ES327 Project

Contents

Figures & Tables

Executive Summary

Misclassification and failure to detect objects in harsh weather and low-light conditions are key factors contributing to AV crashes. This project investigates the causes of these errors in vision models for autonomous vehicles (AVs) and explores ways to improve object detection accuracy. The goal is to enhance the model's ability to detect pedestrians, vehicles, and other road users, particularly in challenging environments.

The approach involved selecting a suitable model, training it using the nuImages dataset with diverse environmental conditions, and optimising its parameters. A YOLOv10L model was chosen for its balance between detection accuracy and inference speed. Adjustments were made to loss function weights to improve recall, particularly for smaller objects like pedestrians. Results showed an increase in recall from 0.545 to 0.556, improving object detection at the cost of reduced precision from 0.850 to 0.840.

Future work includes expanding the dataset for better generalisation, refining loss weighting strategies, and testing on AV-specific hardware. This project contributes to improving AV perception systems, reducing misclassification errors, and improving road safety.

1 Introduction

The evolution of autonomous vehicles (AVs) has brought a transformative era in transportation, promising enhanced safety, efficiency, and accessibility. At the core of this innovation lies the integration of machine learning and advanced vision systems that enable vehicles to perceive, interpret, and respond to their environment in real-time. Recent progress in computer vision, particularly in object detection, has significantly improved AVs' ability to recognise pedestrians, vehicles, and other road users. However, challenges remain in complex environments with varying lighting and weather conditions.

This project, undertaken as part of the ES327 module, focuses on the development and analysis of a vision model for autonomous vehicles. The aim is to bridge the gap between advancements in computer vision and their practical implementation in autonomous systems.

This project addresses challenges in object detection by using advanced techniques in computer vision and machine learning, to contribute to the broader goal of creating safer and more reliable AVs.

1.1 Aims

The primary aim of this project is to investigate the causes of errors in existing vision models for autonomous vehicles and modify the parameters of an existing model to address these issues. By gaining an understanding of the architecture and key parameters this project will adapt the model to better suit the dataset and improve its performance.

1.2 Objectives

The following objectives outline the specific tasks and milestones to guide the project towards its goal of improving detection in autonomous vehicles. These objectives aim to develop accurate and efficient vision models for real-time application in AVs, with a focus on addressing common sources of error in AV crashes. Each step contributes to refining the model's performance, from initial development to testing and evaluation.

- **Literature Review**: Conduct a comprehensive review of autonomous vehicle safety, the evolution of neural networks, and object detection models.

- **Model Development**: Develop a Vision model capable of accurately detecting and classifying pedestrians, cars, buses, bicycles, trucks and motorcycles in driving scenarios.

- **Testing and Evaluation:** Evaluate the model using performance metrics such as F1 score, recall, class accuracy, object accuracy, mean average precision (mAP) and inference time to assess its effectiveness in real-time applications.

- **Parameter Optimisation:** Modify model parameters and architecture using testing and evaluation results to justify adjustments.

- **Error Analysis:** Analyse the results to identify remaining limitations and refine the model further as needed. Aswell as exploring trade-offs between processing time and accuracy.

- **Documentation and Reporting:** Compile documentation detailing the methodology, results, and insights gained during the project. Conclude with discussion of the project's findings and their implications.

2 Literature Review

The literature review provides an overview of key concepts and advancements of vision models relevant to AVs. It highlights the significance of AVs, the development of vision models, and the role of convolutional neural networks (CNNs) in enabling object detection. Additionally, it explores various object detection models, discussing different architectures, their performance, and the trade-offs in the context of AV perception.

2.1 Significance of autonomous vehicles

Accurate detection is essential for improving the safety of autonomous vehicles (AVs), and this project focuses on that capability using vision-based models. Autonomous vehicles must reliably identify and classify objects such as cars, trucks, cyclists, and pedestrians to avoid accidents and ensure safe navigation (Azam et al., 2020). California is often used for AV safety studies because it permits extensive AV testing, providing a large dataset for analysis. The statistics show that AVs experience significantly more crashes per 1,000 vehicles motor vehicles as well as more crashes per million vehicle miles travelled.

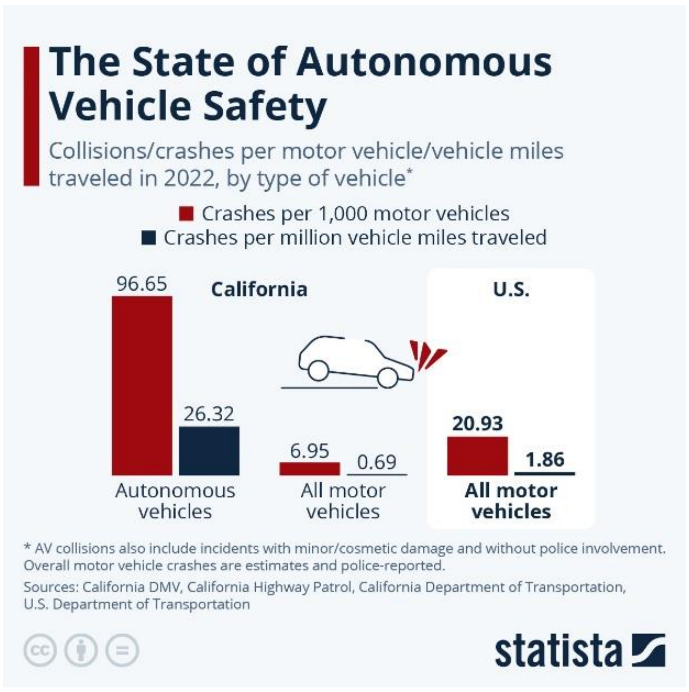


Figure 1 AV Crash Rates vs. Motor Vehicle Crash Rates (Zandt, 2024)

Furthermore, before AVs can be considered ready for mass marketing, real-world testing in diverse driving scenarios is necessary. This will provide valuable information in situations where errors occur, leading to dangerous situations, such as collisions or unsafe manoeuvres. In 2022 the California Department of Motor Vehicles' (DMV) tested 1,522 AVs (Zandt, 2024). The data highlighted significantly higher crash rate for AVs compared to motor vehicles. In California, AVs experienced 96.65 crashes per 1,000 vehicles, whereas all motor vehicles had a much lower rate of 6.95 crashes per 1,000 vehicles. These figures suggest that while AV technology is advancing, it still faces challenges.

To further investigate the causes of these accidents and explore potential solutions, a good starting point would be to examine the types of accidents involving AVs. A paper analysed data from the Californian Department of Motor Vehicles to assess factors of AV crashes (Kurse et al., 2025). Among the top ten factors identified were decision making errors and environmental decisions. To an extent these are factors that can be minimised with more accurate and robust detection technology.

However, a separate study analysing AV crash data in California from 2017 to 2021 found that in 93.8% of AV-involved crashes, the other vehicle's driver was at fault (Houseal et al., 2022). While this suggests that human error plays a major role, improving AV performance is still crucial. Instead of focusing on external factors beyond control, it is more effective to address aspects that can be influenced, such as **environmental awareness**. The study identified time of day, lighting conditions, and intersection control as factors that impact an AV's ability to interpret its surroundings.

The problem this project aims to solve is reducing misclassification in challenging environments and in unique driving scenarios, through a diverse dataset with varying light and weather conditions and applying data augmentations. The results will have a detailed analysis of object by class to find any trends in evaluation metrics. Additionally, the loss function weights will be modified based on observations from results, this is expected to improve detection accuracy.

The current state of research indicates that existing automotive safety methods may not sufficiently address the unique challenges encountered by AVs (Collin et al., 2020). By improving the object classification capabilities of AVs, this project contributes to improving road safety, ensuring that AVs can better detect and respond to potential hazards.

2.2 Vision models

This section explores the development of vision models, beginning with their origins in neural networks (NNs) and deep learning, progressing into Convolutional Neural Networks (CNNs) designed for computer vision tasks, and concluding with a comparison of object detection models suitable for this project.

Understanding progress in this field, particularly object detection architectures, is important for selecting and effective approach. Furthermore, training neural networks can be highly resource intensive, requiring significant computational power, time, and energy. Therefore, this section aims to provide a foundation for making informed decisions regarding model architecture selection, parameter tuning, and performance analysis. The goal is to maximize the model's performance in object detection tasks while minimizing resource consumption, (cost, training time, and energy expenditure) ensuring the development of an efficient and effective system for autonomous driving.

Early Neural Networks

Perceptrons are considered the first type of neural network, developed by Frank Rosenblatt in 1958 as part of his research in machine learning and cognitive systems (Rosenblatt, 1958). They operate by receiving several binary inputs and producing a single binary output. Each input is associated with an adjustable weight and a bias term is added to these inputs.

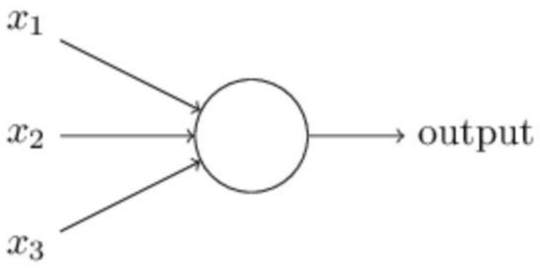


Figure 2 Basic diagram of a perceptron (Nielsen, 2015)

This performed well for linearly separable problems, making them suitable for binary classification. To improve upon the limitations of the single layer perceptron model, a more flexible approach was necessary, where small changes in weights would result in correspondingly small changes in the output.

The sigmoid activation function was the one of the first non-linear activation functions. Its outputs were continuous values between 0 and 1, allowing neurons to represent probabilities rather than binary decisions (Popescu et al., 2009). As deep learning evolved, the Rectified Linear Unit (ReLU) became a more popular alternative. It sets all negative values to zero and is favoured for its efficiency and ability to mitigate the vanishing gradient problem (Agarap, 2018).

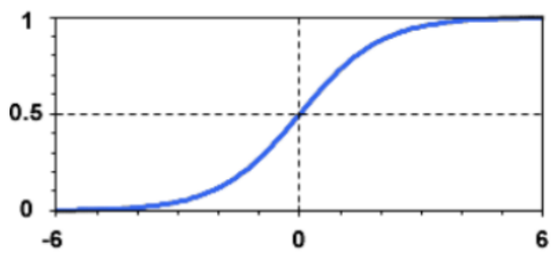


Figure 3 Sigmoid graph (Popescu et al., 2009)

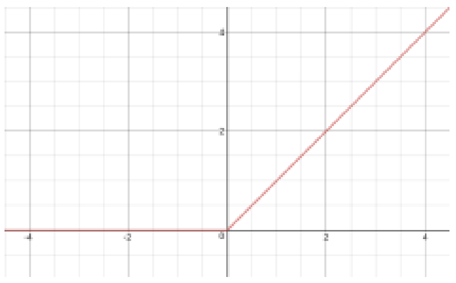


Figure 4 ReLU (Agarap, 2018)

In addition to this, the development of the multilayer perceptron (MLP) organising neurons into multiple layers comprising an input layer, one or more hidden layers, and an output layer—MLPs introduced hierarchical feature extraction (*Ramchoun et al., 2016*). More layers enabled the model to learn better by breaking down input data into features, while non-linear activation functions allowed for non-linear learning, leading to more complex pattern recognition.

