Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Evaluating the Generalisability of Convolutional Neural Networks Based Models for Detecting Obstructed Traffic Signs

by
Tudor Stefan Haba
2420645

Supervisor: Mark Huntly
Second Marker: TBD

Project submitted as part of the requirements for the award of MSc Software Engineering

January 2026

Table of Contents

1 Introduction 4

1.1 Motivation 4

1.1.1 Research question 5

1.2 Aim and Objectives 5

1.2.1 Aim 5

1.2.2 Objectives 5

2 Literature Review 6

2.1 Autonomous Vehicles 6

2.2 Computer Vision 8

2.2.1 Data preprocessing 10

2.3 Neural Networks 10

2.4 Convolutional Neural Networks 12

2.5 Object Detection Models 14

2.5.1 YOLO 14

2.5.2 ResNet 19

2.5.3 DenseNet 20

2.6 Data augmentation 21

2.7 Previous Research 23

2.7.1 Traffic Sign Detection and Recognition (TSDR) System 23

2.8 Chapter summary 24

3 Methodology 25

3.1 Hardware, Software & Data 27

3.2 Implementation 28

3.2.1 Augmented Dataset 28

3.2.2 YOLO Model 31

4 Results & Discussion 33

4.1 Results 34

4.2 Discussion 36

5 Limitations & Future Work 40

5.1 Limitations 40

5.2 Future Work 41

6 Conclusion 41

7 Appendix 43

8 Bibliography 50

Abstract

The current rise in autonomous vehicles (AVs) has led to significant research into the safety of in-board computer vision systems meant to improve the safety of the car. Most AVs use camera-based systems to capture data and feed it into the decision-making algorithm. A common object detection algorithm used by AVs is YOLO (You Only Look Once). As the name suggests, the model can look at an image once and detect all the objects within it along with their location and what each object is. The model is chosen for its accuracy and fast object detection speeds, making it a reliable model to use. While studies show that cameras have good performance under perfect conditions, they have also shown there are certain condition where they struggle which include adverse weather and night-time.

Based on the future work identified by previous studies, this study has trained a YOLO model on an augmented dataset to be able to identify and classify partially obstructed traffic signs. The augmented dataset includes multiple scenarios such as rain, fog or night. The model was compared against a model that has only been trained on a perfect dataset, and the results have shown that the augmented model has outperformed the original model in every evaluation it has been tested against. The augmented model has achieved a 92% accuracy on a test dataset made up of unobstructed traffic signs and an 84% accuracy on a test dataset made up of obstructed traffic signs. The study demonstrates that training a YOLO model against an augmented dataset made up of obstructed traffic signs has improved the generalisation on unseen data.

1 Introduction

1.1 Motivation

Convolutional neural network (CNN) models are a good way to identify and classify images based on the features the model learns from them. They have proven to work very well in multiple real-world industries such as medical and more recent in the autonomous vehicle industry [1][2]. The autonomous vehicle industry is a fast-growing industry gaining a lot of interest in the media, especially since some manufacturers such as Tesla are already using their autopilot vehicles on roads in America. Although, some other manufacturers such as Waymo are using different technologies for their self-driving taxis, Tesla is known to be using CNN models as part of their decision-making software [3]. Models such as YOLO (You Only Look Once) have made it possible for self-driving cars to be able to detect objects in front of them with a good accuracy and a fast detection rate. There is a clear rise in self-driving vehicle technologies that rely on systems like CNNs in the background, which process data from outside cameras that gather environment information from around them, to determine appropriate driving actions. Recent news show that the UK government is considering allowing testing of autonomous vehicles on the UK roads [4]. However, the question remains about how accurate a model could be in more obscure environments and with muddled data, such as obstructed traffic signs, where the model must be able to generalise to unseen data that differs from the training dataset. The technology must be safe enough to be on the road for the safety of pedestrians and other people on the road. The road could have a lot of variants that could cause some indecisiveness in CNN models, adverse weather which is a lot more common in the UK, could also affect the accuracy of the model at detecting traffic signs. In [2] the authors have examined the extent to which CNN models are used in autonomous vehicles, similarly, they used data augmentation to increase the size of the dataset. However, data augmentation could be used in many more instances than just increasing the available dataset. Data augmentation could be used to create completely new data that could simulate different weather events that could obstruct traffic signs. As identified by [5], data augmentation could improve the accuracy of deep learning models with partially obstructed signs. However, their study only focused on traffic signs obstructed by objects and does not examine how weather events, such as fog or rain, which can also partially block traffic signs, affect the generalisation of the models on unseen data.

1.1.1 Research question

The research question of this project is whether a CNN based computer vision model can maintain accurate traffic sign recognition when the signs are partially obstructed by weather events in real world scenarios. The project aims to evaluate and compare the performance of two CNN based models, that have been trained on different datasets, and how they behave when evaluated on unseen test data created using data augmentation. The objective is to see if data augmentation has an impact on the generalizability of the models. To test this the two models will be fed different data, one of the models will be trained with clear data of traffic signs, the ideal scenario that would be occurring on the roads, while the other model will be trained with an augmented dataset of traffic signs that adds more imperfection to the data like we would see in the real world. Differently from [2] data augmentation will not only be used to increase the training dataset size but it will also be used to simulate environmental obstructions affecting the visuals of traffic signs. This seeks to develop a more robust system at detecting real-life scenarios thus verifying the suitability to be used on roads.

1.2 Aim and Objectives

1.2.1 Aim

The overall aim of this project is to compare and evaluate whether data augmentation can help convolution neural network (CNN) models into more accurately classifying partially obstructed traffic signs based on real world scenarios. Analysing the accuracies of CNN models that have been trained with and without data augmentation by testing on unseen data.

1.2.2 Objectives

1. Critically analyse the current literature on autonomous vehicle systems and understand the current challenges posed by traffic sign detection on convolutional neural networks.

Understand and summarise the background research that took place in the self-driving vehicle industry and computer vision. Explain the difference between the different computer vision architectures such as YOLO, RESnet and DESnet and explain how the help autonomous vehicles in the decision-making process to identify different objects on the road focusing on traffic sign detection.

2. To develop an augmented dataset and train a convolutional neural network to detect obstructed traffic signs.

Crate an augmented dataset of images using different techniques to produce a series of real-world scenarios encountered by self-driving cars such as precarious weather and

time of day. The aim is to improve data preprocessing and model architecture to identify obstructed traffic signs with a high accuracy of above 70% on unseen test data to test the generalizability of the model.

3. Thoroughly evaluate the results to investigate whether improving data preprocessing has made an improvement in a convolutional neural network ability to identify and correctly classify obstructed traffic signs.

Test a convolutional neural network that has been trained on a non-altered dataset of traffic sign images and compare the accuracy with which it is able to identify obstructed traffic signs to a model that's been trained on an augmented dataset. How do the accuracies differ and has the data preprocessing helped the CNN model to correctly identify obstructed traffics signs.

2 Literature Review

2.1 Autonomous Vehicles

Autonomous vehicles are systems capable of perceiving the environment around them and navigating independently with no input required from a human driver while at the same time being able to make quick decisions and follow traffic regulations. These vehicles are designed to protect the pedestrians and the driver with a focus on driving safety and minimising the risk of traffic accidents [6]. The Society of Automobile Engineers (SAE) describes the six levels of automation, starting from level 0, indicating no automation, to level 5, denoting full autonomy. On the SAE scale, Fig. 1, for a vehicle system to be classed as a level 5 it has to be able to perform all driving tasks under all conditions, such as weather, without the need of any human input [7][8].

