Deep Learning in Webots

Izabela Zelek and Ayomide Otinwa

Abstract

This project explores the usage of object recognition in determining behaviours in autonomous vehicles based on detected road signs. MobileNetV2 and GoogleNet were chosen as the object recognition models due to their efficiency on power-sensitive devices and their lightweight build. Both were trained on images of 17 types of road signs and their performance was compared. MobileNetV2 achieved a better accuracy of 86%, in comparison to GoogleNet's 70%. The model was implemented in a simulated environment in Webots with a vehicle modelled after the Zeus car which included a camera, a sonar sensor and infrared sensors. The vehicle used the left infrared sensor to detect incoming road signs, before starting the MobileNetV2 to recognise the sign type. Behaviours were implemented for each road sign, including: speeding up, slowing down, stopping, and turning. The sonar sensor was used to halt the vehicle when an obstacle was detected. Despite the 86% accuracy achieved, the model struggled in identifying signs when travelling at high speeds due to blurring of images. The project demonstrated sensor-based autonomous behaviour, as well as provided recommendations for future improvements of the system, focusing on better training data.

Introduction

In today's modern world, researchers are searching for ways to improve human lives by delegating menial or unpleasant tasks to artificial intelligence. With around 20% of the Irish population over 18 years old not owning a driving license as of 2024 (The AA, 2024), the addition of artificial intelligence to cars to create automated vehicles may improve the accessibility to people unable to drive. This, however requires a flawless system in place to avoid casualties and ensure safety. An important aspect to consider when creating automated vehicles is the ability to obey road rules, specifically to follow the road signs which can determine speed, driving conditions and when to stop. By using object recognition, a model can be created to detect which sign the car is passing and establish how the vehicle should behave.

Objective

The objective of this project is to develop an efficient object recognition model to recognize passing signs. Each sign will have a specific behaviour attached to it, which will allow a vehicle implemented in Webots to traverse the road while obeying road safety rules.

Development

For the entirety of this project, a Waterfall Framework was used. The requirements for this project were thoroughly discussed and determined before any programming was completed. The project was completed in various stages: Planning, Dataset Creation, Model Creation, Webots Implementation, and Testing. Each phase was completed in its entirety before the next phase commenced. Model Creation was performed individually to allow for testing of different

models, MobileNetV2 and GoogleNet. The other phases were completed in cooperation by using Pair Programming.

Requirements Specification for the ICPS

The car simulated in Webots was modelled after the Sunfounder Zeus Car Kit.

Power Requirements

The project requires constant sensor use to detect incoming road signs and obstacles, as well as capacity to run an object recognition model to affect the behaviour of the vehicle. The estimated power life of the Zeus car is 90 minutes, which can rapidly decrease if power-heavy programs are being run (SunFounder, 2023). In order to preserve the battery life in the case of real-world implementation, an Infrared sensor is used to detect incoming objects at the side of the road. Upon detection, the vehicle runs the object recognition model to detect the road sign. This ensures that the object recognition model is not run constantly which can be detrimental to the battery life. The model used for object recognition is based on MobileNetV2 which is lightweight and efficient on edge devices. It was chosen due to it requiring less computational power without sacrificing accuracy.

The system is built around the driving regulations in Ireland, which means the vehicle drives in the left lane. For this reason, only the left Infrared sensor is used to detect incoming road signs. The right Infrared is never used or triggered which allows for saved power.

Communication

There is no external communication used during the simulation, however if implemented on the Zeus car, the user has the ability to connect remotely through Wi-Fi to control the vehicle. For the purpose of this project, this function is unused as the vehicle controls itself through object recognition of road signs and designated behaviours based on the signs detected. However, constant internal communication is required to start behaviours when a sensor has been triggered.

Processing

As mentioned MobileNetV2 is used for object recognition to detect road signs. MobileNet is lightweight and efficient on edge devices which ensures the model does not affect the processing power. To aid in efficient processing, the MobileNet model is only ran when an object is within the sensor's defined limits. This process is also performed on a separate thread to ensure the vehicle does not halt while performing predictions.

Safety

In order to ensure safety, a sonar sensor placed at the front of the vehicle is constantly checking for obstacles. The vehicle is programmed to halt whenever an object is detected within a 5m range of the sensor.

In the case the vehicle fails to recognise a sign, it will continue moving at the previous speed. This ensures the vehicle does not suddenly halt its movement which may cause accidents. However, the drawback of this is that the new behaviour may fail to follow the safety rules implemented for that specific stretch of road.

