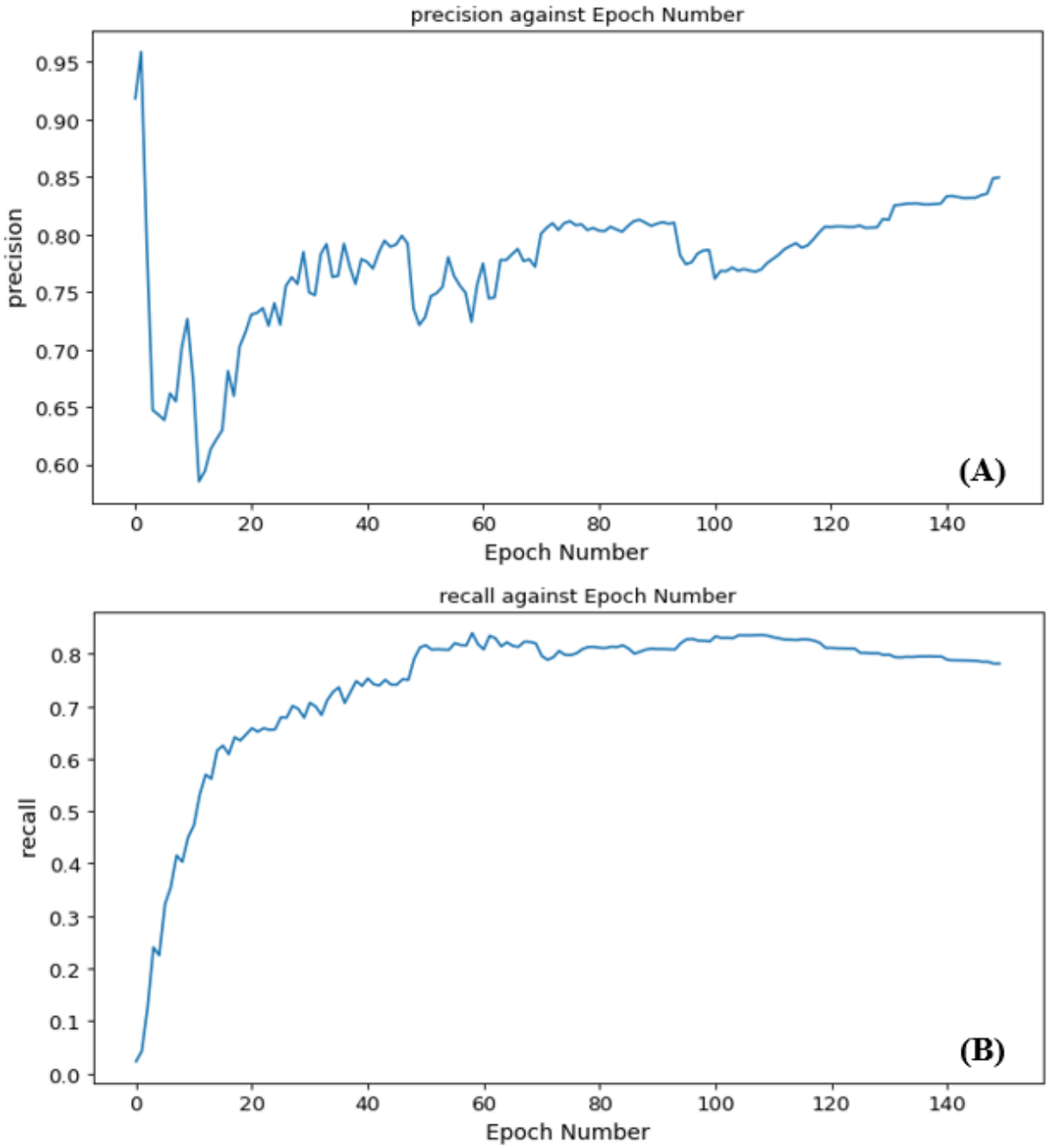


Figure A9.1: Graphs of Precision (A) and Recall (B) against the number of epochs for training using image cropping.