For level 5 to be achieved, a vehicle must rely heavily on sensors and cameras capable of accurately sensing the environment and capturing meaningful data, which is then fed into the decision-making model that controls the actions of the vehicle. Currently, Tesla is operating at a level 2 on the scale despite having one of the most advance self-driving pilots. As of now no vehicle is fully automated and operating at a level 5 [9]. The cameras are used for a 360-degree view of the surrounding environment, and the data gathered, such as images of traffic signs, is then processed by the on-board machine learning (ML) algorithm. However, one problem with cameras is the way they perform under adverse weather conditions and during night-time, making them unreliable under these conditions. The accuracies are always higher under the ideal environment they have been tested on. Other sensors commonly used by autonomous vehicles are LiDAR (Light Detection and Ranging) sensors. LiDAR works by projecting laser beams into the environment and calculating the time it takes for the laser to hit an object. Based on the

time it takes the model is able to map the surrounding environment and create a 3D representation of it. While LiDAR outperforms cameras in many scenarios it brings a major drawback to many manufacturers trying to keep costs down. The cost of adding the sensors to the vehicle compared to cameras is vastly more expensive [6]. The high cost of LiDAR makes many manufacturers rely on camera data instead.

The data gathered by the cameras can then be fed into computer vision algorithms to help the vehicle understand the surrounding environment. There are four different computer vision tasks applied by an autonomous vehicle: depth detection, lane detection, object detection and traffic sign detection [7]. Traffic sign detection is a critical task that allows the vehicle to not only detect there is a sign ahead but also interpret what the sign means and act accordingly. A misinterpretation of traffic signs can lead to dangerous driving decisions. In the real-world traffic signs could be partially obstructed due to weather events such as rain, fog or snow. The research in this paper touches on object detection and focuses on traffic sign detection with the aim of identifying whether data pre-processing affects the model generalisability on unseen obstructed traffic signs.

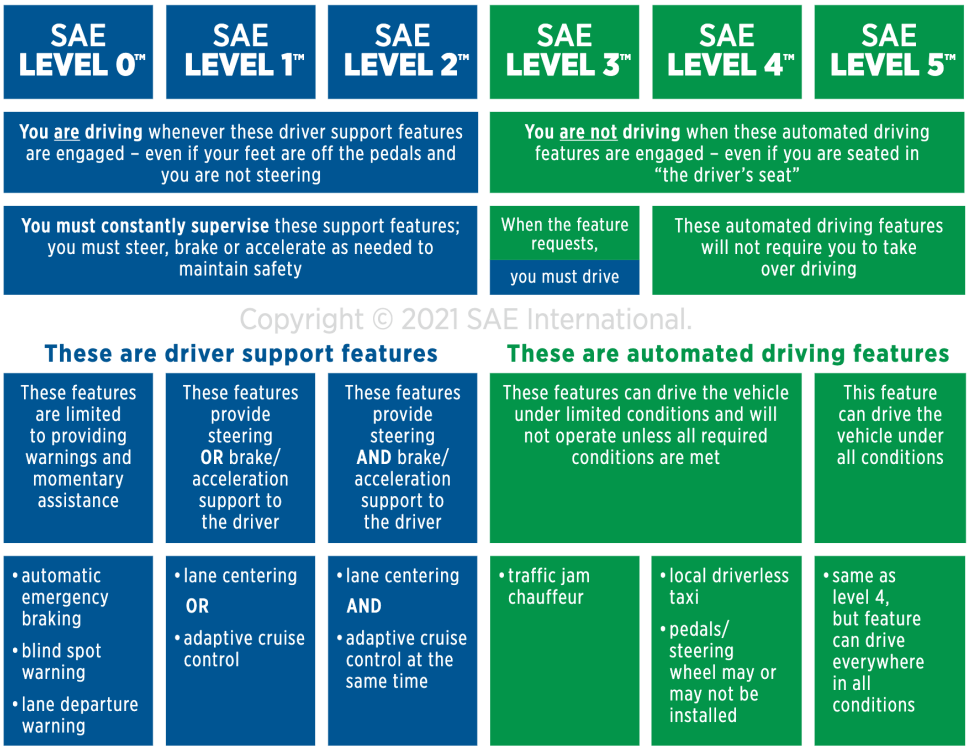


Fig. 1: SAE Levels of Driving Automation Scale [8]

2.2 Computer Vision

Computer vision is a research area within deep learning, which falls under artificial intelligence, that allows computer systems to extract features out of images and map the surrounding environment so that the computer can make informed decisions about it. Computer vision works through image processing and through being able to map exact location of objects within an image. It then uses deep neural networks to train and understand what the image represents. Each computer vision task requires a different approach as the complex mathematic equations behind them are different depending on the task given, whether that is object detection, image classification or image segmentation [10]. Through computer vision systems, computers are able to interpret and understand the physical world similar to how humans perceive it.

There is a difference between the early days of computer vision, where a human would have to interact with the algorithm to extract essential features out of images and then pass the data to a shallow neural network, Fig. 2, and modern deep learning computer vision models. The deep learning models are based on deep neural networks, composed of convolutional neural networks (CNNs), referred to later in the paper, with many convolutional layers that are able to extract the features out of images automatically and it requires little human involvement [10]. The crucial step that still involves human intervention is data pre-processing, finding and labelling a dataset correctly for the algorithms to learn from, Fig. 2.

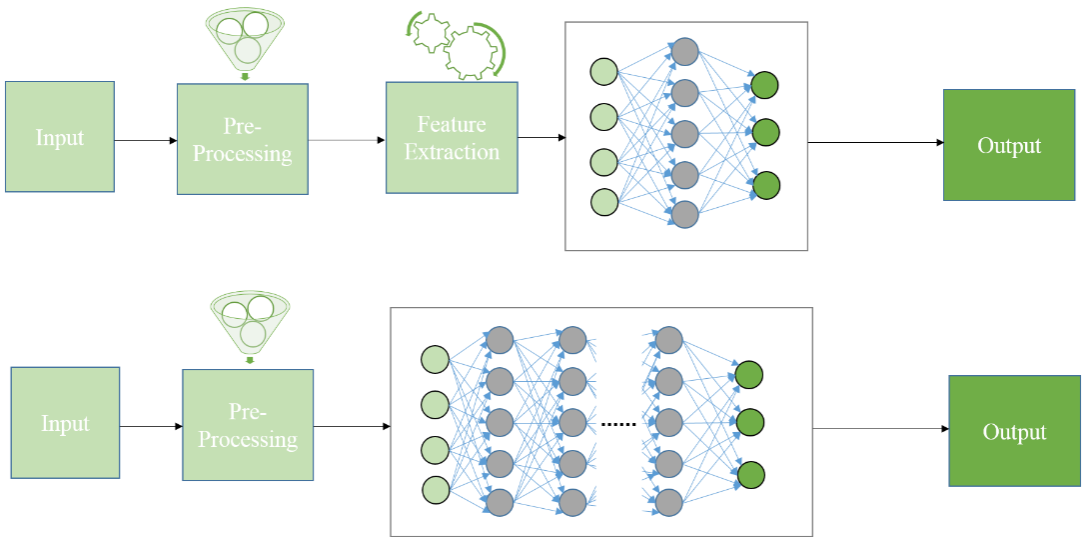


Fig. 2: Comparison between a shallow neural network and deep learning [10]

The shift from the shallow neural networks to deep learning has allowed computer vision to work within complex real time environments such as autonomous vehicles where the aim is to improve safety. Paired with CNNs, computer vision plays a major role into the decision-making process by enabling object detection, segmentation and recognition [10]. Computer vision is an important factor in how the autonomous vehicles perceives

the environment around them. The key tasks performed by computer vision within a self-driving vehicle are:

- Object detection and classification, computer vision is able to identify and classifying various objects, such as, obstacles, pedestrians and traffic signs using the various cameras around the vehicle. They make use of CNN models to verify and check how far away from the vehicle the object is, this allows the systems to respond appropriately to a possible incident. As part of object detection, computer systems are able to identify objects and obstacles. CNNs are then used to extract features, which are used to create a 3D map, that allows the vehicle to identify the object and take corresponding actions based on what it was able to visualise. This is an extremely important feature in real world environments to ensure the safety of everyone on the road [11].
- Lane detection and tracking, autonomous vehicles use computer vision systems to detect the road markings positions and keep the vehicle within the lanes while driving or while changing lanes.
- Semantic segmentation is a complex part of a computer vision system used to better gauge the environment around the vehicle. It maps every pixel within the image to a class such as road, tree, person, to enable better decision making in more complex environments.
- Traffic sign recognition, in order to make sure the autonomous vehicle follows road regulations and the law computer vision systems are used to interpret various traffic signs [11]. However, since most autonomous vehicles are using cameras to gather data, there is potential of various issues with relying on cameras for such important tasks. One example would be obstructed traffic signs due to branches or adverse weather, such as fog or heavy rain, blocking parts of the sign.