The model was trained on bright and clear, as well as dark and blurry images, however the Webots implementation is performed in daylight conditions with a clear view of each road sign. The performance of the model may suffer if there are improper lighting conditions or if the vehicle is moving too fast to be able to take clear images.

The simulation is modelled after the Irish traffic system, meaning it is expecting road signs to appear on the left side of the road. The camera is focused on the left side of the image to make predictions, which allows the program to only consider the safety rules of the lane it is in, ensuring the road signs for the right lane are not detected. In the case a simulation is needed for the right lane, minor adjustments would need to be performed to use the right infrared instead, as well as to change the camera focus to the right side. Performance of multi-lane roads has not been tested and so is unrecommended.

No user data is processed or saved. This ensures compliance with Data Safety regulations while also preventing possible hackers from having access to routes and schedules of users.

Literature Review

Traffic Sign Recognition has become an important component for autonomous vehicles and intelligent driver assistance systems. Recent developments in deep learning along with computer vision have improved the ability to detect and classify traffic road signs in real-world environments. These methods offer better accuracy, scalability and robustness in different environmental conditions.

Huang et al. (2017) proposed a traffic sign detection system that utilizes color-based filtering. Their architecture first filters irrelevant regions using the colour and geometric heuristics and using the CNN model is used to classify the left-over areas. This method is closely aligned with the preprocessing approach used in our work, where the images captured from the left half of the camera's field of view were cropped, resized, and normalized before being passed to the deep learning model for classification. By focusing the cameras viewpoint at the left side of the road where traffic signs most often exist, the two systems show a significant improvement in classification accuracy, especially in high-density environments.

In the case of real-world performance, the work of Saouli et al. (2021) showed the deployment of a lightweight deep learning framework using Tiny-YOLOv3 to support traffic sign recognition. The system balanced the need for accuracy and edge AI hardware along with k-means clustering. They noted that motion blur, insufficient lighting, and small object size had a huge effect on the performance. These issues show the need to tune the sensing and preprocessing components to achieve consistent detection across different conditions.

To reduce the performance cost of real-time detection, Abri et al. (2020) proposed a multi-threaded object detection using YOLOv3. This approach used threading to distinguish image processing from inference operations, this minimized the delay and maximized the use of resources. The idea was implemented in this system to enable simultaneous image classification while the vehicle moves. This setup enabled the camera to run without requiring the vehicle to be stopped while predicting.

These studies provided valuable insights into the system design, model choice and implementation strategy. By incorporating their results into our work, we were able to successfully create a traffic sign recognition system with improved adaptability in a controlled environment.

Model Creation

Dataset Creation

For the purpose of this project, seventeen classes were chosen. These classes were as follows: 5 km/hr, 15 km/hr, 20 km/hr, 30 km/hr, 40 km/hr, 50 km/hr, 60 km/hr, 70 km/hr, 80 km/hr, 90 km/hr, 100 km/hr, 120 km/hr, Crosswalk, Left Turn, Right Turn, and Stop. The dataset was created by combining three existing Kaggle Datasets. The first dataset (Tuan_Ai, 2024) contained a wide range of road signs, however the number of samples for our desired classes was low. The second dataset contained over 4.9 thousand images with all our desired classes (Karimi Darabi, 2024), however once again the sample size was too low. The final dataset contained over 800 images, however these images were all in one folder and not separated into their own sample folders (Maranhão, 2020). These three dataset were then manually pre-processed to remove unrelated or blurry images, and then added to separate folder per class type.

Dataset Preprocessing

In order to standardize the images, four processes were coded to fix existing issues with the dataset. The first issue was the small amount of samples in the ‘Left’ turn sample folder. Considering the ‘Left’ and ‘Right’ turn signs are identical, apart from the orientation, images were taken from the ‘Right’ folder, flipped and transferred to the ‘Left’ folder. The second issue was the overall small sample size in the majority of folders. Existing images from each folder were taken and augmented in two ways: zoom, to allow for better detection in both close-up and faraway objects, and shear, to allow for better detection at different angles. These new images were saved in a temporary folder for manual inspection. Once all images were inspected and deemed appropriate, the dataset was then balanced by determining a base sample number of 500 per folder, and the excess images were removed at random. The final issue was the varying naming conventions in all images due to the combination of three datasets, as well as varying image formats, ranging from ‘png’ to ‘jpg’. Each image was renamed to a determined naming convention of the folder name followed by a number starting at one and increasing by one. The images were then all converted to pngs.