Backpropagation

To further improve neural network training, a method was needed to distribute error to each layer, allowing the model to adjust weights effectively. This challenge led to the development of backpropagation, popularised for neural network training in 1986 by Rumelhart, Hinton, and Williams in 1986 (Rumelhart et al., 1986). Backpropagation allowed errors to propagate backward through the layers of MLPs.

During the forward pass, inputs are fed through the network layers (e.g. frame of driving scenario) to produce outputs that are evaluated against the target outputs using the cost function (used to quantify how far off model is to correct answers). The resulting error is then used in backpropagation (backward pass), where the network adjusts its weights to reduce this error (Fan & Zhao, 2019).

To determine the adjustments to weights that most effectively reduces the cost function it is important to evaluate how sensitive C is to changes in weight. Ideally, we would like to directly compute this value, but this is not possible because the cost function depends in the final layer output, while each weight influences the cost indirectly through multiple layers. Instead, the chain rule is applied to express this dependency, allowing gradients to be computed layer by layer. (Nielsen, 2015). The image illustrates the connections between neurons across layers, highlighting how backpropagation relies on the chain rule to adjust weights by computing gradients layer by layer.

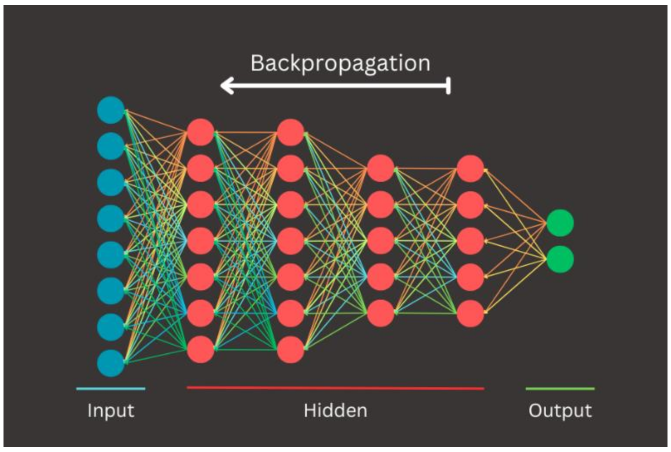


Figure 5 Neuron connection in an MLP (Causevic, 2024)

This process is repeated for each neuron. In practice computations are optimised using vectorised operations to simultaneously calculate multiple gradients, enabling neural networks to learn more complex patterns and relationships within data (Fan & Zhao, 2019).

Gradient Descent

Once gradients are determined through backpropagation, **gradient descent** is applied to update weights, minimizing the cost function by adjusting weights in the direction that reduces error. (Ruder, 2017).

Rather than updating the model parameters for each individual training example (Stochastic gradient descent) a widely used approach is mini batch gradient descent. Gradients are averaged over a small subset of the data (a mini batch) before updating (Gultekin et al., 2020). This approach leverages the hardware of GPUs and provides stable updates.

Deep learning relies on key mechanisms: **non-linear activation functions** enable complex decision boundaries, **multiple layers** in neural networks allow hierarchical feature extraction, and **backpropagation** adjusts weights efficiently using gradients. These principles are the foundation of modern vision models and other deep learning applications.

2.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) is an advancement in deep learning models specifically designed to processs structured data, making it more suitable for computer vision tasks, such as image processing, classification, segmentation, and object detection *(Bhandare et al., 2016*).

The architecture of CNNs is designed to extract hierarchical features from data, capturing low-level patterns such as edges and textures in the initial layers and progressively learning high-level representations such as objects and shapes in deeper layers (Lorenzo, Luca and Giorgio, 2020). This feature extraction capability enables CNNs to outperform traditional fully connected neural networks (MLPs) in tasks involving high dimensional structured data, particularly images (Jeyalakshmi and Rangaraj, 2021). The fundamental operation in CNNs is convolution, which applies a small kernel (filter) to an input, systematically moving across the input to produce a feature map.

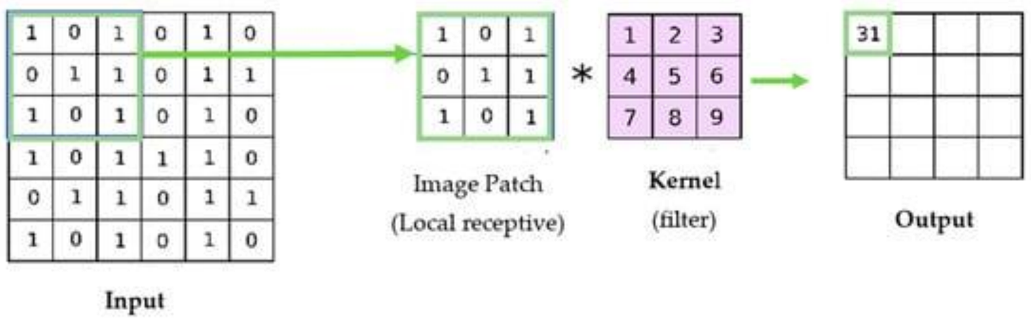


Figure 6 Convolution with kernel to produce an output feature map (Salehi et al., 2023)

These convolution operations are widely used in image processing for feature extraction. Some common applications include edge detection (e.g., Sobel, Prewitt filters), blurring and noise reduction (e.g., Gaussian blur) and Sharpening (e.g., Laplacian filters) used in image processing (Coady et al., 2019). In CNNs, instead of using predefined filters, the network learns optimal convolutional filters during training, enabling it to adaptively extract relevant features from data.

CNNs can be broken down into five different layers: convolutional, activation, pooling, fully connected and ouput layers.

Convolutional Layer: This layer applies filters (or kernels) to the input data to detect specific features, such as edges, textures, or shapes resulting in feature maps (Khan, et al. 2020).

Activation Layers: These apply an activation function (such as ReLU or sigmoid) to introduce nonlinearity into the model, which helps the network learn complex patterns. ReLU (Rectified Linear Unit) is a commonly used activation function in deep learning (Khan, et al. 2020).

Pooling Layer: This layer reduces the spatial size of the feature maps produced by the convolutional layer. This helps to reduce the computational load and focuses on the most important information.

Fully Connected Layer: After feature extraction, the fully connected layer flattens the data and connects every neuron in one layer to every neuron in the next (same as in MLPs). It's usually placed near the end of the network and helps in making final predictions or classifications (Bhandare et al., 2016).

Output Layer: the final layer that produces the predictions or outputs of the model.

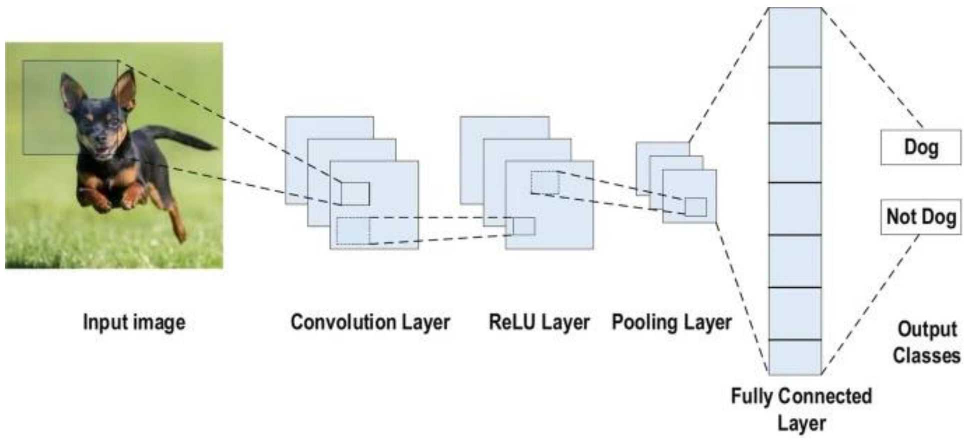


Figure 7 Illustration of CNN architecture, showing how image features are processed through layers to classify image

Parameters in CNNs

Given the structure of CNNs, they introduce additional parameters that require fine tuning. These parameters can be categorised into two groups: **learnable parameters**, optimised during training through backpropagation and **hyperparameter** which are predetermined (Kohzadi Chegeni et al., 2023). This section will focus on parameters specific to CNNs, though it is important to note that other more general parameters (e.g. learning rate) also influence model performance.

Learnable parameters are values the network adjusts during training to minimise error through backpropagation, key parameters to CNN:

Table 1 CNN learnable parameters

Parameter type	**Description**	**Example in CNNs**

Kerne weights	Values within convolutional filters, tuned to detect spatial features	A 3×3 filter in the first layer learning to recognize edges or gradients.
Biases	Offset terms added to outputs of neurons.	A scalar added to the output of each filter to shift activation thresholds.
Fully connected Weights	Weights in dense layers that map flattened feature maps to output class probabilities.	Weights linking the flattened feature maps (input dimension) to the output layer (number of classes).

Hyperparameters are predetermined setting that define the model's architecture or training protocol (how model is trained). They are set before training and remain fixed unless manually adjusted *(Salehi et al., 2023).* Key hyperparameters of CNN:

Table 2 CNN hyperparameters

Hyperparameter	**Description**	**Impact on Model**
Kernel size	Dimensions of convolutional filters.	Larger kernels capture broader patterns but increase computational cost.
Stride	Step size of kernel movement during convolution.	Larger strides reduce spatial resolution, aiding down sampling but risking information loss.
Padding	Extra pixels added to input borders to preserve dimensions.	Prevents edge information loss, critical for small objects.
Number of filters	Number of kernels per convolutional layer.	More filters increase feature diversity but expand model size.
Type of Pooling layer	Operation to down sample feature maps	Reduces spatial dimensions while preserving important features.

Backpropagation in CNNs differs from fully connected layers due to weight sharing and local connectivity. In convolutional layers, gradients are computed by convolving the error with the inputs, updating shared filters. For a more detailed explanation, refer to sources such as Aghdam and Heravi (2017), which discuss backpropagation in CNNs.

CNN are applied across many different industries including automotive for fast vehicle detection (Nguyen, 2019), medical for X-ray image recognition (Zhang et al., 2021) and agricultural for crop monitoring (Chen et al., 2021). They have become an important tool in modern AI application due to their capacity to automatically learn hierarchical features.

2.4 Object detection models

This project will be focused on using 2D object detection model for autonomous vehicles. Object detection aims to locate and classify objects in images (Chen et al., 2023). It is a computer vision task that heavily uses deep learning techniques to interpret and understand visual information. Deep learning is a subset of machine learning which is further encompassed by Artificial Intelligence (AI).