Some more advanced autonomous vehicle systems, such as drive pilot, make use of computer vision to sense and understand the environment around them while also making a map to estimate the position of the vehicle, this allows them to be able to operate on their own for speeds of up to 40 mph. This is achieved using vision based path planning and vision based mapping and localisation to autonomously plan the trajectory and location of the vehicle [7]. The most commonly used method of gathering data for computer vision are cameras, they are cheap compared to other sensor and have a good performance under perfect conditions. The images gathered by the cameras are then fed into neural networks, most commonly a convolution based neural network, to help aid the vehicle into taking the most appropriate decision. However, weather massively affects the performance of cameras making them very unreliable under harsh conditions. Cameras also operate on visible light so only using cameras during night or in low light environments makes it a challenge and it affects the reliability of the data [7] [11].

2.2.1 Data preprocessing

Data preprocessing is a vital step for a computer vision algorithm to be able to read and understand the data its being fed. Fig. 2, shows that data preprocessing as a process has been around since shallow neural networks were still in use. Data preprocessing involves preparing the data for the algorithm and it's a process that cannot be fully automated due the complex nature of data and the importance it plays in the performance of a computer vision task. Part of preparing the data involves data cleaning, where any outliers and noisy data is identified and decided what techniques are applied in order to fix the data from deleting any missing data to applying different methods that reduce noise within the data [12]. In supervised computer vision tasks data preprocessing also includes data labelling. This is when data, images, is hand annotated with ground truth values such as class labels, bounding boxes, ensuring the model could learn from the input data [13]. The data preprocessing pipeline also includes:

- Data integration, bringing together data from multiple sources into a single space
- Data transformation, converting the data to a suitable format for the model that's being trained
- Dimensionality reduction, reducing the dimensions within the data and making it harder for the model to overfit.
- Handling class imbalance, by using different techniques such as under-sampling and oversampling to reduce the effect an imbalance class would have on the model's performance. [12]

2.3 Neural Networks

A neural network (NN) is a machine learning model that tries to simulate the way a human brain would work to process information [14]. A NN is made up of different layers, starting with the input layer, one or more hidden layers which decide the outcome of the data and a final output layer. A traditional neural network processes the information through the hidden layers of interconnected nodes all of which have a weight and a threshold value. Each node will then take the data and apply the weighting to it, if the resulting output is above the threshold value, then the node is called an active node and it can pass data down to the other layers, Fig. 3. Some neural network architectures do not pass the data forward if it has not met the threshold value while others replace the threshold value by an activation function that calculates the output of an node and passes it forward[14][15].

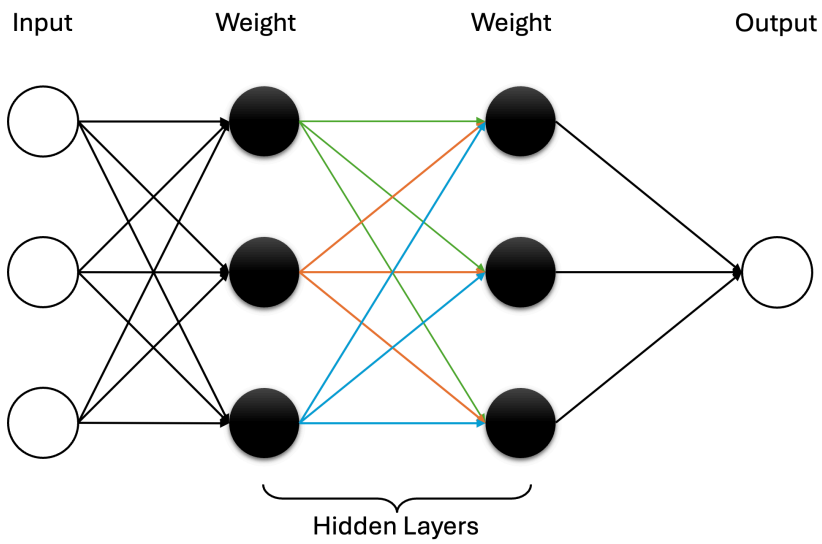


Fig. 3: Simple representation of a neural network

The way a NN works is by providing it with some input data and applying the formula from Fig. 4, with the resulting outcome getting passed into each node and checked against the threshold value, which typically has a value between 0 and 1, to determine which of the nodes will get to pass the data forward. This happens for every layer until we get an output set of data. Neural networks are good at learning patterns from datasets and they are good at improving fault tolerance over time, meaning that small errors in data do not have an effect to the output [14]. The project will make use of fault tolerance to initially create a model which recognises traffic signs and then test the model with a different dataset that introduces a various number of faults, seeing how well the model is at tolerating inaccuracies in the images. The project intends to use a specialised neural network, a convolution neural network (CNN), to process the image datasets and to extract the features of the images in order to identify the traffic signs.

$$\sum w_i x_i + bias = w_1x_1 + w_2x_2 + w_3x_3 + bias$$

$$output = f(x) = 1 \text{ if } \sum w_1x_1 + b >= 0;\ 0 \text{ if } \sum w_1x_1 + b < 0$$

Fig. 4: Formula to calculate the output of a neural network node from its input

2.4 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a type of neural network focused around solving deep learning problems specialising in pattern recognition. CNNs are great at extracting features out of image and audio input datasets and classifying them. They have been making a massive advancement in fields such as facial recognition and autonomous driving [16]. CNNs are complex neural networks made up of an input layer, one or more convolution layer followed by a pooling layer and more fully connected layers (FC) before producing an output, Fig. 5.

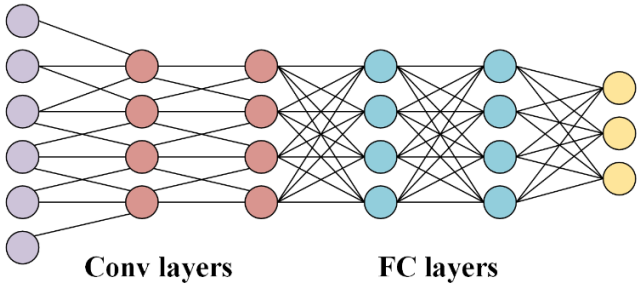


Fig. 5: Example of a CNN model with convolutional layers and FC layers [16]

Convolutional layers are made up of filters. A filter or a feature detector is a matrix, Fig. 6, that goes on top of the image and multiplies the pixel values from top left to bottom right until a new matrix is produced [16][17]. The resulting matrix is a score of how closely the pixels in the image match the pattern of the filter and is the activation map of the given image. The role of the filter is to find matching patterns in the image while at the same reducing the amount of data the network has to process. Using different feature detectors produces different activation maps which can be used by the CNN to identify different features in an image. The activation maps combined together produce a convolution layer that is used by the model to understand larger parts of the image [17].