Image Preprocessing

All images were standardized to a 400x400 size to avoid bias. The pixel values of the images were normalized to a 0-1 range to allow the model to converge faster. All images were split into 3 categories: Training, with 80% of the dataset, Testing, with 10% of the dataset, and Validation, with 10% of the dataset. A random seed of 37 was used to allow for reproducibility, with Validation having a seed of 42 to allow for different samples than in Testing. The Validation and Testing data had no augmentations, while the Training data was randomly rotated up to 30 degrees, and brightened at random.

Methodology

MobileNetV2 was chosen due to its simplicity and efficiency on edge devices, which allows for fast responses. It was loaded as the first layer, pretrained on ImageNet to speed up training time, and frozen so that the weights don't update during training. The final layers were removed, to be trained later, which left the base model for feature extraction. GlobalAveragePooling2D was then added to reduce spatial dimensions and aid in reducing overfitting, followed by Batch Normalization to stabilise the training. A Dense Layer with 128 neurons was then used to learn the high-level patterns, followed by a Dropout layer of 40% to again aid in reducing overfitting. A final Dense layer was then used to output probabilities for the 17 classes. Three callback methods were used to aid in training. Model Checkpoint was used to monitor the validation accuracy and save the best found weights throughout training. A Learning Rate Scheduler was used to automatically reduce the learning rate when the model's performance stopped improving, using validation loss as the monitored variable. Finally, Early Stopping was used to stop training once the model stops improving for 10 epochs, which allows for shorter training time. The model was compiled using Adam with a 0.001 learning rate at first. Accuracy was monitored throughout training and categorical cross entropy was used as the loss function. The model was trained on the Training dataset and evaluated on the Validation dataset. The Testing dataset was then used to calculate the evaluation metrics. For further fine-tuning, the final twenty layers on MobileNet was unfrozen to allow for the task-specific layers to be refined. The model was then recompiled with a smaller learning rate. Pickle was used to save the training history due to Jupyter Notebook removing it once the notebook is closed. The best weights were then loaded in and the model was saved to a h5 file.

Additional Model Testing

For the sake of comparison, a GoogleNet model was also implemented. GoogleNet is another lightweight model which works efficiently on devices with power restrictions. However, there were issues with implementing this model due to its specifications. GoogleNet requires a 28x28 image size as input. This was not ideal for the purpose of this project as the dataset consisted of images of around 400x400 pixels. Decreasing the sizes of these images would lose a large amount of important features which would affect the accuracy. Considering the project deals with road safety, accuracy and a high-quality training dataset were needed.

Another requirement of GoogleNet is grayscale images, however, it was determined that the colour of the road signs is an important aspect to consider when recognising road signs and so grayscale images were not ideal.

However, even with those specifications, this model achieved a 70% accuracy. This was an acceptable performance, although with room for further improvement. As MobileNetV2 achieved 86% accuracy, this was the model chosen for the project.

Evaluation Metrics

The model was evaluated with four metrics: Accuracy, Precision, Recall and F1. Precision calculated the amount of True Positives out of all positives, Recall calculated the amount of Actual Positives out of Actual Positives and False Negatives, and F1 balanced them out to

show any imbalance. A confusion matrix with a heat map was then used to visualise the performance of the model across all classes.

Results

The model achieved an 86% accuracy, further refining proved unsuccessful as the model always plateaued on 86%. A larger dataset with more samples may be required to further improve the accuracy. A more thorough investigation may be needed to determine whether the augmentations applied are helping or hurting the results. As seen in Fig. 1, the model particularly struggled on road signs that looked too similar, such as 30, 50, 60, 80 and 90 with precision scores all under 0.75. It is our assumption that the model struggles with the visual similarity of the round curves on the numbers and misinterprets them. It performed the worst on 30, however this can be attributed to the majority of the samples being dark and/or blurry. The low precision and low recall numbers of 0.56 and 0.66, respectively, shows that there are high numbers of False Positives and False Negatives. The other class the model struggles on is the 80. Similarly to 30, the majority of images in this class are dark and/or blurry which may be the reason for the low scores. The low recall score shows that the model does not quite understand how to identify the 80 category and only correctly identified 22% of the actual images and achieved a high number of False Negatives. For further work, the images in both class folder would have to be replaced with better suited ones. The model achieved perfect scores on five of the classes: 5, 15, Crosswalk, Left and Stop. It misinterpreted one Left as a Right, however the performance of the model on Left remains satisfactory. Nine out of the seventeen classes achieved score over 0.9, which shows a good performance however further work should be done on the remaining eight classes,