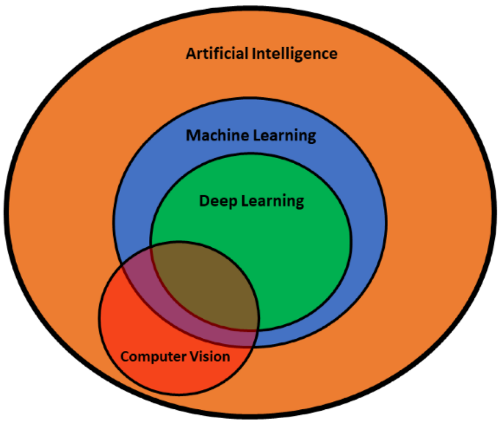


Figure 8: Relationship Between AI, Machine Learning, Deep Learning, and Computer Vision (Mohimont et al., 2022)

Furthermore, object detection is a supervised learning task (machine learning technique) where the model is trained using labelled data, to learn patterns in images by associating input images with their correct labels.

Comparing Models & Selection for this Project

Selecting models for autonomous vehicles involves choosing from various types of object detectors. The most prominent among these include single-stage detectors, two-stage detectors, and vision transformers (ViTs). The table below compares some leading models within these categories by outlining their types and key architectural characteristics:

Table 3: Comparing the architecture of different models

Model	**Type**	**Architecture**
YOLO (You Only Look Once)	Single-stage detector	Uses unified detection head to predict bounding boxes and class probabilities; typically uses a CSPDarknet or similar backbone for feature extraction *(Gragnaniello et al., 2023).*
CenterNet	Single-stage detector	Detects objects as a triplet of key points (one centre and two corners) Uses centre pooling and cascade corner pooling to more accurately find corners of objects *(Duan et al., 2019).*
EfficientDet	Single-stage detector	EfficientNet backbone with a BiFPN (bidirectional feature pyramid network) and multi resolution feature extractor *(Gragnaniello et al., 2023).*
Swin Transformer	Vision Transformer detector	Uses hierarchical feature maps and patch merging, excels at capturing long-range dependencies *(Gragnaniello et al., 2023).*
Faster R-CNN	Two-stage detector	Combines a Region Proposal Network (RPN) with a CNN backbone *(Bilous et al., 2024).*

Single-Stage Detectors:

Single-stage detectors such as YOLO, CenterNet and EfficientDet perform object detection in one unified pass through the network. They are designed to balance speed and accuracy, making them suitable for real-time applications like autonomous vehicles.

Two-Stage Detectors:
Two-stage detectors like Faster R-CNN first generate a set of candidate regions (region proposals) and then classify and refine these regions. This additional step typically results in higher detection accuracy, especially in complex scenarios, but comes at the cost of increased computational complexity and slower inference speeds.

Vision Transformer (ViT) Detectors:
Vision Transformer-based detectors, such as those incorporating the Swin Transformers, work by splitting an image into fixed-size patches, embedding each patch, and feeding them into a standard transformer model. The model applies multi-head self-attention to capture relationships between patches globally and uses a classification token (CLS) gathers information from all patches to make the final prediction *(Dosovitskiy et al., 2020).*

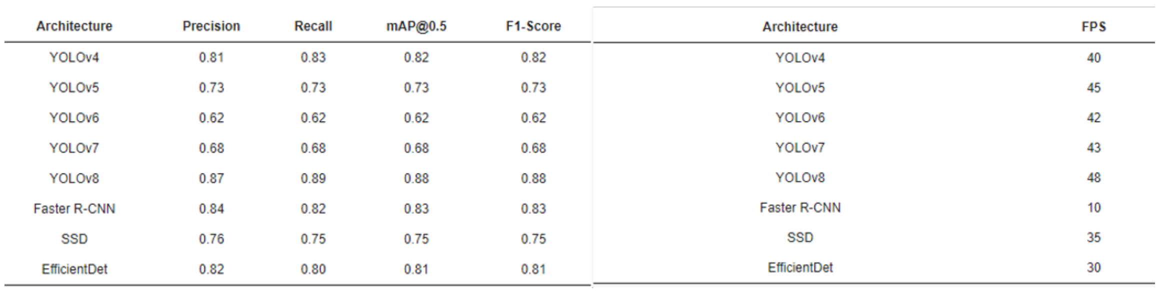
Why Choose YOLO?

When selecting vision models for autonomous vehicles, several critical performance metrics must be considered to ensure reliability, efficiency, and safety. The primary factors include inference speed, computational complexity (FLOPS), and accuracy (mAP), which directly impact real-time decision-making and overall system performance.

Table 4: Defining key performance metrics in object detection

Performance Metrics	**Definition**	**Measured in**
Inference Speed (Latency)	The time taken for the model to process an image or frame and make a prediction.	Milliseconds per frame (ms) or Frames Per Second (FPS)
Computational complexity	The amount of computational power required to run the model, affecting feasibility for real-time applications.	Floating Point Operations Per Second (FLOPS)
Mean Average Precision (mAP)	A measure of the accuracy of object detection models, calculated as the average precision across multiple classes and Intersection over Union (IoU) thresholds.	Percentage (%) or decimal value (0-1)

In terms of inference speed, YOLO-based architectures typically achieve the highest frames per second (FPS) among popular object detectors (Wang, Bochkovskiy & Liao, 2022). Compared to YOLO, transformer-based and two-stage detectors can achieve higher accuracy; however, their increased computational costs and slower inference speeds make them less practical for real-time applications. In situations where low latency is crucial, such as autonomous driving, these models may not be ideal despite their better mAP. The table below compares faster R-CNN, YOLO models and EfficientDet.



Architecture	Precision	Recall	mAP@0.5	F1-Score
YOLOv4	0.81	0.83	0.82	0.82
YOLOv5	0.73	0.73	0.73	0.73
YOLOv6	0.62	0.62	0.62	0.62
YOLOv7	0.68	0.68	0.68	0.68
YOLOv8	0.87	0.89	0.88	0.88
Faster R-CNN	0.84	0.82	0.83	0.83
SSD	0.76	0.75	0.75	0.75
EfficientDet	0.82	0.80	0.81	0.81

Architecture	FPS
YOLOv4	40
YOLOv5	45
YOLOv6	42
YOLOv7	43
YOLOv8	48
Faster R-CNN	10
SSD	35
EfficientDet	30

Figure 9: Comparing performance metrics of different models (Bilous et al. 2024)

Single-stage detectors achieve competitive accuracy, with YOLO and EfficientDet particularly excelling in balancing performance and computational efficiency for real-time applications. Although CenterNet offers strong detection capabilities, it has slower inference speed compared to other single stage detectors. The table below provides a comparison of models including CenterNet and YOLO, informed by recent research on deep learning methods for ship detection in marine surveillance (Hong et al., 2021).

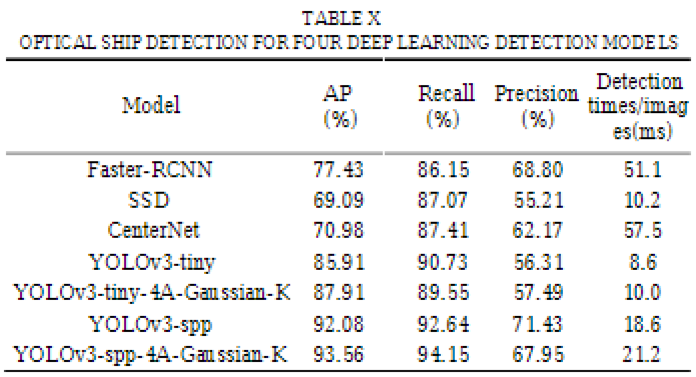


TABLE X
OPTICAL SHIP DETECTION FOR FOUR DEEP LEARNING DETECTION MODELS

Model	AP (%)	Recall (%)	Precision (%)	Detection times/images(ms)
Faster-RCNN	77.43	86.15	68.80	51.1
SSD	69.09	87.07	55.21	10.2
CenterNet	70.98	87.41	62.17	57.5
YOLOv3-tiny	85.91	90.73	56.31	8.6
YOLOv3-tiny-4A-Gaussian-K	87.91	89.55	57.49	10.0
YOLOv3-spp	92.08	92.64	71.43	18.6
YOLOv3-spp-4A-Gaussian-K	93.56	94.15	67.95	21.2

Figure 10: Comparing performance metrics of different models (Hong et al., 2021)

In summary, the choice of object detector depends on the specific requirements of the task. For applications in autonomous vehicles, where real-time performance and efficient use of hardware are critical factors, YOLO is a suitable choice. By striking a balance between these key attributes it meets demands for real time detection in dynamic and environments where safety is crucial.

2.5 Ethics and Safety Considerations

Developing object detection models for autonomous vehicles (AVs) requires careful attention to ethical and safety standards throughout the development cycle. This project leverages the nuImages dataset (Caesar et al., 2019). Since it is a large and diverse dataset this should help mitigate unintended biases related to specific scenarios or demographics.

Furthermore, the data collection of nuImages aligns with ISO 23053, ensuring it complies with data privacy regulations. Faces are blurred or images removed to prevent misuse of sensitive information. The project also maintains clear documentation of data sources, any pre-trained models used and evaluation methods applied to promote transparency and ensure reproducibility.

Despite efforts to improve detection accuracy, AVs will inevitably encounter situations where accidents cannot be avoided. This raises ethical concerns about how AVs should be programmed to respond in high-risk scenarios. Moreover, questions about liability and responsibility of accidents are unclear, whether this falls on the manufacturer, software developers or regulators.

3 Methodology

The methodology outlines the end-to-end process followed in developing the object detection model. It covers dataset preparation, the selection and architecture of the YOLOv10 model, and the training configurations used. Additionally, it details the model's outputs, loss functions, te overall training workflow as well as the software and hardware setup. The methodology is designed to ensure transparency and reproducibility, allowing others to replicate and build upon the project.

3.1 Dataset Preparation

The dataset used in this project is a subset of the nuImages dataset, taken from the larger nuScenes multimodal dataset for autonomous driving (Caesar et al., 2019). The nuImages dataset was selected due to its diversity and extensive 2D annotations, making it suitable for training an object detection model for autonomous vehicles. It provides high-quality 2D annotated images and a range of challenging driving scenarios:

- **Large scale**: Contains 93,000 2D annotated images and 1.2 million camera images in total.
- **Diverse scenarios**: Includes varied driving conditions such as day, night, rain, and snow.
- **High-quality annotations**: Features 2D bounding boxes, instance masks, and segmentation labels.
- **Real-world applicability**: The dataset is derived from real driving logs and reflects a wide range of traffic and environmental conditions.