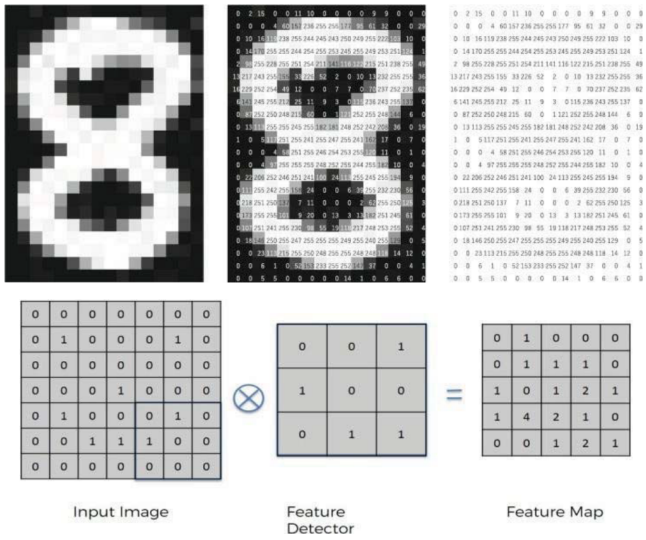


Fig. 6: Detecting features out of an image using a feature detector matrix [17]

It is common for CNNs to use a pooling layer after every convolution block. A pooling layer reduces the size of the image and can be achieved in many different ways such as max pooling, average pooling, stochastic pooling, spatial pyramid pooling. Max pooling being the most commonly used in CNNs [17]. Pooling works in a similar way that filters work, a matrix is slid on top of the image and depending on the method chosen a result is obtained, however, the pool filter does not cross on pixels that it has already worked on, thus being able to reduce the size of the image. A 2 x 2 max pooling for example, Fig. 7, will take the maximum pixel value and only this value will be now present in the resulting feature map. This way pooling only keeps the important features of an image discarding the rest and reducing the size of the image for the next layers thus reducing the amount of time the training will take overall [17] [18]. Pooling also helps resolve the overfitting problem that could happen to some models. This is when the model learns the training dataset and gives high accuracies only because it has memorised the images but fails to find patterns in new unseen images, resulting in poor model performance and poor generalisation on unseen data [19]. CNNs flatten the last feature map before passing it to the fully connected layers to classify the image. This means that from a 2D or a 3D array the feature map gets converted into a 1D vector that can be understood by a traditional neural network. The final flattened vector only contains the most important features of the image [1]. Once the vector gets passed into the fully connected layers, the NN can make a decision about what it thinks the data should be classified as based on the learned features.

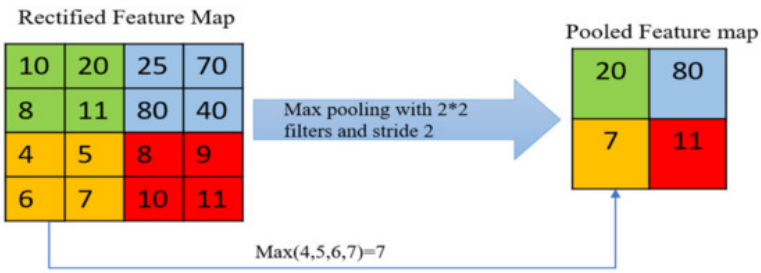


Fig. 7: Max pooling being applied to a feature map [18]

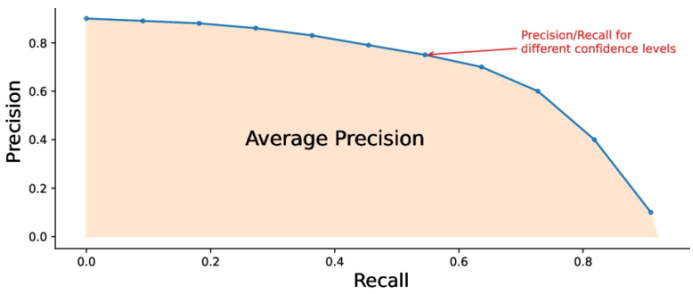
Two CNNs models will be used in the project. The first model will be trained to identify traffic signs, and all the training data will have clean, unobstructed traffic signs. To test the model the dataset will be split into 80/20, with 80% of the data used for training and the rest of 20% used to validate and evaluate the model. The test data to thoroughly evaluate the models will come from a different dataset. The second model will be trained with an augmented dataset that aims to introduce some obstruction in the images. Different architectures for the models will be tested until both models have a high accuracy of above 80% for the one not trained on the augmented data and 70% for the augmented data trained model.

2.5 Object Detection Models

2.5.1 YOLO

You Only Look Once (YOLO) is a real time object detection model based on a CNN architecture. YOLO has been applied in various other fields for its fast ability to detect objects in real time. Some examples include medical imaging for cancer detection, in agriculture to spot various crop diseases and pests. Robotics and the autonomous driving industry are the most relevant fields, to this study, in which YOLO has been used before. It is used to identify and classify various obstacles on the roads such as people, vehicles and other objects like traffic signs, as they appear on the road [20]. This works by using multiple cameras positioned around the vehicle to capture live data from the environment and feed the data back to the incorporated model to help assist the car in the decision-making process. This real time object analysis allows the autonomous vehicle to respond quickly to dynamic road conditions, such as a pedestrian crossing, slowing down traffic or unexpected objects on the road, this is discussed further in the YOLOv1 section. YOLO plays a major part in some autonomous vehicles manufacturers, to help aid the vehicle and improve the overall safety of pedestrians [21]. The name YOLO comes from its ability to detect objects in images with a single pass through the network of the model. The model is able to detect all the objects within an image, but it also detects the position where those objects are located. Unlike other models YOLO performs detection as a regression problem, where it directly predicts the bounding boxes coordinates and the probabilities all at once. This is a major improvement compared to previous classification models that would have to scan the same image hundreds of times which made them much slower in comparison [20][22].

Average Precision (AP) is a metric used to evaluate the performance of object detection models. It is based on precision-recall relationship, Fig. 8. Precision measures the accuracy of true positive predictions made by the model, how many of the model's total classifications were correct. Precision does not measure the model's accuracy on the whole dataset [20]. Recall is the measure that evaluates how many of the actual objects in the dataset were successfully detected by the model. For example, the model identifies that the traffic sign is a stop sign 7 out of 10 times, out of those 7 times, the sign was a stop traffic sign only 5 times. The precision formula will calculate the precision to be 5 out of 7, however, the recall will be calculated to equal 5 out of 10. There is often a trade-off between precision and recall [20]. Labelling more objects as stop signs will result in an increased recall, however, this can result in more false positives therefore lower precision. To determine the best results for the model, an AP curve would have to be used, where the area underneath the curve represents the AP score, Fig. 8.



$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Fig. 8: Average Precision graph and formula for Precision and Recall [23] [24]

Yolov1

YOLOv1 was the first version of YOLO models, and it was the first object detection model that would be able to detect objects from an image in real time with a single pass through the network. In computer vision, object detection works by identifying the objects present in an image and putting them inside their own 'bounding boxes' [20][25]. These bounding boxes are rectangular frames that show exactly where the objects are present within the image, Fig. 9.

YOLOv1 detects all the bounding boxes in an image simultaneously along with all the probabilities. To do this the model splits the given image into a S x S grid [20]. Each grid cell is responsible for predicting B bounding boxes along with the confidence score, C. The confidence score is the model's confidence that the object is present within that bonding box. Each grid cell can predict more than one bounding box and YOLO picks the bounding box with the highest IoU (Intersection over Union). The IoU calculates and compares how much the predicted bounding box overlaps the actual ground truth box, Fig. 9, a hand labelled bounding box which provides the location of the actual object for the model to train on. YOLOv1 handles object detection as a regression problem. Instead of first detecting part of an image and then classifying it, the model directly predicts the bounding box coordinates and size (x, y, width and height) and what the object is [20][26]. The model might say *'at position x and y, there is a box of size w and h, and the object is a stop sign'*. This is different from other CNN models that split up object localisation and object detection into separate steps.