Accuracy: 0.86
Precision: 0.86
Recall: 0.86
F1-Score: 0.85

Classification Report:

	precision	recall	f1-score	support
100	0.79	0.76	0.78	50
110	0.91	0.98	0.94	50
120	0.94	0.88	0.91	50
15	1.00	1.00	1.00	50
20	0.89	0.94	0.91	50
30	0.56	0.66	0.61	50
40	0.94	0.92	0.93	50
5	1.00	1.00	1.00	50
50	0.74	0.62	0.67	50
60	0.70	0.90	0.79	50
70	0.83	0.90	0.87	50
80	0.65	0.22	0.33	50
90	0.73	0.94	0.82	50
CrossWalk	1.00	1.00	1.00	50
Left	1.00	0.98	0.99	50
Right	0.98	1.00	0.99	50
Stop	1.00	0.98	0.99	50
accuracy			0.86	850
macro avg	0.86	0.86	0.85	850
weighted avg	0.86	0.86	0.85	850

Figure 1: Precision, Recall and F1 for each class

Further observations can be seen on the confusion matrix shown in Fig. 2. The overall representation shows a good performance for the majority of the classes, with a clear diagonal line. Visually, the worst performing class is the 80 with more incorrect predictions than correct. Even though numerically, 30 performed the worst, visually 30 achieved a good score. The vast

majority of classes achieved a strong result in 40's and up. Better representation in the dataset is needed to improve the performance of the weaker classes.

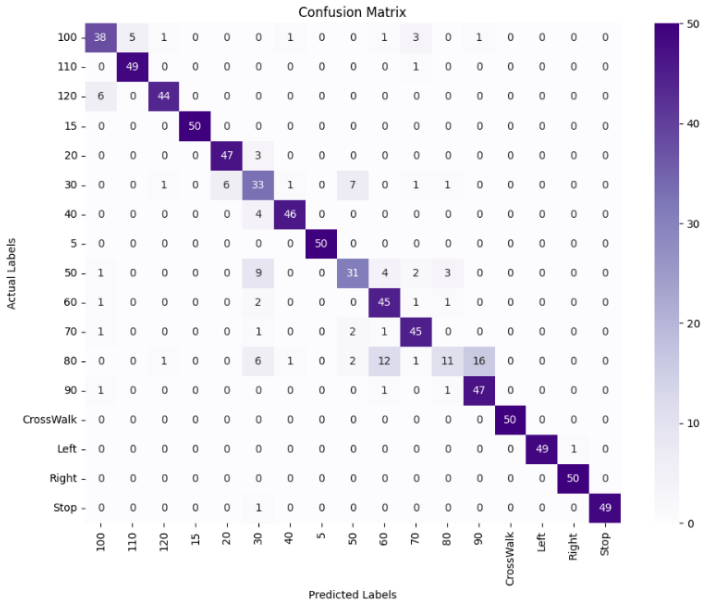

Figure 2: Confusion matrix displaying performance of each class

Webots System Design and Implementation

Vehicle Modelling

The Zeus car was recreated in Webots using the same real-world dimensions which includes four 6 centimetre wheels and a dimension of 6.65 m in length, 6.50 m in width, and 3.70 m in height. All the kinematic parameters such as the wheel radius match the physical vehicle to make sure the behaviours of the vehicle in this controlled environment would match appropriately with the Zeus car in the real world.

Sensor Configuration

The vehicle has three sensor types that mirror the Interface and integrate onboard sensors. A forward-facing RGB camera is mounted at roof height, two infrared distance sensors, one on each side of the car facing diagonally to measure proximity to roadside objects and a single distance sensor in the front. The camera was used for capturing traffic signs and was configured to focus on the left half of its field of view so as to avoid obstruction from the car's structure, such as the left wheel arch and ensuring it processes the signs due to them being on that half of the view, thus ensuring a clear view of roadside signs as seen in Fig 3. The left infrared sensor was used as the proximity trigger, which enabled the system to determine the optimal point at which a road sign should be detected. Meanwhile, the front sonar sensor was specifically tasked with detecting obstacles when the car approached crosswalk signs, ensuring a realistic and reactive stopping mechanism.