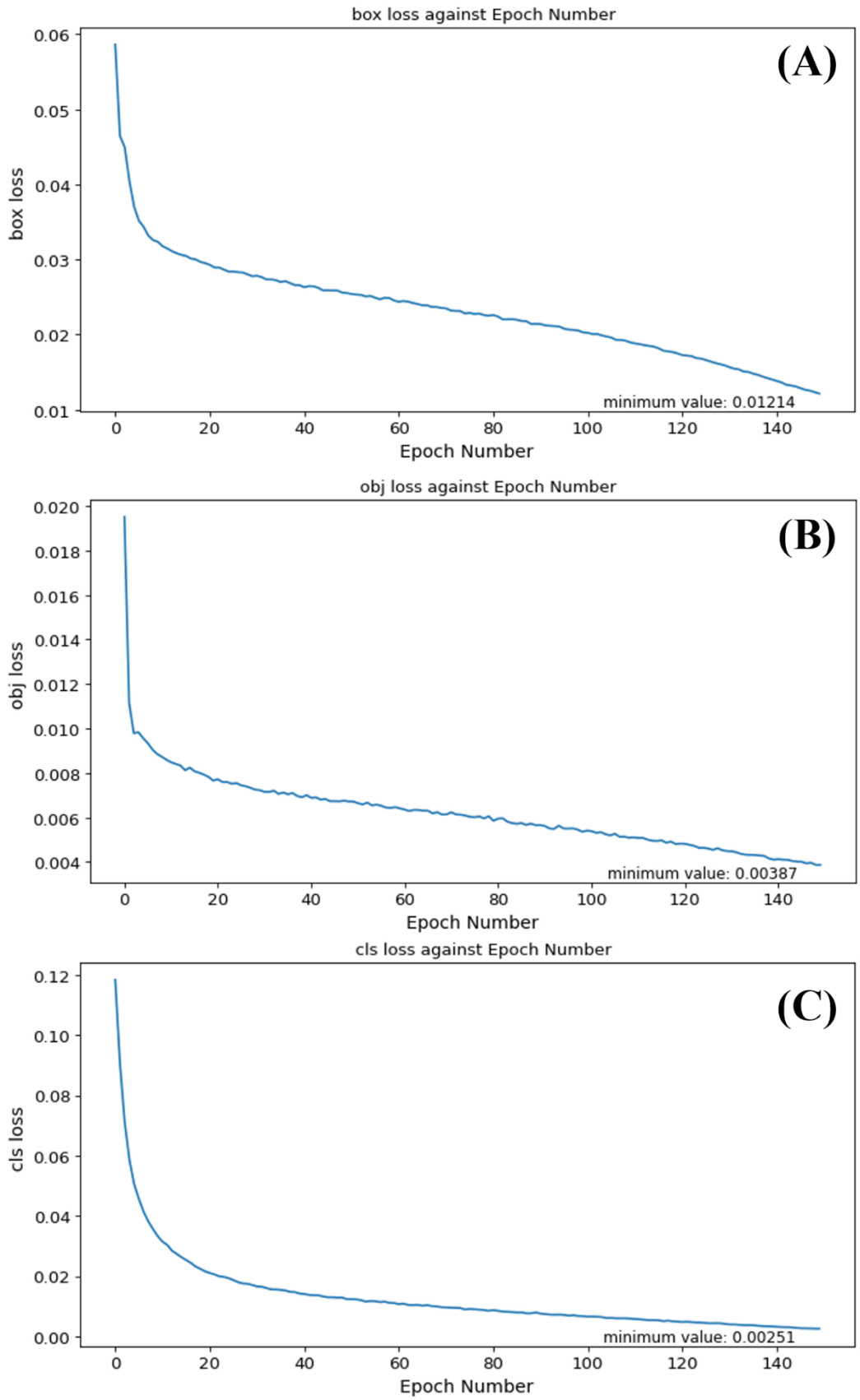
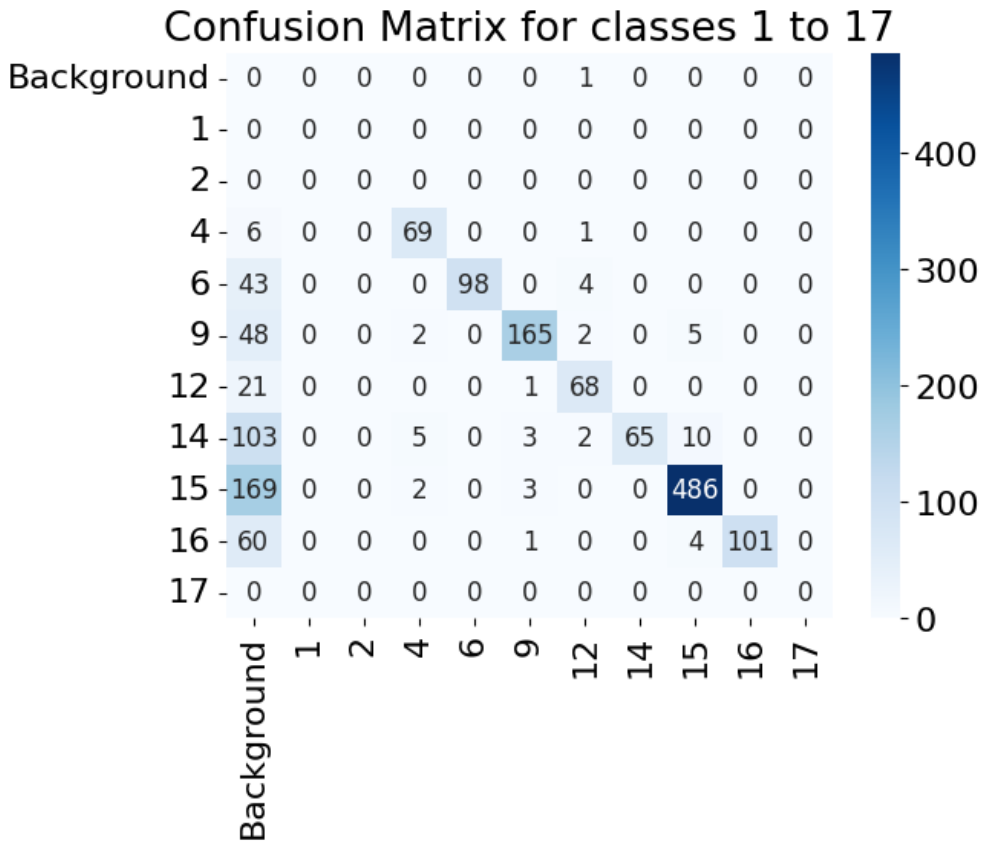
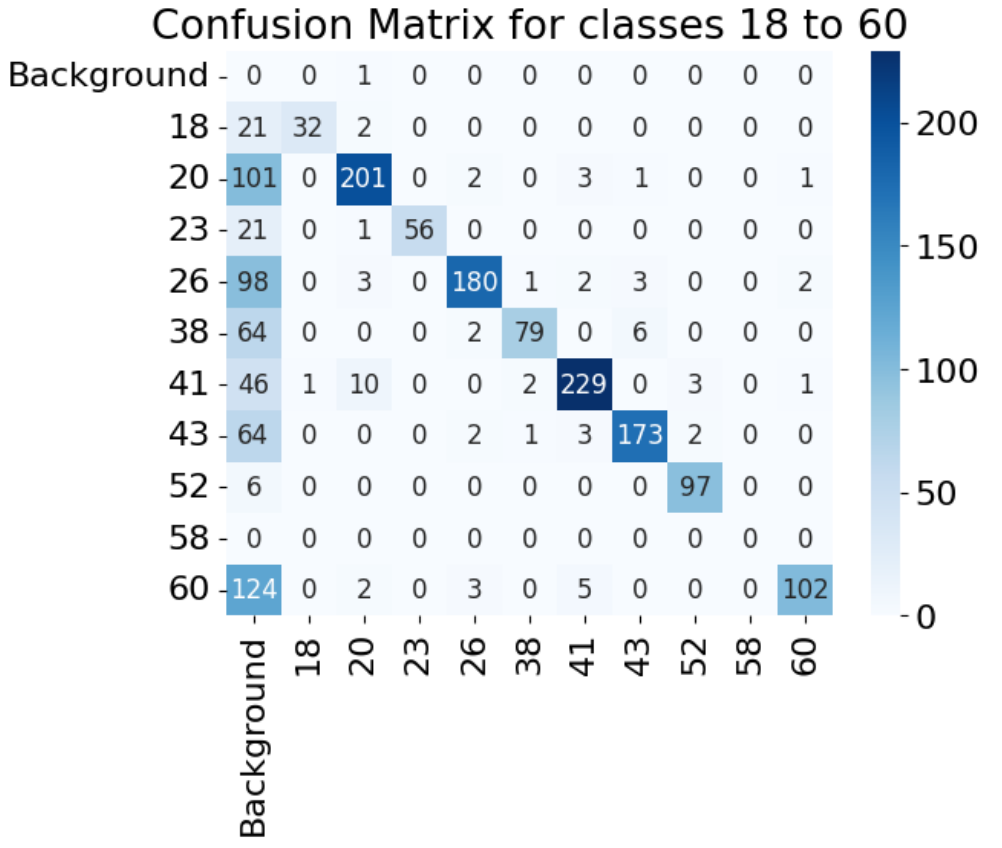