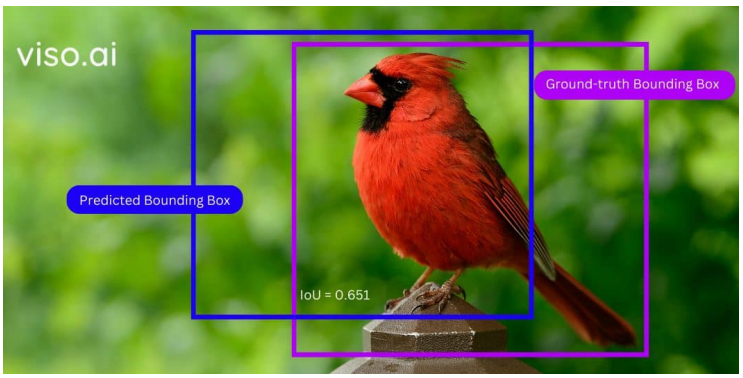
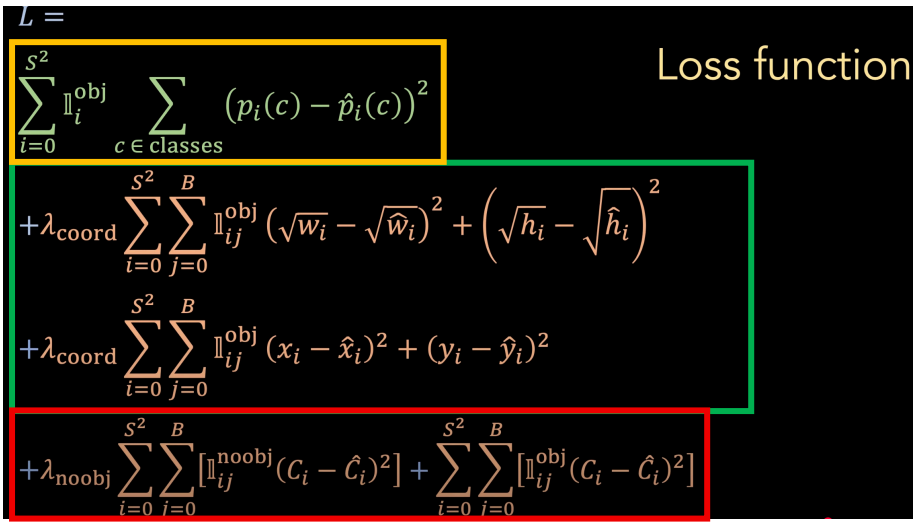


Fig. 9: Image showing a model's predicted bounding box, the ground truth box and the calculated IoU [27]

YOLOv1 uses a specifically designed loss function to measure how wrong the model's predictions are from the ground truth values. During training, the model adjusts its parameters based on the total loss function, Fig. 10, to improve the accuracy of future predictions. The goal of the loss function is to minimise the difference between the model's predictions and the actual ground truth. Fig. 10, shows the total loss function that is made up of 3 different components [20][22].

- Classification loss, outlined in orange, checks whether the model's prediction of what the object is accurate against the ground truth.
- Localisation loss, outlined in green, checks how far the model's prediction for the position of the bounding box is from the actual position of the ground truth bounding box.
- Confidence loss, outlined in red, compares the confidence score predicted by the model to how accurate the prediction actually was, this is based on the IoU.

From the loss function the model learns to predict both the object classification and the bounding box coordinates, if an object is present in a grid cell. If no object is present on the cell, then it learns to predict a low confidence score. This allows YOLOv1 to focus the training on relevant cells where objects are more likely to be present [20].



$$L = \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c)\right)^2$$
$$+\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2$$
$$+\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2$$
$$+\lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} [\mathbb{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2] + \sum_{i=0}^{S^2} \sum_{j=0}^{B} [\mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2]$$

Fig. 10: Loss function formula [28]

<u>Architecture</u>

The architecture of YOLOv1 is made up of 24 convolutional layers followed by 2 fully connected layers, Fig. 11. The convolutional layers are used by the model to extract important features from the dataset, they will look at patterns, edges and shapes from the image. The last 2 fully connected layers are responsible for predicting the bounding box coordinates and the object class probabilities. All the convolutional layers in the model use a leaky ReLU activation function with the exception of the last fully connected layer which uses a liner activation function, meaning it passes the input to the next layers

unchanged to predict the output [22]. A leaky ReLU activation function is a type of non-linear activation function. The use of leaky ReLU prevents the vanishing gradient problem, discussed further in the literature review, which improves the training of the model by allowing small positive outputs on negative inputs [29].

The model architecture starts with a convolutional layer using a leaky ReLU activation function, followed by a max pooling layer, Fig. 11. Layers 3 and 4 have the same structure as the first two layers. This is then followed by a series of four convolutional layers and another max pooling layer. After that, another series of four convolutional layers followed by a max pooling layer. The last set of layers consists of six convolutional layers, ending with 2 fully connected layers. The feature extraction network, the convolutional layers, in a YOLO model are called the backbone of the model and are typically pre-trained. The model alternates between 3x3 and 1x1 convolutional layers [22]. The 3x3 convolutional layers are responsible for extracting standard features from the image, while the 1x1 layer purpose is to reduce the depth (RGB layers) of the image. This depth reduction helps save computational power for the following set of layers. In a YOLO model this is called the bottle neck of the model, and its purpose is to reduce computational complexity and the number of parameters in deep networks while preserving essential features. The max pooling layers are there to reduce the overall size of the image taking only the biggest value out of the grid and creating a new feature map out of the max values. This helps reduce the height and width of the image as it moves through the network [22].

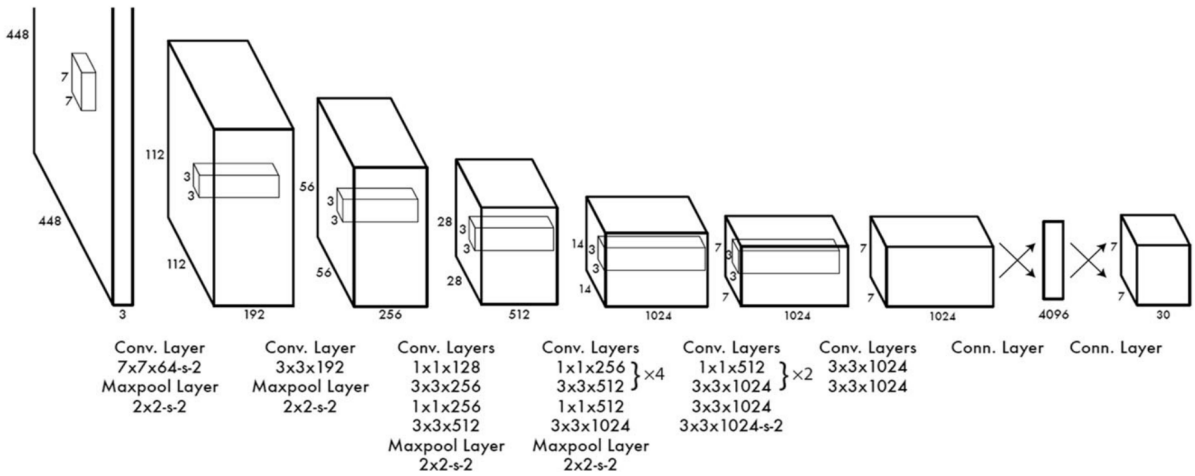


Fig. 11: YOLOv1 Architecture [22]

YOLOv2

YOLOv2 introduces anchor boxes to improve the bounding box fitting and uses a new backbone, DarkNet19. This allows for new features such as batch normalisation and higher resolution input images to enable better feature extraction. These improvements allow it to classify more classes compared to YOLOv1 [25].

YOLOv3

YOLOv3 uses a deeper backbone, DarkNet53, and introduces multi-scale detection which allows it to detect small and large objects more effectively in an image. This version replaces the softmax classifiers with logistic classifiers enabling multi-label classification [25] [30].

YOLOv4

YOLOv4 is created by different developers, and it comes with multiple enhancements which include improved detection accuracy, speed and computational efficiency. It uses a CSPDarkNet backbone to reduce computational bottlenecks. YOLOv4 uses the improved Mish activation function, and it applied advanced training techniques such as mosaic augmentation and CIoU loss. All the improvements allow the model to train faster on standard GPUs while maintaining high performance [31].