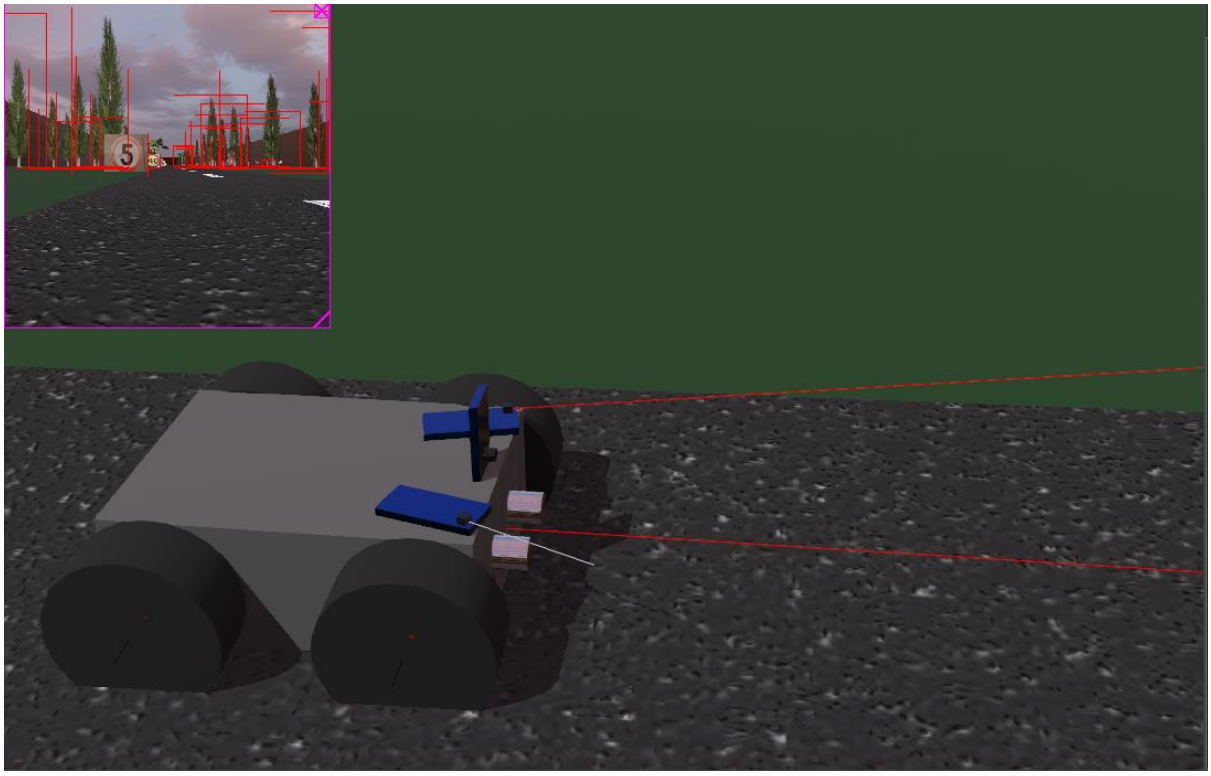

Figure 3. Design of Zeus car in Webots Environment

Traffic Signals

Traffic signs in Fig. 4 were created within the Webots environment as solid objects with applied visual textures sourced via URLs. Each sign had a bounding object attached to it to allow for accurate recognition by the camera sensor and detection by the Infrared sensor. These signs were placed along a custom-designed roadway to simulate a realistic driving experience. This setup enabled the car to detect and respond to various traffic instructions in real-time. Signs such as speed limits, crosswalks, left and right turns, and stop commands were implemented. The bounding objects ensured that each sign could be captured within the camera frame and subsequently classified by the deep learning model.