Data Preprocessing:

To emphasise moving objects in a driving scenario that pose the most risk of damage or injury, the dataset labels were narrowed down to six classes **pedestrian, car, bus, bicycle, truck** and **motorcycle.** Police cars and ambulances from the original dataset were merged with cars and trucks, respectively, since they were rare, appearing in only 132 and 40 images out of 93,000. This ensured that if they did appear, they would still be classified under a relevant category. After this, the original dataset annotations were relabelled and converted into YOLO format.

The dataset of 4722 images was then split into **train (70%)**, **validation (20%)**, and **test (10%)** sets.

1. The **training set** is used to teach the model by adjusting weights based on patterns in the data.

2. The **validation set** helps fine-tune hyperparameters and monitor performance to prevent overfitting.

3. The **test set** provides an unbiased evaluation of the model on unseen data.

To mitigate potential performance issues from underrepresented classes, across the training and validation there are at least **1,500 instances** of every class. Data was selected so that once the 1,500 threshold was met, only images from underrepresented classes (with fewer than 1,500 instances) were selected. Despite these adjustments, some class imbalance remains, reflecting the distribution of real-world traffic scenarios.

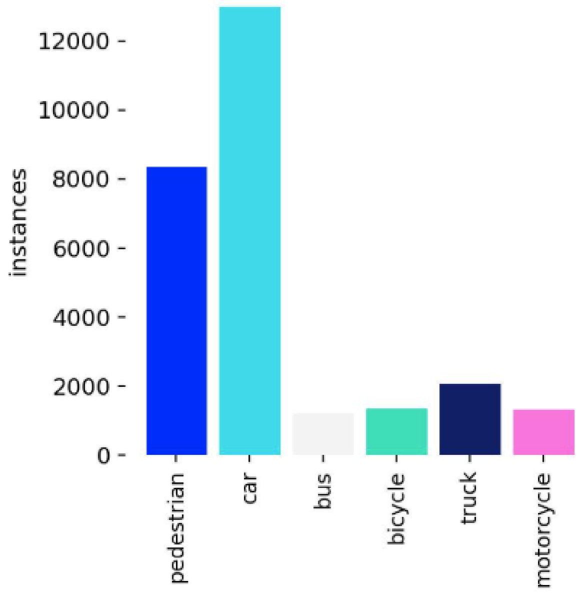


Figure 11: Breakdown of object instances

3.2 Model Architecture & Selection

YOLOv10 was chosen because it was specifically designed for object detection by researchers at Tsinghua University (Wang et al., 2024). It builds upon strengths of earlier YOLO models and introduces new innovations to achieve high performance under tight latency constraints, making it suitable for autonomous driving.

While CNNs are composed of convolutional, activation, pooling, fully connected, and output layers to learn features from images, object detection models arrange these layers into a more specialised structure designed to both locate and classify objects within an image. Typically, object detectors are divided into three main parts: a **backbone**, a **neck**, and a **head**.

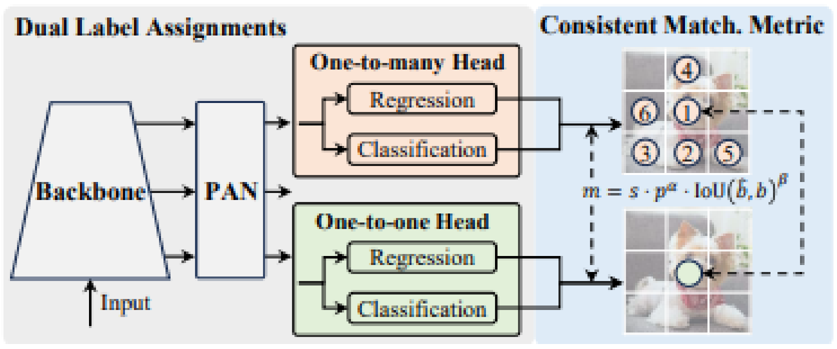


Figure 12: YOLOv10 architecture (Wang et al., 2024).

Backbone: YOLOv10 uses an improved CSPNet (Cross Stage Partial Network) as its CNN backbone. This network is a feature extractor used to generate feature maps, which is how the model learns and remembers important details and patterns of the objects.

Neck: The neck acts as a bridge between the backbone and the detection head, the neck in YOLOv10 is a Path Aggregation Network (PAN). PAN fuses features from different scales, which helps in detecting objects of various sizes.

Head: Traditionally, YOLO detectors use one to many label assignment and then use non maximum suppression (NMS) to remove redundant predictions (Kateb et al., 2021). In YOLOv10, the head is split into two parts: one part for training (using a one-to-many label assignment that provides detailed feedback) and another for inference (producing a single, high-confidence prediction per object to streamline the process and reduce latency).

Model Selection

To evaluate trade-off between accuracy and efficiency, three variants of YOLOv10 are trained – YOLOv10n (Nano), YOLOv10m (Medium), and YOLOv10l (Large). These models differ in size and complexity, allowing for a comparative analysis to determine the optimal model for real-time detection.

Transfer learning is used on each of these models, they have been pre-trained on the COCO dataset, a large dataset containing a diverse range of 80 object categories (Lin et al., 2014). The pre-training allows the model to retain general feature extraction capabilities, reducing training time and computational cost.

3.3 Training Configurations

For a fair and robust comparison across the object detection models used in this project, the training setup is standardised. Two key areas in the configuration are: **data augmentation** and **training parameters.**

Data Augmentation

Training with data augmentation helps simulate real world conditions to improve model robustness. Especially important in driving scenarios where there could be a variation in lighting, weather conditions and perspectives that can cause errors, leading to accidents (Shu, Shen, Lin & Goldstein, 2021). To ensure a reliable comparison between models the same set of augmentations are applied. The table below summarises these augmentations, their settings and their intended impacts.

Table 5: Description of data augmentations

Augmentation	**Description**	**Purpose**
Rotation	Maximum rotation ±10°	Randomly rotates image to simulate camera tilt, to help model recognise objects from varied angles.
Translation	Factor 0.1	Translates image horizontally or vertically to mimic small camera movement or misalignment
Scaling	Factor 0.2	Resize image to simulate objects at different distances
Horizontal Flip	Probability 0.5	Mirrors images
HSV Adjustments	– hsv_h: 0.015, – hsv_s: 0.7, – hsv_v: 0.4 (Factor)	Adjusts the hue saturation and brightness of the image. Simulates varying lighting conditions.
Mosaic	Factor 1.0	Combines four training images into one, simulate crowded and more complex scenarios.
Random Erasing	Probability: 0.4	Randomly erases a portion of the image, simulate obstacles and helps model to learn parts of objects.

Training Parameters

The training parameters are chosen to balance convergence speed, model accuracy and computational efficiency. Parameters such as learning rate, batch size, and weight decay influence how the model learns, while techniques like momentum and warmup help stabilise training (Kvietkauskas and Stefanovič, 2023). The table below outlines training parameters used.

Table 6: Description of training parameters and their values

Parameter	**Value**	**Description**
Epochs	100	Total number of complete passes through the training dataset.

Batch Size	16	Number of images processed per training iteration.
Image Size	640	Resolution of input images (640×640 pixels)
Learning Rate	Lr0: 1e-3 Lrf: 1e-4	Initial learning rate, linear scheduler used to decrease from initial (Lr0) to final (Lrf) value.
Momentum	0.9	Helps accelerate gradients in the relevant direction, smoothing the optimisation process.
Weight Decay	5e-4	Regularization term to prevent overfitting by penalizing large weights.
Warmup Epochs	3	Number of epochs for gradually increasing the learning rate to stabilize early training.
Optimiser	AdamW	Adjusts models weights after each epoch
Patience	10	Number of epoch training process would stop if the validation metrics did not improve.

3.4 Model Output & Loss Function

The YOLOv10 model generates predictions of bounding box coordinates (YOLO format), confidence score and class probability for each class. The confidence score indicates the model's certainty that a detected object exists within the predicted bounding box, while the class probability represents the likelihood that the object belongs to a specific class.

YOLO bounding box format: [x_center, y_center, width, height] with values normalised between 0 and 1 as shown in figure 14.

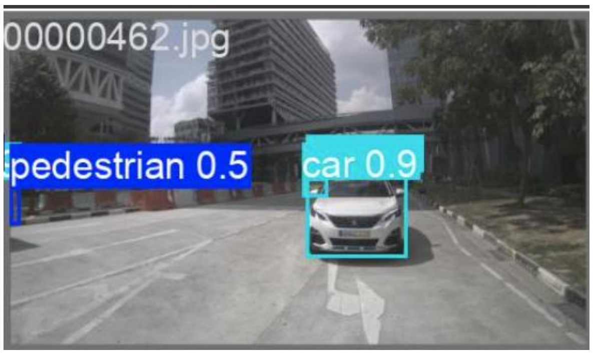


Figure 13: Example prediction from YOLOv10 model

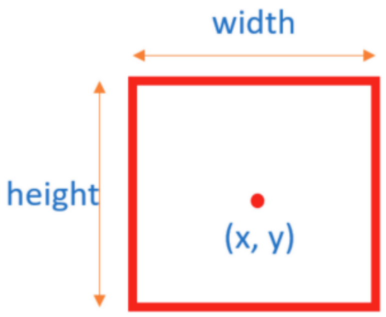


Figure 14: YOLO bounding box format (Son, Jinhwan & Jung, 2024)

The training process is guided by feedback from three primary loss components: bounding box, classification loss and distributed focal loss. Each loss function is designed to optimise a specific aspect of object detection task (Jiang, Qin, Zhang & Zheng, 2020). Below is explanation of each loss function.

Table 7: Description of loss function and their weights

Loss Type	**Loss Function Used**	**Description**	**Weight**
Bounding Box Loss	Complete Intersection over Union (CIoU) Loss	Measures the accuracy of predicted bounding box coordinates relative to ground truth.	7.5
Classification Loss	Binary Cross-Entropy (BCE) Loss	Measures the accuracy of predicted class probabilities relative to ground truth labels. Penalises incorrect class predictions.	0.5
Distribution Focal Loss	Softmax-weighted Cross-Entropy	Measures the precision of bounding box coordinate predictions by focusing on the distribution of possible values. Focuses on small or difficult to detect objects by distributing the loss depending on how hard it is for the model to predict the bounding box coordinates accurately.	1.5

Figure 15 shows example validation loss graphs, where losses start high and decrease steeply before gradually stabilizing. This trend reflects rapid early learning followed by convergence.