YOLOv5

The language of YOLOv5 was switched to use PyTorch, which makes it easier to train, and it makes it more available to use by more people. The model benefits from a smaller size which allowed more people to experiment with it. It also benefits from added features such as automatic learning anchor boxes and mosaic data augmentation which improves the detection efficiency [25].

YOLOv6

YOLOv6 introduces several architectural modifications, new training strategies, and optimized loss functions, with a strong focus on real-world applications such as real time surveillance and autonomous systems. This version of YOLO uses anchor free architecture, an EfficientRep backbone and new RepPAN neck, with advanced loss functions, which result in a more efficient and optimised model that could be used on more devices such as mobiles [25] [32].

YOLOv7

YOLOv7 added new scaling methods along with new training strategies, including E-ELAN, which enhances feature learning and efficiency. As a result, the model's performance was more efficient and at the time of its release it was the fastest and most accurate YOLO model [25].

YOLOv8

YOLOv8 is the model that will be used going forward for this project because of its advances compared to the other models and its availability. YOLOv8 introduces a fully anchor free detection head, supporting instance segmentation and better small object detection [33]. This is particularity useful for this experiment as traffic signs can be relatively small depending on the range the car system is picking them up from. Better detection of small objects would result in better performance during the testing phase. The anchor free design is important as it takes away extra work the model was doing in previous versions. Previous versions used anchors, a set of pre-defined shapes, where the model would try to fit those anchors around different objects within the image and chose the one that fit best, highest IoU. Then it would predict a bounding box adjusted to fit the object from the anchor size. With YOLOv8, there is no need for the predefined anchors, instead, when the model detects an object, it would predict the bounding box size and shape directly, to fit around the object [34][35].

2.5.2 ResNet

Using convolutional neural networks for deep learning models has proved challenging because of the vanishing gradient problem which made it harder for models to properly train. The vanishing gradient problem is a major challenge, and it happens when the traditional neural networks become too deep and the gradient becomes extremely small during back propagation. This causes the model to stop updating the weights correctly therefore causing the deeper layers within the model to stop learning from the data [36].

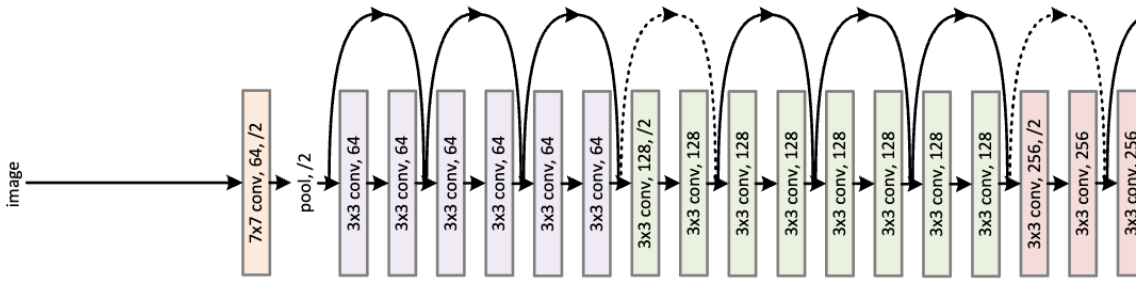


Fig. 12: A ResNet network being made up by residual blocks [37]

The challenge of the vanishing gradient was addressed when ResNet was released in 2015, it introduced the idea of residual connections, which allowed some of the data to bypass some of the layers and be fed directly into the deeper ones. These connections made the learning process of the network take less time and be more efficient, thus, enhancing the overall performance of the network. These connections work by allowing information to be passed from previous layers onto later layers in the architecture by skipping one or more levels. The ResNet architecture is made up of residual blocks, Fig.

12, which include convolutional layers, a batch normalization layer and a ReLU activation function [38]. Each block has residual connections which are made up of a shortcut connection, able to skip layers, and a normal connection that moves through all the layers within the network. The output of the block is produced by combining the output of the shortcut connection with the output from the normal connection after which a ReLU activation function is applied to generate a feature map that can be passed onto the next layer in the network [36] [39]. Instead of learning a full mapping from input to output, ResNet, learns the residual, which is the difference between the input and the desired output of the block, which is easier to learn compared to a full function. ResNet is mainly composed out of convolutional layers used to extract the features out of input images in order to classify them. ResNet also makes use of batch normalization, to speed up the training process and make it more stable [39]. Batch normalisation is applied after every convolutional layer to help stabilise the learning process by normalising the inputs at every layers. This is achieved by adjusting the values of the input activations so they have a consistent scale and distribution, with zero mean and a unit variance of one [38]. It works by adjusting the values so that they are centred around zero and have a standard spread. This helps the network train faster and maintain more reliable gradients through the training process by reducing big changes in the gradient.

This not only solved the problem of the vanishing gradient, but it allowed for even deeper models to be created. Some other models have been built on top of ResNet such as DenseNet to enhance the performance for various computer vision tasks such as object detection and picture classification [36].

2.5.3 DenseNet

DenseNet (Densley Connected Convolutional Networks) is a deep learning model based on ResNet, therefore it fixes the vanishing gradient problem as well, and it uses CNNs to extract features out of images and classify them. In DenseNet all of the previous layers are fully connected to the next layers, allowing the information to flow freely and more efficiently through the network. Due to all the network being fully connected, DenseNet is better at learning and understanding the features of images [40].

Proposals for DenseNet came from the challenges that were left unsolved by previous models, such as ResNet. These challenges included low recognition efficiency, poor recognition accuracy, sparse feature extraction and overly complex classifiers, which all led to the image classification to perform poorly in real environments. This meant that the range of situations where the application could be used was limited. DenseNet is meant to solve these problems by using dense connections, every layer connected to each other, and the promotion of feature reuse, which reduces the complexity of the model. This structure, that allows for extensive feature reuse, means that the networks have

access to all the feature of all the previous layers. As a result, DenseNet has a better feature representation and learning efficiency with a simpler architecture and a smaller number of parameters, Fig. 13 [40].

The architecture of DenseNet, Fig. 13, begins with an initial convolutional layer that processes the input data, followed by a series of dense blocks which are separated by transition layers. The transition layer is made up of convolution layers and a pooling layer where the feature map's dimensionality undergoes a reduction operation, and the final size of the feature map is reduced by half. The dense blocks, Fig. 13, are composed of multiple layers, starting with a batch normalization (BN) function followed by a ReLU activation function which is followed by a convolutional layer. Within the dense blocks, each of the layers are densely connected with each other, allowing for the feature map from previous layers to be passed to later layers [40].

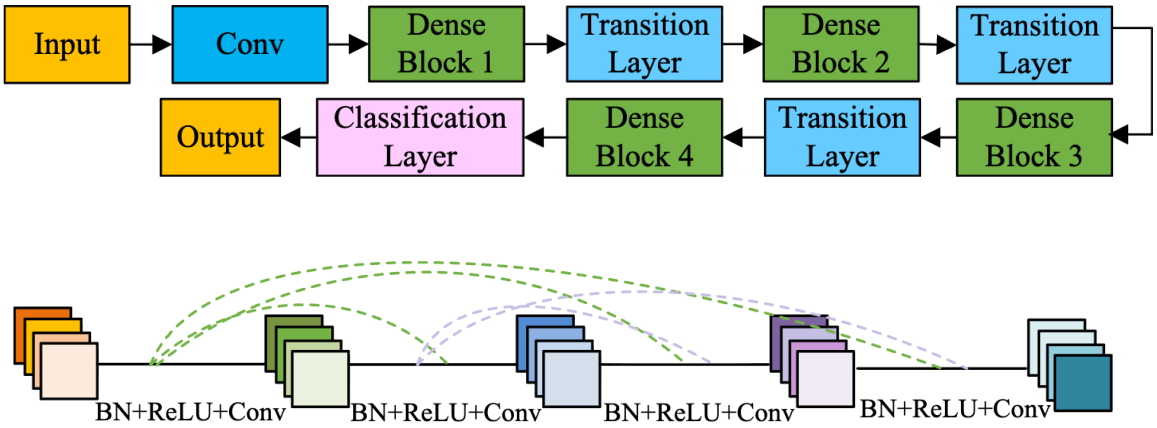


Fig. 13: Simplified DenseNet structure diagram and Simplified Dense Block diagram

However, many deep neural network models, including DenseNet, increase the depth and the width of the network to achieve higher accuracy in object detection. This adds multiple parameters and affects the computability cost of the model, making it unusable for real time object detection present in autonomous vehicles, even with powerful GPUs. Other problems present in DenseNet include challenges with gradient propagation in deeper models, not supporting multi scale feature integration and overall high computational costs which come from the network being densely connected. While, the model tries to fix these using convolutional layers and pooling layers they are still present in deeper networks [41].