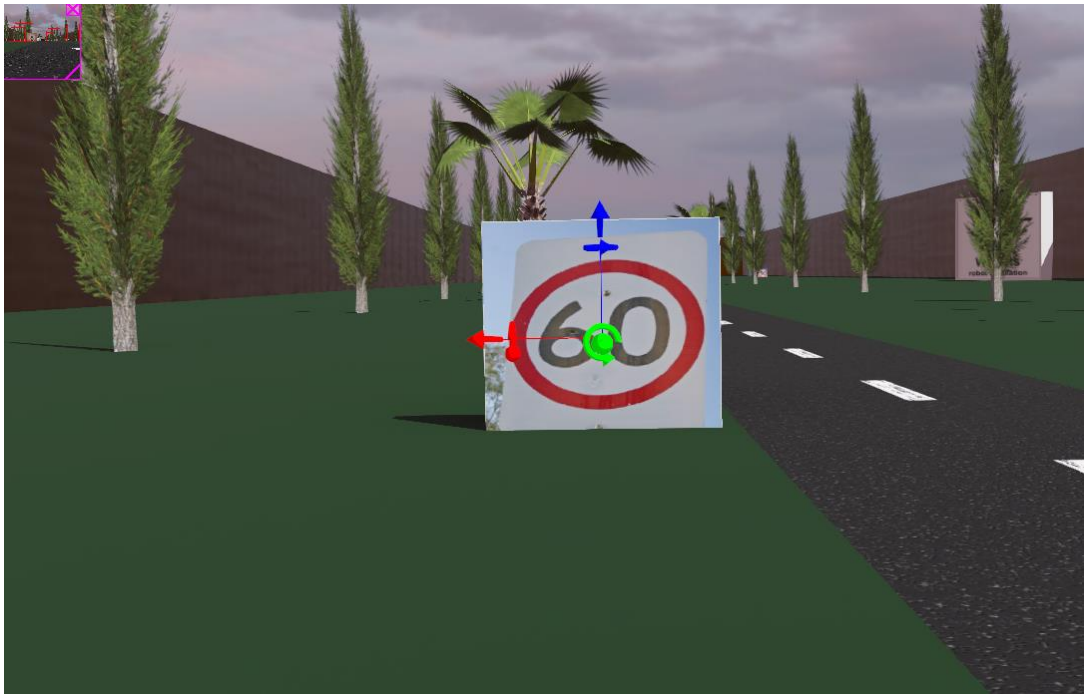

Figure 4. Traffic signal

Model Integration with Webots

The integration of the machine learning model was the main part of this project. The pre-trained and fine-tuned MobileNetV2 model was loaded at the start of the simulation using TensorFlow which is designed to recognize seventeen different classes of road signs. Once the left infrared sensor detected a distance below 5 m to the traffic signal in Fig. 5, a frame was captured from the left half of the camera's view. The image was cropped then resized to 400×400 pixels, normalized, and passed to the model for prediction. This process ensured consistency in image preprocessing between training and inference.

To make sure that the car maintained a smooth transition during prediction, we implemented threading to the code. A separate thread was implemented in the background which was used to handle the image classification so that the car could continue driving while the model processed the image simultaneously. This prevented the lags and halting that occurred when the system had to wait for the prediction before acting accordingly, which improved the real-world applicability and fluidity of the simulation.

Upon prediction, the output class was matched against predefined behaviour rules for the speeds. Speed limit signs (e.g., "5", "15", ..., "120") adjusted the car's motor velocities proportionally. Turn signs activated an odometry-based function, which precisely rotated the car using differential wheel speeds based on the desired angle. In the case of the stop and crosswalk signs, the car came to a full stop and for the crosswalk, the vehicle remained stationary until the sonar sensor confirmed the absence of an obstacle. This behaviour mimicked an autonomous car logic and provided an effective demonstration of AI-driven response control. The logic also included safeguards to prevent repeated stops and crosswalk detection.