Figure A9.2: Graphs of different loss functions vs epoch numbers. (A) Box Loss (B) Object Loss (C) Class Loss for final training.

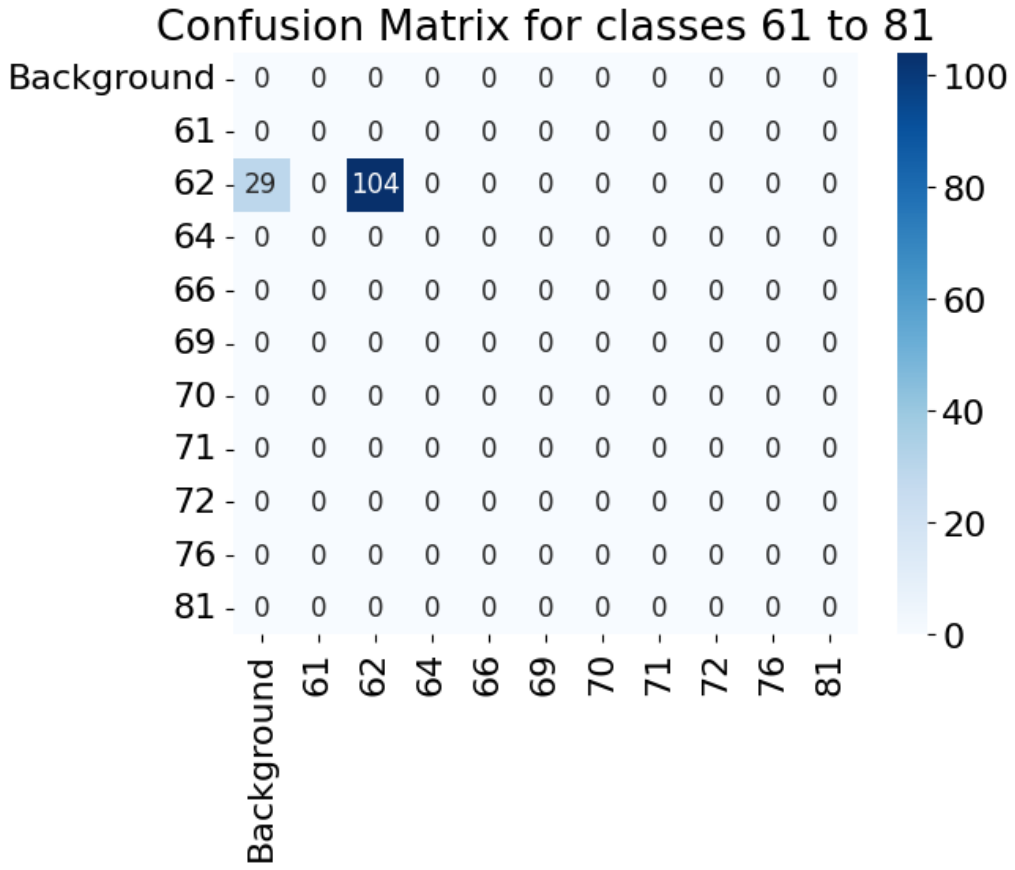
Appendix 11. Largescale Testing Confusion Sub-Matrices