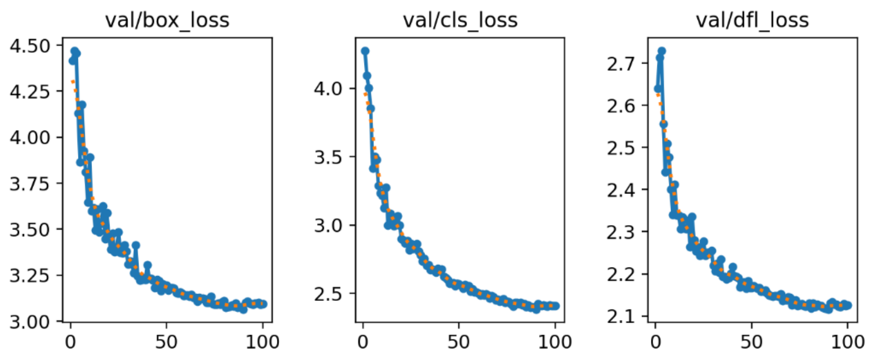


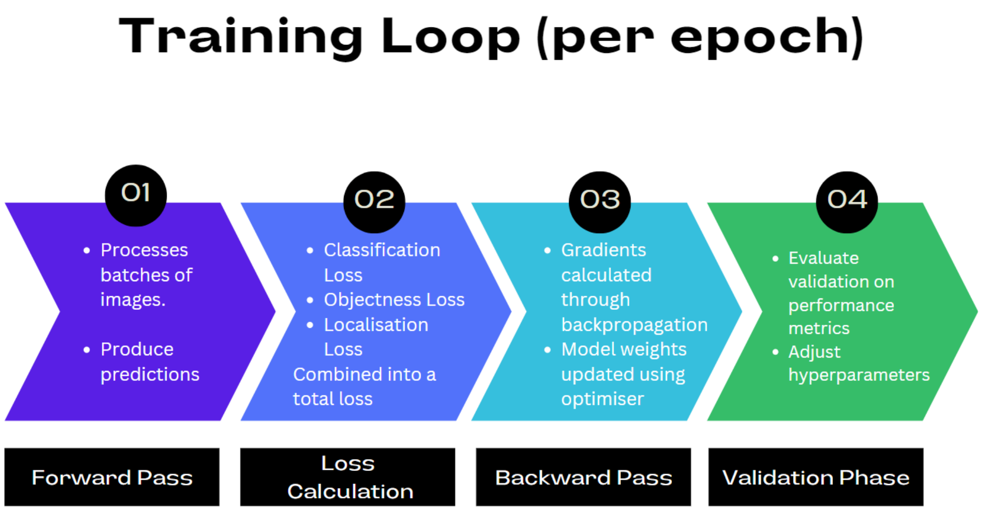
Figure 15: Example of loss graph from first 100 epoch of YOLOv10m model

An initial warmup period of 3 epoch during which the learning rate gradually increased to its maximum value 0.001. A linear scheduler is used to reduce the learning rate the remainder of training. This gradual reduction in learning rate allowed the model to take larger steps early in training when it was far from optimal and then make increasingly finer adjustments as training progressed to improve convergence.

3.5 Training Workflow

During each epoch, the training process iterates over all images in the training set in mini batches of 16. For each mini batch, the model performs a forward pass, computing and tracking losses based on the predicted bounding boxes and class probabilities. The losses are then calculated through backpropagation and the model's weights are updated using an optimiser.

After each epoch the model's performance is evaluated on the validation set. The validation metrics are tracked and compared against previous epochs to update the models best and last weights save, to ensure highest performing version is saved.



Training Loop (per epoch)
01
• Processes batches of images.
• Produce predictions
02
• Classification Loss
• Objectness Loss
• Localisation Loss
Combined into a total loss
03
• Gradients calculated through backpropagation
• Model weights updated using optimiser
04
• Evaluate validation on performance metrics
• Adjust hyperparameters
Forward Pass
Loss Calculation
Backward Pass
Validation Phase

Figure 16: Training loop for an epoch

After the training loop, early stopping was implemented to prevent overfitting by monitoring the validation performance metrics. A patience of 10 was used, meaning the training process would stop if the validation metrics did not improve for 10 consecutive epochs.

3.5 Software and Hardware

The development environment was primarily using Python and the Ultralytics framework for implementing and training the YOLO-based models, which also provided Albumentations library for data augmentation.

All model training was on Google colabs cloud platform, using a Telsa T4 GPU rented through Colab. To optimise cloud resources, checkpointing was implemented to save best and last model weights after each epoch, and training logs were continuously maintained. This approach safeguarded against session timeouts and allowed training to resume from latest checkpoint if interrupted.

4 Results & Discussion

The evaluation aims to compare the performance of three YOLOv10 variants—YOLOv10n, YOLOv10s, and YOLOv10l—on the nuImages dataset. The goal is to identify the best-performing model for autonomous driving applications,
balancing accuracy and inference speed. The selected model will then be fine-tuned by adjusting the classification loss, object loss and Distribution Focal Loss (DFL) weights to further optimise performance.

Each model was initially trained for 100 epochs, then continued for up to 100 more with patience 10. The total epoch counts varied due to early stopping.

Table 8: Total number of epoch each model was trained for

	YOLOv10n	**YOLOv10m**	**YOLOv10L**
Total Epoch	197	152	200

4.1 Evaluation Metrics

Mean Average Precision (mAP) is a key evaluation metric for object detection models, measuring how well a model detects and classifies objects (Henderson & Ferrari, 2016). It assesses both **accuracy** (how well bounding boxes align with objects) and **classification accuracy** (how correctly objects are identified).

The Intersection over Union (IoU) is the overlap between the predicted and ground truth bounding boxes:

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union}$$

mAP-50 measures the model's precision when the IoU threshold is 0.50 (50%)

mAP-50-95 metric averages mAP across multiple IoU thresholds, from 0.50 to 0.95 in 0.05 increments. It evaluates how well the model performs well across varying degrees of precision.

Inference Speed measured in milliseconds or frames per second (FPS), indicating real-time processing capability.

TP - True Positive, FP - False Positive, TN – True Negative, FN – False Negative.

Precision measures how many detected objects are actually correct.

$$Precision = \frac{TP}{TP + FP}$$

Recall measures how many actual objects were correctly detected.

$$Recall = \frac{TP}{TP + FN}$$

F1-Score balances precision and recall, measuring how well the model avoids FP and FN.

$$F1 - Score = 2 \times \frac{Precision\ \times Recall}{Precision + Recall}$$

To evaluate the performance of YOLOv10 models in a real-world scenario, a **confidence threshold** of **0.5** was used, with a **batch size** of **6** to simulate the multi-camera setup used in the nuScenes. This setup reflects a realistic data collection stream, where multiple cameras continuously capture frames in a driving environment.

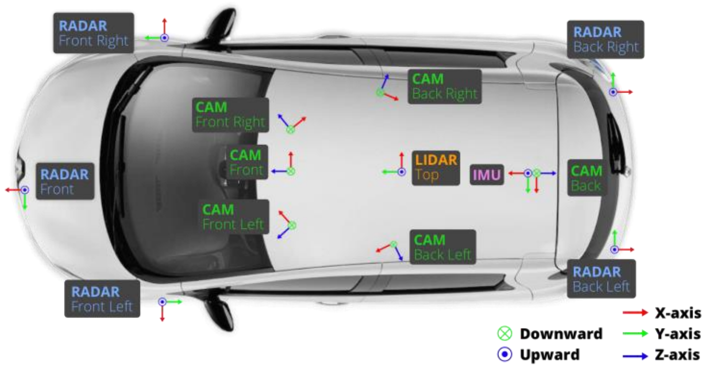


Figure 17: nuScenes multi-camera setup

The Model Comparison Table provides insights into the detection performance and efficiency of three different YOLOv10 variants (nano, medium, and large).

Table 9: Comparing performance metrics of YOLOv10 variants

Model	**mAP-50**	**mAP-50-95**	**F1-Score**	**Inference time (ms/FPS)**
YOLOv10n	0.600	0.424	0.493	7.6 / 132
YOLOv10m	0.669	0.478	0.599	9.4 / 106
YOLOv10L	0.711	0.526	0.660	13.0 / 77

Based on the results, YOLOv10L was selected as the final model for further fine tuning, as it had better detection performance.

- Achieved the **highest mAP-50 (0.711) and mAP-50-95 (0.526)**, ensuring better object localization across varying IoU thresholds.
- **F1-Score (0.660)** indicates a strong balance between precision and recall, reducing both false positives and false negatives.
- While its **inference time** (**13.0 ms / 77 FPS**) is slower than the other models, the accuracy gains justify this trade-off, especially for an autonomous vehicle scenario where detection reliability is critical.

Table 9 & 10 show examples of YOLOv10L model predcitions under harsh weather conditions and poor lighting conditions.

Table 10: Validation examples under harsh weather conditions

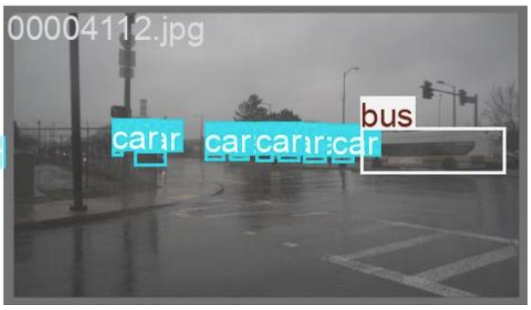
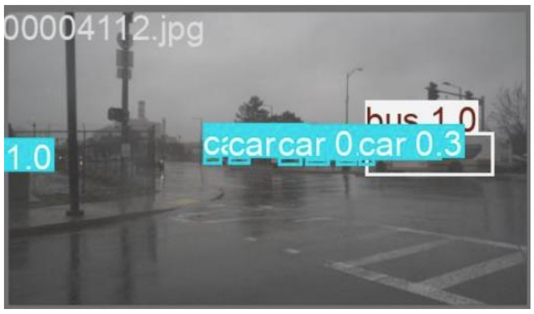
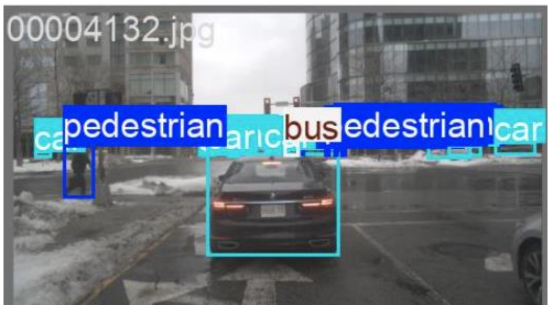
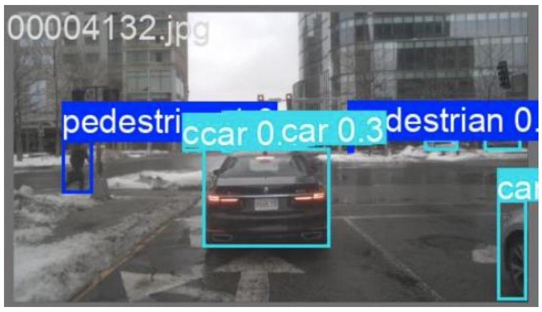
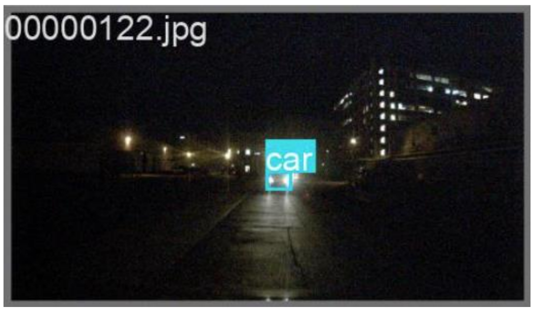
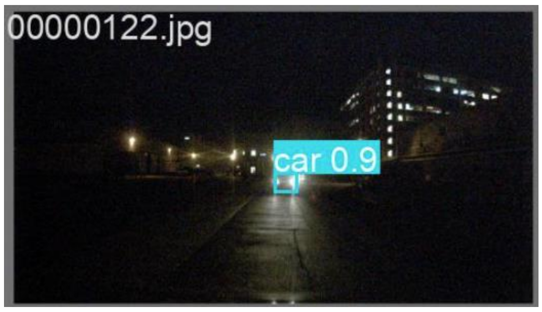
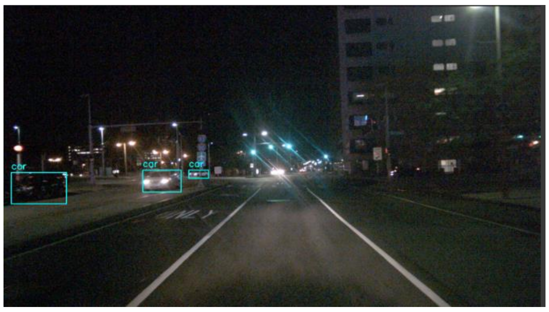
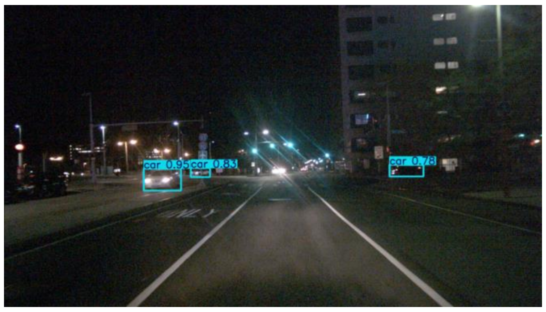
Ground Truth	Predictions
	