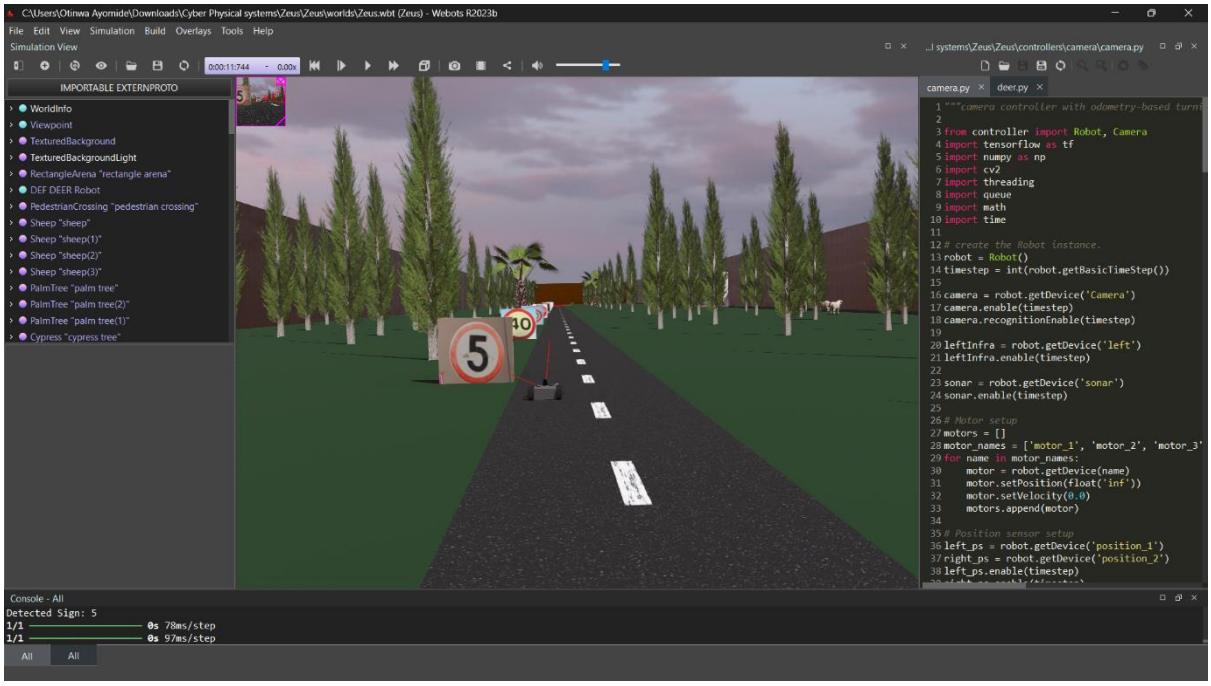

Figure 5. Traffic Sign Detection

Performance Testing

Calibration of Detection Range

Testing began by determining the optimal range at which the camera should capture and classify the traffic signs. Several trials were conducted to observe the left infrared sensor's reading for the point of successful prediction. The results showed that the most consistent predictions occurred when the object was detected at distances under 5m. This made us choose the decision to use the left IR sensor's reading as a threshold to trigger the camera's preprocessing and classification sequence.

Real-Time Model Validation

The system was tested along a straight road with multiple traffic signs positioned at various intervals. During the simulations, the vehicle successfully adjusted its speed or performed a turn based on the traffic sign detected in 80% of cases. The few instances of incorrect behaviour were caused by similar-looking signs such as 90, 100 and 110 due to higher speeds where motion blur affected image clarity. This calls attention to the importance of balancing driving speed with detection accuracy in real-time systems.

Crosswalk and Stop Functionality Testing

The crosswalk and stop sign were given specific functionalities for evaluation. For the crosswalk sign, the car came to a halt and used its front sonar sensor to detect potential obstacles. In test scenarios where an object was placed directly in front of the car, it only resumed motion after the sonar confirmed the space ahead was clear. This demonstrated not only the effectiveness of the classification model but also the integration of reactive sensor-

based decision-making. Similarly, the stop sign triggered an immediate stop followed by a timed wait before resuming prior speed as seen in Fig. 6.