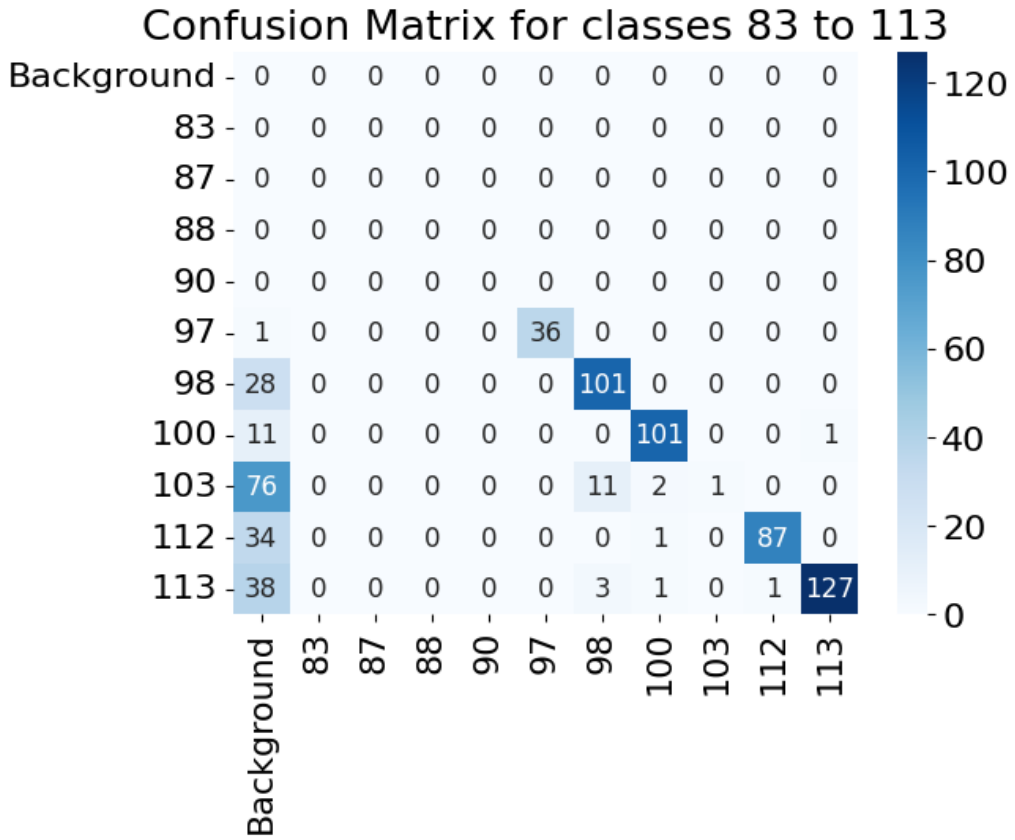
(A)



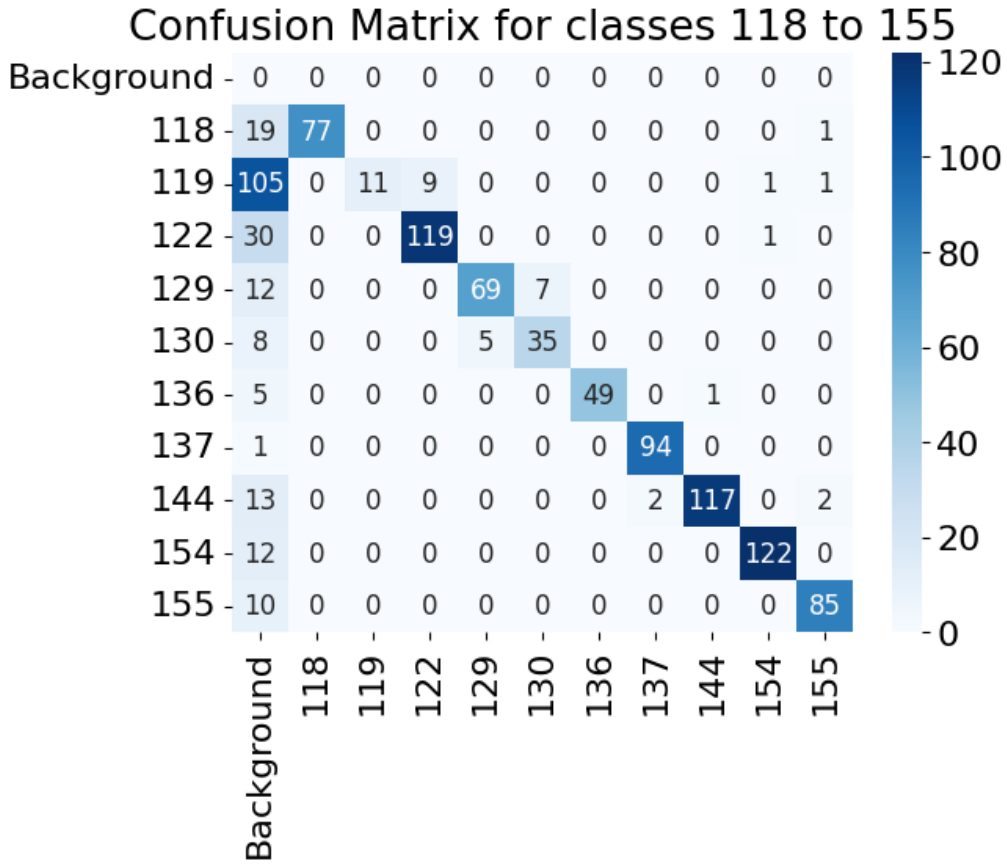
(B)