	

Table 11: Validation examples under poor lighting conditions

Ground truth	Predictions
	
	

4.2 Further Analysis

To further analyse the model's performance, the project examined evaluation metrics for each class, including **precision, recall, and mAP scores**. Additionally, the **average size of objects in each class** (measured as a percentage of the total image area) was analysed to identify potential relationships between object size and detection performance (Zhang and Zhang, 2024). By observing errors made by the model, possible reasons for variations in performance across different object sizes and categories were explored. The table below summarises the evaluation on the test dataset.

Table 12: Performance metrics of YOLOv10L model across different classes

Class	Area (%)	Precision	Recall	mAP50	mAP50-95
Pedestrian	0.18	0.918	0.442	0.681	0.43
Car	0.55	0.879	0.65	0.786	0.627
Bus	1.74	0.804	0.493	0.667	0.544
Bicycle	0.70	0.813	0.599	0.748	0.538
Truck	1.20	0.755	0.431	0.584	0.445
Motorcycle	0.54	0.91	0.653	0.801	0.57

Smaller objects (pedestrians) had high precision but low recall, indicating that the model struggled to detect them. However, when detected, they were classified correctly. Larger object (buses and trucks) also had low recall despite their size, suggesting difficulties in detecting them accurately. Medium sized objects (cars, motorcycles, and bicycles) demonstrated the best overall performance, achieving a better balance between precision and recall. This was also evident in precision-recall graph.

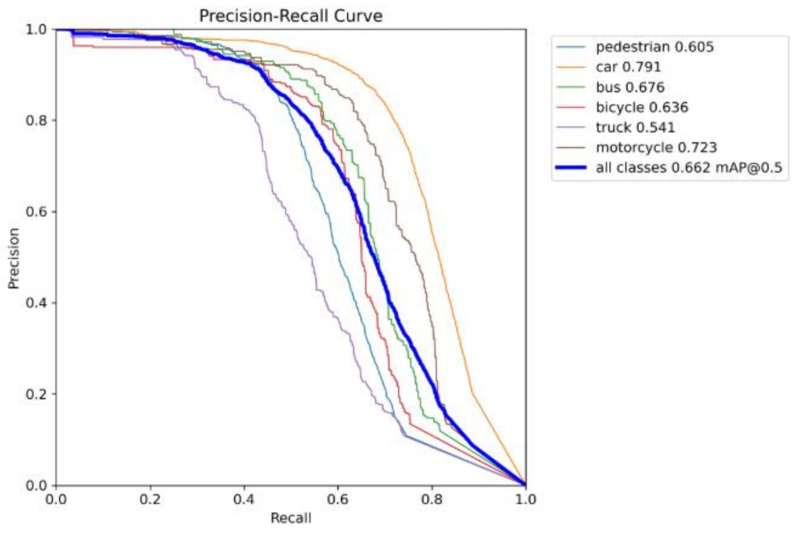


Figure 18: Precision-Recall graph

Recall was consistently lower across all classes compared to precision, indicating that the primary source of error came from false negatives. This issue was particularly evident for smaller objects, which were frequently missed during detection. Zhang et al. (2020) proposed using a lower IoU threshold for mAP in golf ball detection, prioritising detection over a strict bounding box. A similar approach can be applied here to balance precision and recall.

To better understand the distribution of false negatives, a confusion matrix was used (Figure 19). The confusion matrix summarises the model's predictions by comparing them to the actual labels. It shows how many instances were correctly classified and where the model made errors, including false positives and false negatives.

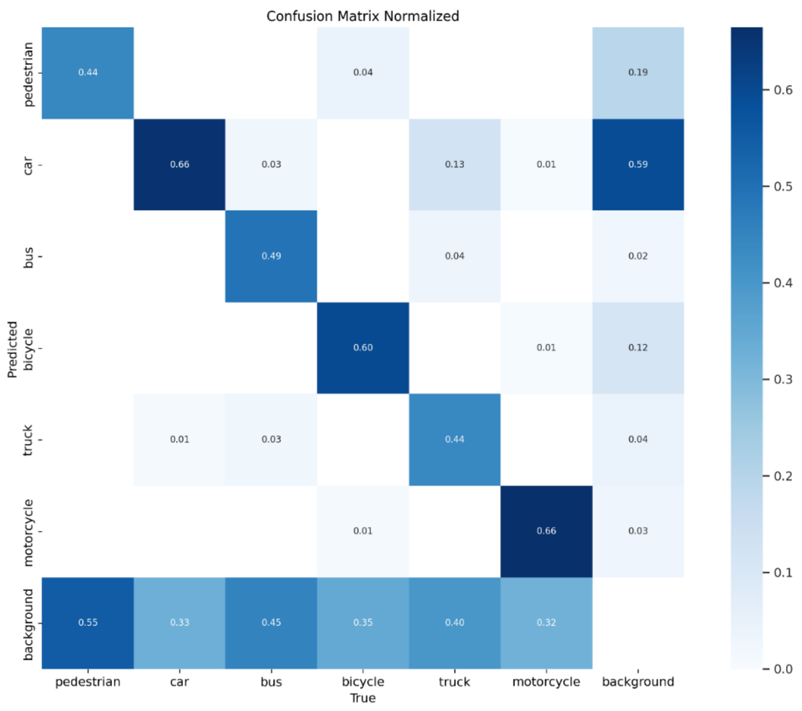


Figure 19: Normalised confusion matrix

The final row of the matrix, representing false negatives for the background class, aligns with the precision and recall data. It confirms that pedestrians had the highest number of false negatives, followed by larger objects such as trucks and buses. The matrix also indicates that the model effectively distinguished between different object classes. However, it reveals that cars were the most common source of false positives, meaning the model would mostly likely predicted a false positive as a car,.

Additionally, large objects were often misclassified as background, but the background was not commonly mistaken for large objects. This could indicate that trucks and buses blended into the environment, making them harder to recognise.

Fine tuning Loss function Weights

To reduce model errors, the loss function weights are adjusted to control the emphasis placed on each loss component. Loss graphs provide insight into the overall training behaviour, helping identify potential overfitting or underfitting. However, may not always directly indicate how to adjust loss function weights for improved detection performance. Instead, this project relies on evaluation metrics such as precision, recall, and mAP to more effectively identify weaknesses in the model's predictions.

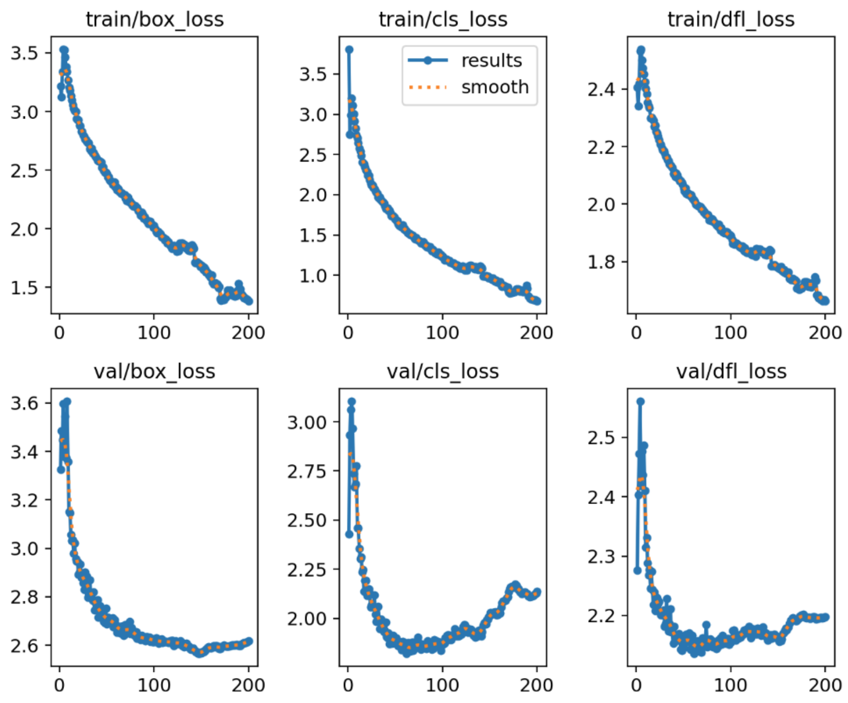


Figure 20: YOLOv10L training and validation loss graphs

The standard loss weight setup for box loss, classification loss and distributed focal loss are 7.5, 0.5 and 1.5 respectively. Based on the evaluation data, adjustments are necessary to improve the model's ability to detect smaller objects, differentiate large objects from the background, and increase recall for both large and small objects (Li et al., 2020). The following changes have been made, along with their justifications:

Table 13: Loss adjustments and their reasons

Loss type	**Change**	**Reason**
Box loss	Decrease (7.5 → **6.0**)	Lower weight to shift focus on classification and confidence. Additionally, reducing box loss will mean model is less strict about perfect box placement and more focused on detecting objects. Improve recall and small object detection.
Classification loss	Increase (0.5 → **1.0**)	Increasing classification loss encourages the model to be more confident in detections. Adjustment should benefit classes with low recall.
Distributed focal loss	Increase (1.5 → **2.5**)	Improve models' ability to distinguish and detect objects from the background.