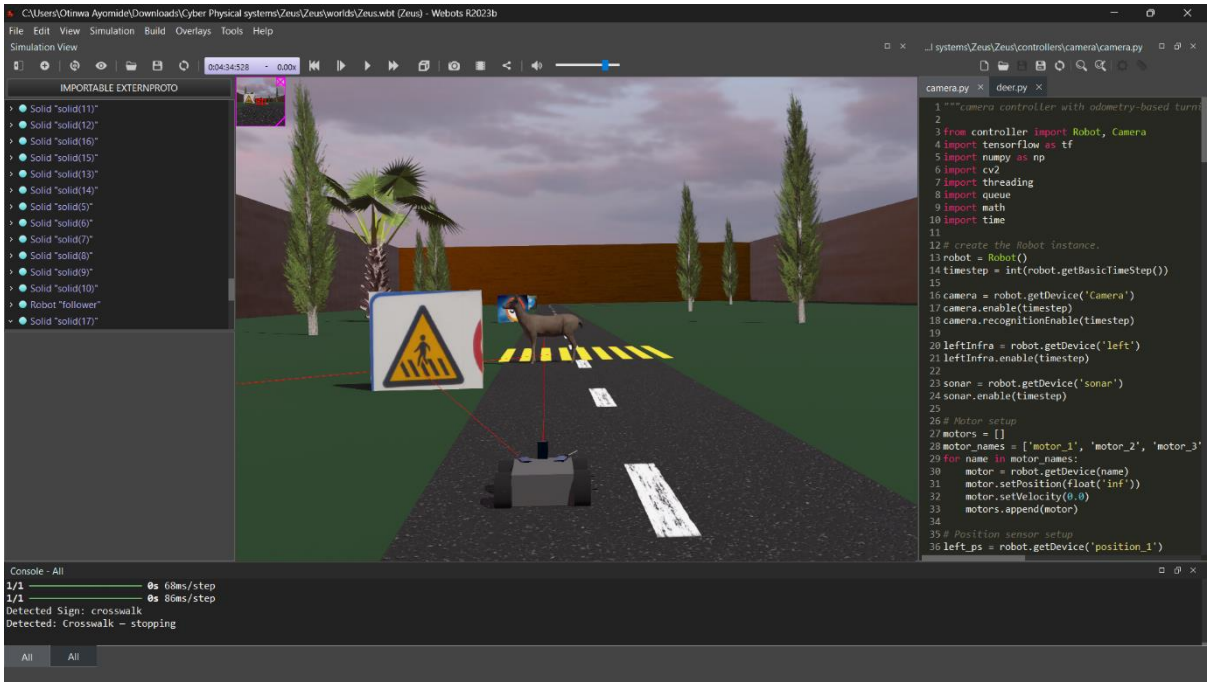

Figure 6. Crosswalk detection in action

Observed Limitations and Considerations

While the system performed well overall, certain limitations were observed during testing. One of the issues was the inconsistency in predictions made by the model. In a few instances, the same traffic sign predicted different classification results after multiple runs under similar conditions. This infrequent action affected the reliability of the system in controlled environments. Additionally, at higher speeds the camera sometimes failed to capture clear images of road signs due to motion blur which caused us to reduce the maximum speeds for the signs.

Conclusion

This project demonstrated the use of object recognition in detecting road signs to allow for autonomous behaviours in vehicles. A MobileNetV2 model was chosen due to its efficiency and lightweight build. This model was trained on 17 road signs which were manually inspected to ensure good data. This dataset was also pruned and augmented to allow for non-biased data and to improve performance on unseen data. The model achieved 86% accuracy, though the accuracy suffered on the “30” and “80” classes. Further work is needed, particularly on a clearer dataset, to further improve the performance.

Performance testing of the Webots implementation showed reliable behaviour in the majority of cases, however signs with similar shape and design were shown to be incorrectly identified when the vehicle travelled at higher speeds.

In conclusion, this system displays realistic, sensor-based vehicle movement based on object recognition, however further work is needed to solve incorrect speed-based predictions and to further improve accuracy.

References

Abri, R., Abri, S., Yarici, A. & Çetin, S., 2020. *Multi-Thread Approach to Object Detection Using YOLOv3.* Kitakyushu, Japan, Vision & Pattern Recognition (icIVPR), pp. 1-6.

Huang, S.-C., Lin, H.-Y. & Chang, C.-C., 2017. *An in-car camera system for traffic sign detection and recognition.* Otsu, Japan, 2017 Joint 17th World Congress of International Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems (IFSA-SCIS), pp. 1-6.

Karimi Darabi, P., 2024. *Traffic Signs Detection.* [Online]
Available at: https://www.kaggle.com/datasets/pkdarabi/cardetection/data
[Accessed 20 April 2025].

Maranhão, A., 2020. *Road Sign Detection.* [Online]
Available at: https://www.kaggle.com/datasets/andrewmvd/road-sign-detection/data
[Accessed 20 April 2025].

Saouli, A., Margae, S. E., Aroussi, M. E. & Fakhri, Y., 2021. Real-Time Traffic Sign Recognition based AI Edge. *International Journal of Computer Science and Information Security (IJCSIS),*, 19(7).

SunFounder, 2023. *SunFounder Zeus Car Smart Car Kit for Arduino UNO R3.* [Online]
Available at: https://www.sunfounder.com/products/sunfounder-zeus-car-robot-car-kit-compatible-with-arduino-uno-r3?srsltid=AfmBOoougZptB3zvRIo8y7SOIiBCwaAeJir49gfxC-ZiXNUgESf7jQYT
[Accessed 25 April 2025].

The AA, 2024. *Survey by AA Ireland Indicates Irish Drivers are Keeping Their Cars Longer.* [Online]
Available at: https://www.theaa.ie/blog/survey-indicate-irish-dirvers-keeping-cars-longer/#:~:text=Combine%20this%20with%20roughly%2080,of%20life%20in%20today's%20Ireland.
[Accessed 20 April 2025].

Tuan_Ai, 2024. *Traffic-Signs-Dataset.* [Online]
Available at: https://www.kaggle.com/datasets/tuanai/traffic-signs-dataset/data
[Accessed 20 April 2025].