Confusion Matrix for classes 61 to 81
Background 0 0 0 0 0 0 0 0 0 0 0
61 0 0 0 0 0 0 0 0 0 0 0
62 29 0 104 0 0 0 0 0 0 0 0
64 0 0 0 0 0 0 0 0 0 0 0
66 0 0 0 0 0 0 0 0 0 0 0
69 0 0 0 0 0 0 0 0 0 0 0
70 0 0 0 0 0 0 0 0 0 0 0
71 0 0 0 0 0 0 0 0 0 0 0
72 0 0 0 0 0 0 0 0 0 0 0
76 0 0 0 0 0 0 0 0 0 0 0
81 0 0 0 0 0 0 0 0 0 0 0
Background 61 62 64 66 69 70 71 72 76 81
100 80 60 40 20 0

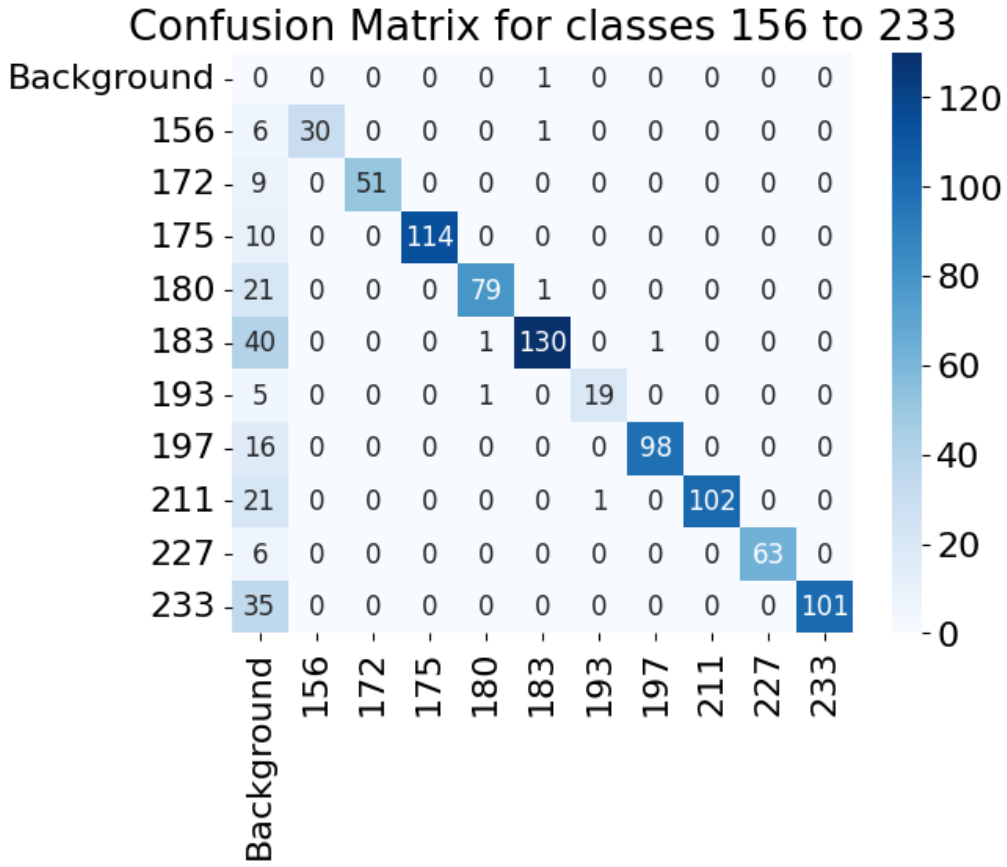
(C)


Confusion Matrix for classes 83 to 113
Background 0 0 0 0 0 0 0 0 0 0 0
83 0 0 0 0 0 0 0 0 0 0 0
87 0 0 0 0 0 0 0 0 0 0 0
88 0 0 0 0 0 0 0 0 0 0 0
90 0 0 0 0 0 0 0 0 0 0 0
97 1 0 0 0 0 36 0 0 0 0 0
98 28 0 0 0 0 0 101 0 0 0 0
100 11 0 0 0 0 0 0 101 0 0 1
103 76 0 0 0 0 0 11 2 1 0 0
112 34 0 0 0 0 0 0 1 0 87 0
113 38 0 0 0 0 0 3 1 0 1 127
Background 83 87 88 90 97 98 100 103 112 113
120 100 80 60 40 20 0

(D)


Confusion Matrix for classes 118 to 155
Background 0 0 0 0 0 0 0 0 0 0 0
118 19 77 0 0 0 0 0 0 0 0 1
119 105 0 11 9 0 0 0 0 0 1 1
122 30 0 0 119 0 0 0 0 0 1 0
129 12 0 0 0 69 7 0 0 0 0 0
130 8 0 0 0 5 35 0 0 0 0 0
136 5 0 0 0 0 0 49 0 1 0 0
137 1 0 0 0 0 0 0 94 0 0 0
144 13 0 0 0 0 0 0 2 117 0 2
154 12 0 0 0 0 0 0 0 0 122 0
155 10 0 0 0 0 0 0 0 0 0 85
Background 118 119 122 129 130 136 137 144 154 155
120 100 80 60 40 20 0