The model was trained again from the same pretrained YOLOv10l model for 200 epoch with same parameters except the modified loss function.

Results of loss fine tuning

Table 14 compares performance metrics between both models **(initial model -> model after loss adjustments).**

Table 14: Comparing performance metric before & after loss adjustments

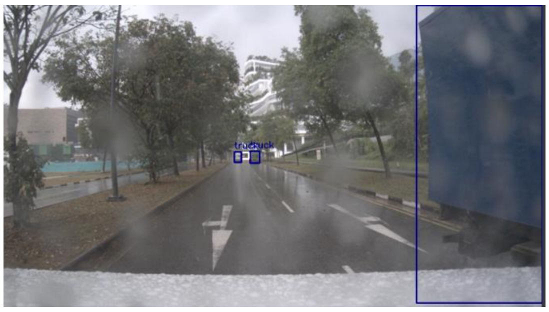
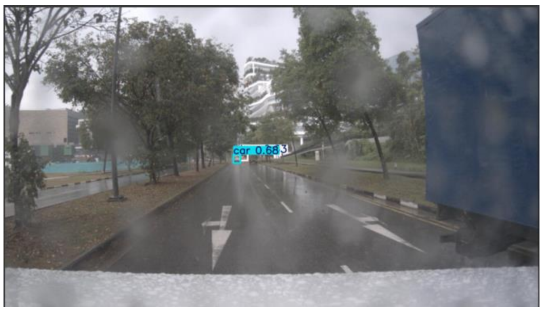
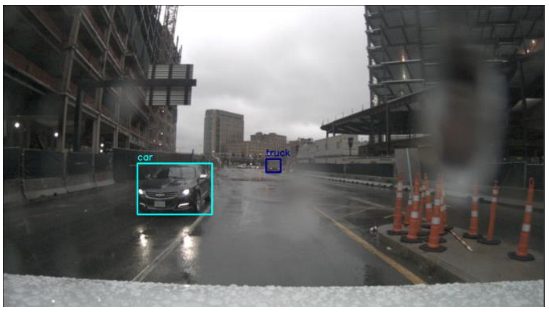
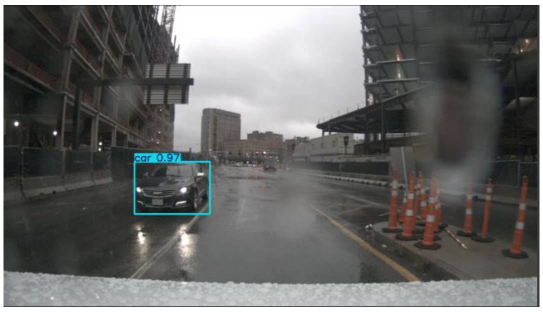
Class	**Area (%)**	**Precision**	**Recall**	**mAP50**	**mAP50-95**
Pedestrian	0.18	0.918 -> **0.907**	0.442 -> **0.451**	0.681 -> **0.687**	0.430-> 0.434
Car	0.55	0.879 -> **0.884**	0.650 -> **0.678**	0.786 -> **0.799**	0.627 -> **0.625**
Bus	1.74	0.804 -> **0.809**	0.493 -> **0.507**	0.667 -> **0.674**	0.544 -> **0.547**
Bicycle	0.70	0.813 -> **0.840**	0.599 -> **0.629**	0.748 -> **0.766**	0.538 -> **0.539**
Truck	1.20	0.755 -> **0.695**	0.431 -> **0.384**	0.584 -> **0.542**	0.445 -> **0.430**

Motorcycle	0.54	0.910 -> **0.907**	0.653 -> **0.688**	0.801 -> **0.815**	0.570 -> **0.563**
Overall	0.53	0.850 -> **0.840**	0.545 -> **0.556**	0.711 -> **0.714**	0.526 -> **0.523**

The results show a slight improvement in **mAP50** and **recall**, but at the cost of a decrease in **mAP50-95 and precision**, suggesting a trade-off between detecting more objects and maintaining higher confidence in predictions.

Table X presents examples where the model failed to detect the truck class.

Table 15: Example of missed truck detection

Ground Truth	**Prediction**
	
	

While **recall** and **mAP50** improved across all classes, the **truck** class experienced a decline in every metric. This drop in performance may be related to the merging of trucks and ambulances during dataset preparation, given that ambulances were extremely rare, appearing in only 40 out of 93,000 images. However, since no cases were observed in the test data where an ambulance was misclassified, this alone does not fully explain the issue

An alternative explanation is that the rigid structure of trucks closely resembles buildings, which the model has likely learned to ignore. This similarity may cause confusion, especially when trucks are farther away or only partially visible, making them harder to distinguish from the background. This unintended effect appears to be a consequence of the adjustments made to the loss function weights, altering how the model prioritises object detection. Table x shows metrics without truck class.

Table 16: Performance metric excluding truck class

Class	Precision	Recall	mAP50	mAP50-95
Overall (without truck)	0.865 -> **0.869**	0.567 -> **0.591**	0.737 -> 0.**748**	0.542 -> **0.542**

Excluding the Truck class, the model showed improvements in **Precision**, **Recall**, and **mAP50**, indicating that the loss adjustments likely worked well for other classes. The decline in truck detection highlights the need for further investigation and experimentation, such as expanding the dataset or incorporating a more diverse range of vehicle types to assess their impact on overall performance.

4.3 Discussion

The project explored the performance of a YOLOv10-based object detection model for autonomous vehicle (AV) vision systems, focusing on the trade-offs between detection accuracy and real-time feasibility. The primary goal was to address leading causes of AV crashes—misclassification and failure to detect objects, particularly in scenarios involving small objects, pedestrians, vehicles, and challenging environmental conditions. The project evaluated key metrics such as precision, recall, and mAP, and adjusted loss function weights to improve detection performance.

Key findings

1. Model Selection:
    - YOLOv10L was chosen over smaller variants (e.g., Nano, Medium) because its performance improvement was more significant than the increase in inference time, making it better suited for AV applications.
2. Detection Performance:

    - Adjusting loss weights—**decreasing box loss**, **increasing classification loss**, and **increasing distributed focal loss (DFL)**—improved recall and mAP50, indicating better detection of smaller and distant objects. However, this came at the cost of slightly lower precision and mAP50-95, highlighting a trade-off between detecting more objects and maintaining high confidence in predictions (Terven et al., 2024).
3. Truck Class Performance:
    - **The** truck class performed was worse than expected, despite having a larger area compared to other, which would typically make detection easier. However, even after adjusting loss weights, performance declined further rather than improving. One possible explanation is that trucks share rigid edges and structural similarities with buildings, which are commonly present in the background of driving scenes. At a distance, the model may struggle to differentiate between trucks and buildings, as their overall features appear similar.

Challenges and limitations

One of the challenges encountered was overfitting, as indicated by the validation loss graphs. While these graphs showed signs of overfitting, performance metrics continued to improve, creating a conflicting outcome. Although the model may have shown signs of overfitting in some aspects, it still performed well on unseen data, indicating that the overfitting did not severely impact its ability to generalise.

Hardware constraints could be a limitation as T4 GPU was used for validation, which may not accurately reflect real-world inference times achievable on autonomous vehicle hardware, such as the Jetson Nano. In addition to this, although improvements in detection metrics suggests a better performing, this does not directly translate to fewer AV crashes. Object detection is only one part of the AV perception.

Approaches attempted that were unsuccessful:

- Training the first YOLOv10L model beyond 200 epochs was attempted, but this led to a decline in performance, suggesting additional training time alone was not beneficial

- Another approach was training YOLOv3, a larger model expected to perform better due to its increased capacity. However, YOLOv3 performed worse in both detection accuracy and inference speed, reinforcing the significant improvements made in newer YOLO models

- Attempted to simulate Jetson nano performance by limiting processing power of GPU. Results were inconsistent so were not included in final project.

Overall, the model performed well on a dataset designed for challenging driving scenarios, achieving an mAP50 of 0.714 with an inference time of 13ms (77 FPS). The results also highlighted the impact of loss weight adjustments on performance, reinforcing the importance of fine-tuning hyperparameters for optimal results. To further improve, exploring techniques such as adaptive loss weighting could be a promising direction (Ocampo et al., 2024). These findings contribute to the ongoing development of object detection for autonomous vehicles to create more robust and safe technology.

5 Conclusion

The aim of this project was to investigate the causes of errors in existing vision models for autonomous vehicles (AVs), select a suitable model and modify the model parameters to improve the model. Misclassification due to environmental factors was a leading cause of AV crashes. To help mitigate this problem nuImages dataset was used, which had varying whether condition and lighting conditions. Data augmentations were also applied to these images to further emphasise harsh driving scenarios.

By analysing the architecture, key parameters and sources of error, allowed appropriate adjustments to be made for more accurately detect pedestrians, cars, buses, bicycles, trucks and motorcycles in driving scenarios. The evaluation identified low recall of 0.545 across all classes, specifically worse for pedestrians (0.442) which had a smaller bounding box size on average. To address this, adjustments were made to the loss function weights, increasing classification (0.5 to 1.0) and dfl loss (1.5 to 2.5) while decreasing box loss (7.5 to 6.0). This increased recall to 0.556 and led to improved detection of vehicles and pedestrians at mAP50. Although the trade-off was reduced precision, which in the context of autonomous driving may be acceptable, because detecting objects reliably could be prioritised over having perfectly precise bounding boxes.

Future Work

For future work, a larger dataset would improve the models' generalisation and detection on new scenarios. Alternative techniques to modifying loss function could also be explored through adaptive loss weighting to mitigate remaining errors. Additionally, testing on AV specific hardware such as the Jetson Nano would provide a more realistic assessment of inference speeds.

In conclusion, this project highlights the importance of optimising object detection models for AV safety. By refining detection strategies and balancing accuracy trade-offs, AV systems can make more informed decisions, reducing the risk of accidents caused by misclassification or missed detections. These findings contribute to the broader goal of improving autonomous vehicle perception.