2.6 Data augmentation

Data augmentation is a technique to artificially generate more data to train a model on from an already existing dataset. It is beneficial in the case the dataset is not large enough to train the model or if it would take a long time to gather new data. Data augmentation

improves the overall performance of the model training on it. The way data augmentation works is by taking existing data and applying different convolutions and filters to modify the data and produce new samples. The new samples that are being produced will have the same corresponding label as the input sample [42][43]. There are a number of techniques that could be used to augment a set of data in the case of images:

- Geometric transformations are a set of operations that include image rotation, image scaling and flipping. Image scaling being the most useful one for the case of traffic signs, it could help the model by zooming in and out of an image giving a more realistic interpretation of how a human would perceive a traffic sign at different distances [44].
- Colour augmentation allows for a shift in the image's colour or a change in brightness [42]. The change in an image's brightness could mimic the time of day the picture was taken, forcing the model to focus on specific feature of a traffic sign rather than its colours.
- Kernel filters are convolutional operations that could be applied to an image to highlight certain features or hide them. A Gaussian blur, Fig. 6, could be applied to an image to hide some of its distinctive feature and make it more challenging for a model to identify thus training it on more niche outliers [44].
- Random erasing, this is when random parts of the image get masked by rectangles in order to reduce the number of features, Fig. 6. This in result will enhance the performance of the model by being able to recognise and classify some of the harder images with missing information [42].

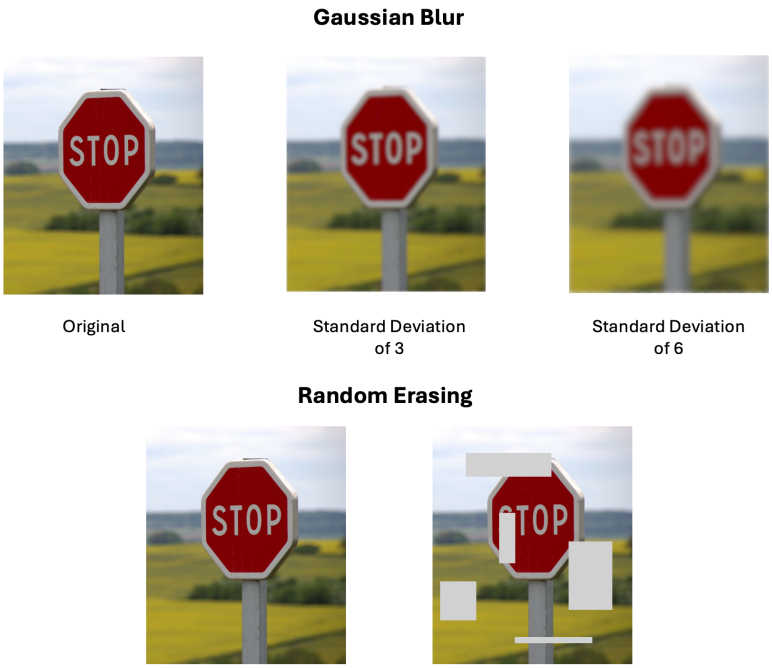


Fig 6. Data augmentation techniques visualised

The project would make use of a combination of data augmentation techniques to modify the dataset the second CNN model will be trained on. Using a combination of one or more techniques the dataset could achieve some more unique real-life scenarios and test whether the model can identify the traffic signs with real life circumstances. One of the real-life events the project wants to train and test the model on would be foggy weather. Using the filters that could easily be simulated by making use of the Gaussian blur and potentially making use of image scaling to make the traffic sign appear further away. Another scenario the project wants to test would be partial sign recognition. Using the random erasing which could simulate tree branches being in the way of the sign or the traffic sign being dirty or defaced and harder to read. Once the augmented datasets are generated and the new data is available, it is intended to split the dataset into 80/20 with 20% being kept for validation purposes. Out of the test data the same image will be fed into both models, the one not trained on the augmented dataset and the one that is specifically trained with the new dataset. This is done to make a comparison between the accuracies and analyse what the models classifies the data as.

2.7 Previous Research

2.7.1 Traffic Sign Detection and Recognition (TSDR) System

There have been a number of previous research papers looking at the traffic sign recognition problem. One of them being [2] where they proposed to improve on an already existing 7 layers CNN model called LeNet-5. The original model was made up of three convolutional and three subsampling layers followed by the fully connected layers producing the output. The model they proposed contained 12 layers with additional work on image preprocessing before feeding the data into the model to train on. Part of the image preprocessing techniques they suggested were colour to grayscale conversion, histogram equalization, unsharp masking, and normalization followed by data augmentation to increase the available dataset. While this project has a similar idea to the project proposed in this paper, the way data augmentation will be used is different. Where in [2] they used it to increase the number of images, the intention of this project would be to create new images containing some form of obstructions. Not only would this increase the size of the dataset, but the aim is to generate some images closer to real world data and harsher environment to improve the robustness of the CNN model. The project was successful with high accuracies in detecting and classifying the traffic signs, of above 99% [2]. As future work they have identified a need to create a model trained on different weather conditions and during different times of the day. This is how the proposed project in this paper intends to digitally alter images to create some of the scenarios stated in the above future work.

The work of [45] has identified multiple current challenges that TSDR systems face. Some of the challenges identified by the paper are variable lighting conditions. Traffic signs are harder to detect and classify when the lighting conditions vary and make it harder to see the sign in the image. This is partly due to the fact that during the pre-image processing phase approached in [45], they have used colour segmentation to reduce the features of an image. This combined with poor lighting conditions will also affect the features a model can identify from an image potentially deleting some of them. The paper has also identified that the condition of a traffic sign affects the model's ability to learn its features. The example they identified were traffic signs that have been vandalised and traffic signs that are partially obstructed. The random erasing technique will be able to simulate this scenario without the need to gather new data and do the labelling process of the data. Both papers [2] and [45] have identified weather playing a major factor into the ability of a model to detect traffic signs. The effects of weather such as rain on the TSDR system make it difficult to accurately detect signs and they lead to an increase in false sign recognition. A suggestion to overcome the weather-based problems were to enhance the quality of the image using pre-processing techniques to make the features in an image clearer for a model to identify.

2.8 Chapter summary

This chapter focuses on explaining the whole journey an autonomous vehicle system takes to identify a traffic signs. The chapter starts with explaining what an autonomous vehicle is and the different levels of autonomy and moving on to the computer vision system behind the decision-making process.

Computer vision system in autonomous vehicles are commonly based on a convolution neural network (CNN) which can adopt many different architectures for image classification and object detection such as ResNet, DenseNet and YOLO. ResNet fixes some of the dense net issues such as the vanishing gradient but still leaves some other issues including poor feature extraction. DenseNet is built on top of ResNet and imporves the feature extraction problem, however, the large densely connected layers have made it hard for vehicle to adopt it due to the high computational costs. Out of the listed models, YOLO (You Only Look Once) is a very fast and accurate object detection algorithm that is being used by manufacturers due to its speed and accuracy. Some papers have looked at potential issues with YOLO and have identified that weather and time of day as being the major drawbacks. To solve these issues, this paper looks at applying a range of data augmentation techniques to simulate adverse weather partially blocking parts of the traffic signs as well as different times of the day.