(E)


Confusion Matrix for classes 156 to 233
Background 0 0 0 0 0 1 0 0 0 0 0
156 6 30 0 0 0 1 0 0 0 0 0
172 9 0 51 0 0 0 0 0 0 0 0
175 10 0 0 114 0 0 0 0 0 0 0
180 21 0 0 0 79 1 0 0 0 0 0
183 40 0 0 0 1 130 0 1 0 0 0
193 5 0 0 0 1 0 19 0 0 0 0
197 16 0 0 0 0 0 0 98 0 0 0
211 21 0 0 0 0 0 1 0 102 0 0
227 6 0 0 0 0 0 0 0 0 63 0
233 35 0 0 0 0 0 0 0 0 0 101
Background 156 172 175 180 183 193 197 211 227 233
120 100 80 60 40 20 0

(F)

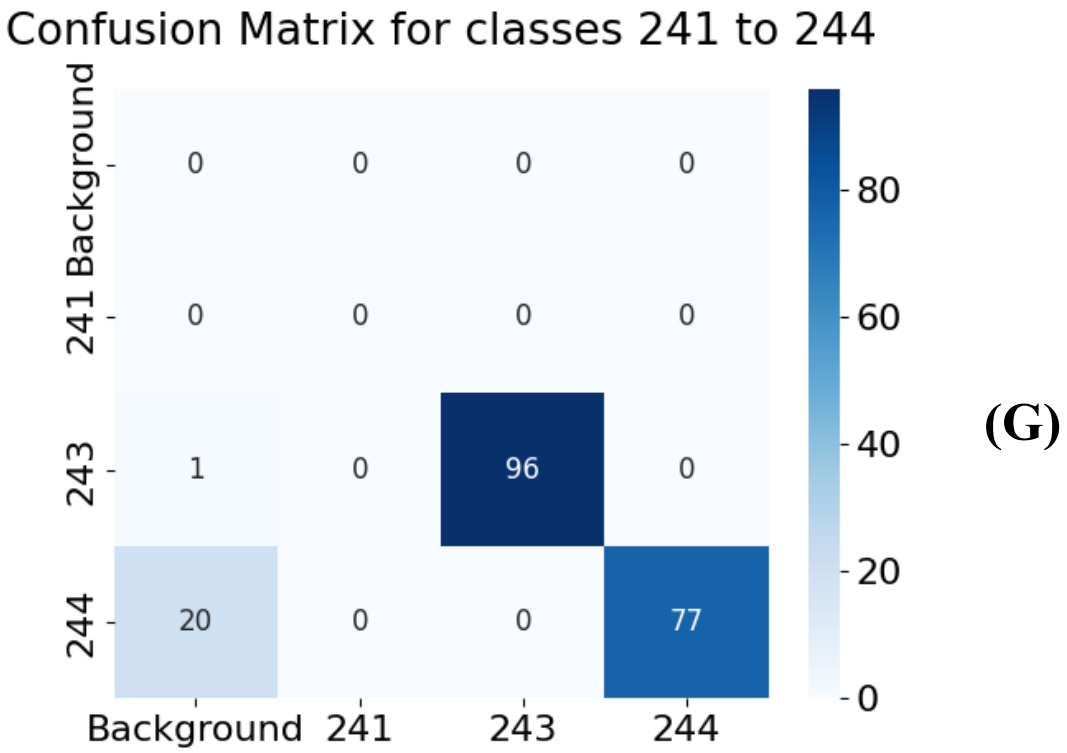


Figure A11: Confusion Matrix created from largescale testing split into smaller sub-matrices shown in figures A-G for visualisation purposes. The background class is shown for all matrices, either indicates a false negative (missed sign) or fase positive. Some diagonals have no entries, predominantly shown in Figure A10C and A10D, indicating there no traffic signs of that class in the testing dataset The complete confusion matrix made these sub-matrices was condensed into a binary confusion matrix shown in Figure 28.

References

1 Toptests.co.uk (2021). The Ultimate List of United Kingdom Driving Statistics for 2021. toptests.co.uk. Available at: https://toptests.co.uk/driving-statistics/. (Accessed: April 18, 2023).

2 Young, K.L. et al. (2019) Distraction and older drivers: An emerging problem?, Monash University. Australasian College of Road Safety. Available at: https://research.monash.edu/en/publications/distraction-and-older-drivers-an-emerging-problem. (Accessed: April 18, 2023).

3 Kashevnik, A. et al. (2021) "Driver distraction detection methods: A literature review and framework," IEEE Access, 9, pp. 60063–60076. Available at: https://doi.org/10.1109/access.2021.3073599. (Accessed: April 18, 2023).

4 Ram, T. and Chand, K. (2016) "Effect of drivers' risk perception and perception of driving tasks on road safety attitude," Transportation Research Part F: Traffic Psychology and Behaviour, 42, pp. 162–176. Available at: https://doi.org/10.1016/j.trf.2016.07.012. (Accessed 16 Apr. 2023).

5 Almost Half of UK Drivers Forget Meaning of Common Road Signs." Driver Trainer, 9 July 2021, www.drivertrainer.org/almost-half-of-uk-drivers-forget-meaning-of-common-road-signs/. (Accessed 16 Apr. 2023).

6 Oviedo-Trespalacios, O. et al. (2019) 'The impact of road advertising signs on driver behaviour and implications for Road Safety: A Critical Systematic Review', Transportation Research Part A: Policy and Practice, 122, pp. 85–98. Available at: https://doi.org/10.1016/j.tra.2019.01.012. (Accessed 16 Apr. 2023).