References

- Zandt, F., 2024. AV testing makes strides in California. Autonomous driving, 6 September. https://www.statista.com/chart/13868/registered-autonomous-vehicles-to-be-tested-in-california/
- Azam S, Munir F, Sheri AM, Kim J, Jeon M. System, Design and Experimental Validation of Autonomous Vehicle in an Unconstrained Environment. Sensors, (2020), https://doi.org/10.3390/s20215999
- Collin, A. Bilka, S. Pendleton and R. D. Tebbens, Safety of the Intended Driving Behavior Using Rulebooks, IEEE Intelligent Vehicles Symposium (IV), (2020) doi: 10.1109/IV47402.2020.9304588
- Kurse, T. K., Gebresenbet, G., Daba, G. F., & Tefera, N. T. (2025). Experimental determination of factors causing crashes involving automated vehicles. Multimodal Transportation. https://doi.org/10.1016/j.multra.2024.100186
- Houseal, L.A., Gaweesh, S.M., Dadvar, S. and Ahmed, M.M., 2022. Causes and effects of autonomous vehicle field test crashes and disengagements using exploratory factor analysis, binary logistic regression, and decision trees. Transportation Research Record, 2676(8), pp.571-586. https://doi.org/10.1177/03611981221084677
- Rosenblatt, F., 1958. The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, https://www.ling.upenn.edu/courses/cogs501/Rosenblatt1958.pdf

- Nielsen, M.A., 2015. Neural Networks and Deep Learning. Available at: http://neuralnetworksanddeeplearning.com/
- Ramchoun, Hassan & Amine, Mohammed & Idrissi, Janati & Ghanou, Youssef & Ettaouil, Mohamed. (2016). Multilayer Perceptron: Architecture Optimization and Training. International Journal of Interactive Multimedia and Artificial Inteligence.
- Rumelhart, D., Hinton, G. & Williams, R. Learning representations by back-propagating errors. Nature 323, 533–536 (1986). https://doi.org/10.1038/323533a0
- Du, Ke-Lin & Swamy, M.N.s. (2014). Perceptrons. DOI:10.1007/978-1-4471-5571-3_3
- Fan, S. and Zhao, Y. (2019). ‘Analysis of DES Plaintext Recovery Based on BP Neural Network’, International Journal of Computer Science and Network Security, 19(11), pp. 13-20. Available at: https://doi.org/10.1155/2019/9580862
- Ruder, S. (2017). An overview of gradient descent optimization algorithms. arXiv. Available at: https://doi.org/10.48550/arXiv.1609.04747
- Popescu, M.-C., Balas, V., Perescu-Popescu, L. and Mastorakis, N., 2009. Multilayer perceptron and neural networks. WSEAS Transactions on Circuits and Systems, 8.
- Gultekin, S., Saha, A., Ratnaparkhi, A. and Paisley, J. (2020). ‘MBA: Mini-Batch AUC Optimization’, IEEE Transactions on Neural Networks and Learning Systems, 31(12),
- Agarap, Abien Fred. (2018). Deep Learning using Rectified Linear Units (ReLU). 10.48550/arXiv.1803.08375.
- Causevic, D., 2024. Mastering Backpropagation: Math and Implementation. https://dinocausevic.com/2024/07/14/mastering-backpropagation-math-and-implementation/
- Bhandare, A., Bhide, M., Gokhale, P. and Chandavarkar, R., 2016. Applications of Convolutional Neural Networks. International Journal of Computer Science and Information Technologies
- Lornezo, P., Luca, P. and Giorgio, G., 2020. Convolutional neural networks for relevance feedback in content based image retrieval. Multimedia Tools and Applications
- Jeyalakshmi, K. and Rangaraj, R. (2021). Accurate liver disease prediction system using convolutional neural network. Indian Journal of Science and Technology, 14(17), 1406-1421. https://doi.org/10.17485/ijst/v14i17.451
- Salehi, A. W., Khan, S., Gupta, G., Alabduallah, B. I., Almjally, A., Alsolai, H., Siddiqui, T., & Mellit, A. (2023). A Study of CNN and Transfer Learning in Medical Imaging: Advantages, Challenges, Future Scope. Sustainability, 15(7), 5930. https://doi.org/10.3390/su15075930

- J. Coady, A. O'Riordan, G. Dooly, T. Newe and D. Toal, "An Overview of Popular Digital Image Processing Filtering Operations," 2019 13th International Conference on Sensing Technology (ICST), Sydney, NSW, Australia, 2019, pp. 1-5, doi: 10.1109/ICST46873.2019.9047683.
- Khan, Asifullah, Sohail, Anabia, Zahoora, Umme, Qureshi, Aqsa Saeed, A survey of the recent architectures of deep convolutional neural networks. Artif Intell Rev 53, 5455–5516 (2020). https://doi.org/10.1007/s10462-020-09825-6
- Michelucci, Umberto. Advanced Applied Deep Learning: Convolutional Neural Networks and Object Detection. Apress, 2019, https://go.exlibris.link/w9wpjTXd
- Kohzadi Chegeni, M., Rashno, A. and Fadaei, S., 2023. Convolution-layer parameters optimization in Convolutional Neural Networks. Knowledge-Based Systems, [online] 261, p. 110210. DOI: 10.1016/j.knosys.2022.110210.
- Aghdam, H.H. and Heravi, E.J., 2017. Guide to Convolutional Neural Networks: A Practical Application to Traffic-Sign Detection and Classification. Springer.
- Nguyen, H. (2019). Improving faster r-cnn framework for fast vehicle detection. Mathematical Problems in Engineering, 2019(1). https://doi.org/10.1155/2019/3808064
- Zhang, Jicun, Song, Xueping, Feng, Jiawei, Fei, Jiyou, X-Ray Image Recognition Based on Improved Mask R-CNN Algorithm, Mathematical Problems in Engineering, 2021, 6544325, 14 pages, 2021. https://doi.org/10.1155/2021/6544325
- Chen, Wei, Zhang, Jingfeng, Guo, Biyu, Wei, Qingyu, Zhu, Zhiyu, An Apple Detection Method Based on Des-YOLO v4 Algorithm for Harvesting Robots in Complex Environment, Mathematical Problems in Engineering, 2021, 7351470, 12 pages, 2021. https://doi.org/10.1155/2021/7351470
- Mohimont, L., Alin, F., Rondeau, M., Gaveau, N., & Steffenel, L. A. (2022). Computer Vision and Deep Learning for Precision Viticulture. Agronomy, 12(10), 2463. https://doi.org/10.3390/agronomy12102463
- Chen, W., Li, Y., Tian, Z., and Zhang, F., 2023. 2D and 3D object detection algorithms from images: A Survey. Array, 19, p. 100305. DOI: 10.1016/j.array.2023.100305.
- Bilous, N., Malko, V., Frohme, M., & Nechyporenko, A. (2024). Comparison of CNN-Based Architectures for Detection of Different Object Classes. AI, 5(4), 2300-2320. https://doi.org/10.3390/ai5040113
- Gragnaniello, D., Greco, A., Saggese, A., Vento, M., & Vicinanza, A. (2023). Benchmarking 2D Multi-Object Detection and Tracking Algorithms in Autonomous Vehicle Driving Scenarios. Sensors, 23(8), 4024. https://doi.org/10.3390/s23084024

- Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., & Tian, Q. (2019). CenterNet: Keypoint Triplets for Object Detection. arXiv:1904.08189 [cs.CV]. https://doi.org/10.48550/arXiv.1904.08189
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2020). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. arXiv:2010.11929. https://doi.org/10.48550/arXiv.2010.11929
- Wang, C.-Y., Bochkovskiy, A., & Liao, H.-M. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arXiv:2207.02696. https://doi.org/10.48550/arXiv.2207.02696
- Hong, Zhonghua & Yang, Ting & Tong, Xiaohua & Zhang, Yun & Jiang, Shenlu & Zhou, Ruyan & Han, Yanling & Wang, Jing & Yang, Shuhu & Liu, Sicong. (2021). Multi-Scale Ship Detection from SAR and Optical Imagery via A More Accurate YOLOv3. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. PP. 1-1. 10.1109/JSTARS.2021.3087555.
- Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., Han, J. and Ding, G., 2024. YOLOv10: Real-Time End-to-End Object Detection,https://arxiv.org/abs/2405.14458
- Kateb, F., Monowar, M.M., Hamid, M.A., Ohi, A. and M., P.D., 2021. FruitDet: Attentive Feature Aggregation for Real-Time Fruit Detection in Orchards. Agronomy, 11, p.2440. https://doi.org/10.3390/agronomy11122440
- Caesar, H., Bankiti, V.K.R., Lang, A., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G. and Beijbom, O., 2019. nuScenes: A multimodal dataset for autonomous driving. *arXiv* https://doi.org/10.48550/arXiv.1903.11027
- Shu M., Shen Y., Lin M.C. & Goldstein T., 2021. Adversarial differentiable data augmentation for autonomous systems. IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, pp. 14069-14075. DOI: 10.1109/ICRA48506.2021.9561205.ISO/IEC 23053:2022
- Son, J., Jinhwan, & Jung, H., 2024. Teacher–Student model using Grounding DINO and You Only Look Once for multi-sensor-based object detection. Applied Sciences. 14. 2232. 10.3390/app14062232.
- Jiang, S., Qin, H., Zhang, B. & Zheng, J., 2020. Optimized loss functions for object detection: A case study on nighttime vehicle detection. https://arxiv.org/abs/2011.05523
- Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. and Zitnick, C.L., 2014. Microsoft COCO: Common Objects in Context. In D. Fleet, T. Pajdla, B. Schiele and T. Tuytelaars, eds. *Computer Vision – ECCV 2014*. Lecture

Notes in Computer Science, vol. 8693. Cham: Springer, pp. 740-755 https://doi.org/10.1007/978-3-319-10602-1_48

- Kvietkauskas, T. and Stefanovič, P., 2023. Influence of training parameters on real-time similar object detection using YOLOv5s. Applied Sciences, 13(6), p.3761. https://doi.org/10.3390/app13063761
- Henderson, P. & Ferrari, V., 2016. End-to-end training of object class detectors for mean average precision. https://arxiv.org/abs/1607.03476
- Zhang, H. and Zhang, S., 2024. Shape-IoU: More accurate metric considering bounding box shape and scale. arXiv preprint arXiv:2312.17663. https://arxiv.org/abs/2312.17663
- Zhang, X., Zhang, T., Yang, Y., Wang, Z. and Wang, G., 2020. Real-time golf ball detection and tracking based on convolutional neural networks. In: Proceedings of the 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Toronto, ON, Canada, 2020, pp. 2808-2813. https://doi.org/10.1109/SMC42975.2020.9283312
- Li, X., Wang, W., Wu, L., Chen, S., Hu, X., Li, J., Tang, J. and Yang, J., 2020. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. arXiv preprint arXiv:2006.04388. https://arxiv.org/abs/2006.04388
- Ocampo, D., Posso, D., Namakian, R. and Gao, W., 2024. Adaptive Loss Weighting for Machine Learning Interatomic Potentials. https://arxiv.org/abs/2403.18122
- Terven, J., Cordova-Esparza, D.-M., Ramirez-Pedraza, A. and Chávez Urbiola, E., 2023. Loss functions and metrics in deep learning: A review. https://doi.org/10.48550/arXiv.2307.02694