7 Hedges & Company (2023) Automotive trends: New car buyer demographics, DEMOGRAPHICS OF CAR BUYERS. Available at: https://hedgescompany.com/blog/2019/01/new-car-buyer-demographics-2019/#:~:text=We%20get%20asked%20a%20lot,new%20cars%20in%20the%20US. (Accessed: April 18, 2023).

8 Volvo (2019) The three-point seat belt – an innovation that saved over 1 million lives, Volvo Buses. AB Volvo. Available at: https://www.volvobuses.com/en/news/2019/jul/the-three-point-seat-belt-an-innovation-that-saved-over-1-million-lives.html#:~:text=A%20brief%20history%20of%20the,manufacturers%20to%20use%20for%20free. (Accessed: April 16, 2023).

9 Kurdthongmee, W. (2008) "Colour classification of Rubberwood Boards for fingerjoint manufacturing using a SOM neural network and image processing," Computers and Electronics in Agriculture, 64(2), pp. 85–92. Available at: https://doi.org/10.1016/j.compag.2008.04.002. (Accessed: April 18, 2023).

10 Fathahillah, F. et al. (2020) "Implementation of programmable logic controller in Multi Machine operations with product sorting and packaging based on colour detection," IOP Conference Series: Materials Science and Engineering, 732(1), p. 012069. Available at: https://doi.org/10.1088/1757-899x/732/1/012069. (Accessed: April 18, 2023).

11 Nawrat, A., & Jędrasiak, K. (2008). Fast Colour Recognition algorithm for robotics. Problemy Eksploatacji, (3), 69-76.

12 Wu, G. et al. (2019) "Automatic recognition of Juicy peaches on trees based on 3D contour features and Colour Data," Biosystems Engineering, 188, pp. 1–13. Available at: https://doi.org/10.1016/j.biosystemseng.2019.10.002. (Accessed: April 18, 2023).

13 Tan, K.W. and Stephen, I.D. (2013) "Colour detection thresholds in faces and colour patches," Perception, 42(7), pp. 733–741. Available at: https://doi.org/10.1068/p7499. (Accessed: April 18, 2023).

14 Ruta, A., Li, Y. and Liu, X. (2010) "Real-time traffic sign recognition from video by class-specific discriminative features," Pattern Recognition, 43(1), pp. 416–430. Available at: https://doi.org/10.1016/j.patcog.2009.05.018. (Accessed: April 18, 2023).

15 Malik, R., Khurshid, J. and Ahmad, S.N. (2007) "Road sign detection and recognition using colour segmentation, shape analysis and template matching," 2007 International Conference on Machine Learning and Cybernetics [Preprint]. Available at: https://doi.org/10.1109/icmlc.2007.4370763. (Accessed: April 18, 2023).

16 Kiran, C.G. et al. (2009) "Traffic sign detection and pattern recognition using support vector machine," 2009 Seventh International Conference on Advances in Pattern Recognition [Preprint]. Available at: https://doi.org/10.1109/icapr.2009.58. (Accessed: April 18, 2023).

17 Fleyeh, H. (2005) "Color detection and segmentation for road and traffic signs," IEEE Conference on Cybernetics and Intelligent Systems, 2004. [Preprint]. Available at: https://doi.org/10.1109/iccis.2004.1460692. (Accessed: April 18, 2023).

18 Nixon, M.S. and Aguado, A.S. (2020) "High-level feature extraction: Fixed shape matching," Feature Extraction and Image Processing for Computer Vision, pp. 223–290. Available at: https://doi.org/10.1016/b978-0-12-814976-8.00005-1. (Accessed: April 18, 2023).

19 Moon, H., Chellappa, R. and Rosenfeld, A. (2002) "Optimal edge-based shape detection," IEEE Transactions on Image Processing, 11(11), pp. 1209–1227. Available at: https://doi.org/10.1109/tip.2002.800896. (Accessed: April 18, 2023).

20 Md Sallah, S.S., Hussin, F.A. and Yusoff, M.Z. (2010) "Shape-based road sign detection and recognition for embedded application using MATLAB," 2010 International Conference on Intelligent and Advanced Systems [Preprint]. Available at: https://doi.org/10.1109/icias.2010.5716193. (Accessed: April 18, 2023).

21 Boumediene, M. et al. (2014) "Multi-roi association and tracking with belief functions: Application to traffic sign recognition," IEEE Transactions on Intelligent Transportation Systems, 15(6), pp. 2470–2479. Available at: https://doi.org/10.1109/tits.2014.2320536. (Accessed: April 18, 2023).

22 Rangarajan, K., Shah, M. and Van Brackle, D. (1989) “Optimal Corner Detector,” Computer Vision, Graphics, and Image Processing, 48(2), pp. 230–245. Available at: https://doi.org/10.1016/s0734-189x(89)80039-8. (Accessed: April 18, 2023).

23 Kuo, W.-J. and Lin, C.-C. (2007) “Two-stage road sign detection and recognition,” Multimedia and Expo, 2007 IEEE International Conference on [Preprint]. Available at: https://doi.org/10.1109/icme.2007.4284928. (Accessed: April 18, 2023).

24 Yakimov, P. and Fursov, V. (2015) “Traffic signs detection and tracking using modified Hough Transform,” Proceedings of the 12th International Conference on Signal Processing and Multimedia Applications [Preprint]. Available at: https://doi.org/10.5220/0005543200220028. (Accessed: April 18, 2023).

25 Liu, C. et al. (2019) “Machine vision based Traffic Sign Detection Methods: Review, analyses and perspectives,” IEEE Access, 7, pp. 86578–86596. Available at: https://doi.org/10.1109/access.2019.2924947. (Accessed: April 18, 2023).

26 Rezaei, M. and Klette, R. (2017) “Computer vision for driver assistance,” Computational Imaging and Vision [Preprint]. Available at: https://doi.org/10.1007/978-3-319-50551-0. (Accessed: April 18, 2023).

27 Ruiz, I. and Serrat, J. (2022) “Hierarchical novelty detection for traffic sign recognition,” Sensors, 22(12), p. 4389. Available at: https://doi.org/10.3390/s22124389. (Accessed: April 18, 2023).

28 Shah, T. (2020) About train, validation and test sets in machine learning, Medium. Available at: https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7. (Accessed: April 18, 2023).

29 Vankdothu, R. and Hameed, M.A. (2022) “Brain tumor MRI images identification and classification based on the recurrent convolutional neural network,” Measurement: Sensors, 24, p. 100412. Available at: https://doi.org/10.1016/j.measen.2022.100412. (Accessed: April 18, 2023).

30 Villalba-Diez, J. et al. (2019) “Deep Learning for Industrial Computer Vision Quality Control in the printing industry 4.0,” Sensors, 19(18), p. 3987. Available at: https://doi.org/10.3390/s19183987. (Accessed: April 18, 2023).

31 Brunetti, A. et al. (2018) “Computer Vision and Deep Learning techniques for pedestrian detection and tracking: A survey,” Neurocomputing, 300, pp. 17–33. Available at: https://doi.org/10.1016/j.neucom.2018.01.092. (Accessed: April 18, 2023).

32 Fan, Y.-C., Yelamandala, C.M., Chen, T.-W. and Huang, C.-J. (2021). Real-Time Object Detection for LiDAR Based on LS-R-YOLOv4 Neural Network. Journal of Sensors, 2021, pp.1–11. Available at: https://doi.org/10.1155/2021/5576262. (Accessed: April 18, 2023).

33 Nissan (2019) Traffic sign recognition: Innovation, Traffic Sign Recognition. Nissan Motor Corporation. Available at: https://www.nissan-global.com/EN/INNOVATION/TECHNOLOGY/ARCHIVE/TSR/ (Accessed: April 16, 2023).

34 Eduonix (2022) Real-world implementations of Yolo algorithm, Eduonix Blog. Available at: https://blog.eduonix.com/software-development/real-world-implementations-of-yolo-algorithm/#:~:text=Autonomous%20Driving%3A%20With%20YOLO%2C%20the,parking%20signals%20and%20traffic%20lights. (Accessed: April 16, 2023).

35 KOCAKANAT, K. and SERİF, T. (2021) 'Turkish traffic sign recognition: Comparison of training step numbers and lighting conditions', European Journal of Science and Technology [Preprint]. Available at: https://doi.org/10.31590/ejosat.1015972. (Accessed: April 18, 2023).

36 Kilic, I. and Aydin, G. (2020) "Traffic sign detection and recognition using tensorflow' s object detection API with a new benchmark dataset," 2020 International Conference on Electrical Engineering (ICEE) [Preprint]. Available at: https://doi.org/10.1109/icee49691.2020.9249914. (Accessed: April 18, 2023).

37 Sichkar, V.N. and Kolyubin, S.A. (2020) "Real time detection and classification of traffic signs based on Yolo Version 3 algorithm," Scientific and Technical Journal of Information Technologies, Mechanics and Optics, 20(3), pp. 418–424. Available at: https://doi.org/10.17586/2226-1494-2020-20-3-418-424. (Accessed: April 18, 2023).

38 Lu, E.H.-C. et al. (2022) "A hierarchical approach for traffic sign recognition based on shape detection and image classification," Sensors, 22(13), p. 4768. Available at: https://doi.org/10.3390/s22134768. (Accessed: April 18, 2023).

39 Manawadu, Mayura & Wijenayake, Udaya. (2021). Voice-Assisted Real-Time Traffic Sign Recognition System Using Convolutional Neural Network.

40 Zhu, Y. et al. (2016) "Traffic sign detection and recognition using fully convolutional network guided proposals," Neurocomputing, 214, pp. 758–766. Available at: https://doi.org/10.1016/j.neucom.2016.07.009. (Accessed: April 18, 2023).

41 Zhu, Y. and Yan, W.Q. (2022) "Traffic sign recognition based on Deep Learning," Multimedia Tools and Applications, 81(13), pp. 17779–17791. Available at: https://doi.org/10.1007/s11042-022-12163-0. (Accessed: April 18, 2023).

42 Buolamwini, J., & Gebru, T. (2018). Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. FAT.

43 Dept. Transport (2015) Know your traffic signs. London: Stationery Office.

44 Pere, C. (2020) What are loss functions?, Medium. Towards Data Science. Available at: https://towardsdatascience.com/what-is-loss-function-1e2605aeb904#:~:text=Range%20of%20values%20for%20this,0.05%3A%20In%20a%20good%20way. (Accessed: April 17, 2023).

45 Ertler, C. et al. (2020) "The mapillary traffic sign dataset for detection and classification on a global scale," Computer Vision – ECCV 2020, pp. 68–84. Available at: https://doi.org/10.1007/978-3-030-58592-1_5. (Accessed: April 18, 2023).