3 Methodology

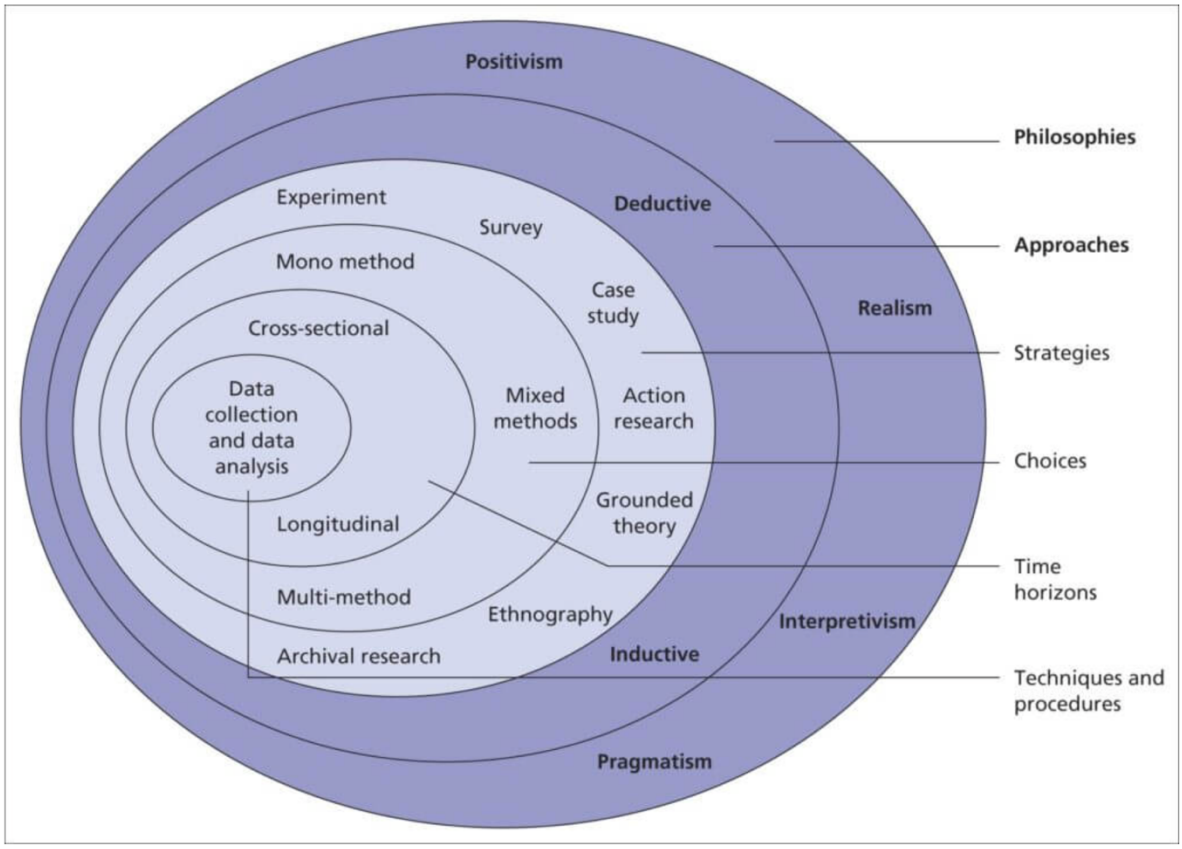


Fig. 14: Research design onion [46]

The philosophy of this research will follow a positivism philosophy, Fig. 14. This is chosen because the research question and implementation rely heavily on quantitative data and measurable outcomes. Since the experiment itself is highly numerical, directly comparing two different YOLO models trained on different datasets, this approach prioritises data findings over the qualitative alternatives. The aim of the study is finding out whether data augmentation to include multiple real life scenarios to a data set, has any impact on the generalisability of the model on identifying and classifying partially blocked traffic signs [47][48]. The aim leads the research into being a very quantitative study with the results coming from a combination of the accuracy percentage of the model, the precision, recall and confusion matrix. This also fits the positivism philosophy as the results could be reproduced and verified by other researchers given they follow the same methodology and use the same data as described in the study. The produced results follow the standard machine learning experiments where bias from researchers is not present and conclusions are made using the produced qualitative data instead of qualitative data coming from user studies [47].

The research will use a deductive approach as the study seeks to tests a clear hypothesis about the impact that data augmentation has on the generalisability of a YOLO model. In

the study's case, the hypothesis is that by adding augmented data, especially data that has been modelled to represent real life scenarios such as weather and different obstructions, will improve the model's ability to recognise partially obstructed traffic signs from images. To test this, one model will be trained using a perfect dataset containing traffic signs from around the world, and a second model will be trained on the same dataset with the addition of augmented images. The algorithm used to augment the images has a 60% chance that an augmentation will be applied meaning that the new augmented data is expected to be about 60% larger than the original. While the original dataset contains about 6000 images of traffic signs the augmented model will have closer to 10000. The model learns the different features of the traffic signs and is able to abstract away from any weather effects, such as rain fog or sun glare, that have been added in the augmented dataset. By comparing the two models trained with different datasets, the study can assess whether the augmented dataset can improve the performance of the model at identifying obstructed traffic signs or if it has little to no effect at all. The results collected will be used to test the effectiveness of each of the models towards the hypothesis [49][50].

The research will follow an experimental approach. The hypothesis of this study, as listed in the previous paragraph, is that applying data augmentation during training will improve the model's ability to generalise to unseen, obstructed traffic signs. In this study, multiple experiments are carried out on the YOLO model to test how it responds under different training conditions [51]. The YOLO model this study will be using is a model that has been trained on a large ImageNet dataset and will have to undergo transfer learning so that it can learn new features from the new datasets. This will mean that the last few layers of the model would be adjusted to fit the dataset from this study. Along with adjusting the last few layers some other parameters could have to change and undergo some fine tuning to make the model more efficient and improve its accuracy [52][53]. The augmented dataset has gone through different parameter shifts and has had multiple experiments applied to the pipeline to get the final version which is used for the model's training. The model's performance will be evaluated on a new unseen dataset that contains partially blocked traffic signs. The performance of the model tested on a clean perfect dataset will be evaluated and compared against the performance of the model trained on the augmented dataset to see whether the experiments carried out have had a positive impact on the model's ability to detect partially obstructed traffic signs.

Since the produced results will all be quantitative, the research will use a multi methods approach that combines multiple quantitative analysis methods. These different quantitative results will come from evaluating the object-detecting YOLO model and producing different metrics such as precision, the recall rate, the F1 score, and the overall accuracy with which the model predicts traffic signs. Using multiple quantitative analysis methods will result in a more robust and trustworthy evaluation of the model because every metric highlights a different feature of the model. This way, when comparing the

two models' performance there are multiple scores that could differ between them. Whilst one model might outperform in one of the metrics it could underperform in the others, this way the research ensures the models performance is thoroughly evaluated. The accuracy percentage evaluates how many of the images the model has accurately predicted. The F1 score looks at the balance between precision and recall. F1 score is the mean between precision and recall and evaluates how good the model is at predicting false positives and false negatives, the higher the score the better the model's performance [54].

The time horizon of this paper takes a cross-sectional approach in order to identify obstructed traffic signs. The approach fits the research design, Fig. 14, because the tests will only happen once when the training of the model has finished, meaning that results are only collected at a single point in time after training [55]. If the models end up with low test scores that require a change to their parameter, those will count as new models therefore new experiments that would require to be tested again. The test data will come from a completely new dataset the model has never seen previously. This new dataset will have partially obstructed traffic signs to test the hypothesis of the study.

3.1 Hardware, Software & Data

The hardware used for this research project was a MacBook Pro equipped with an Apple M2 chip and 32GB of RAM. This system was sufficient for all the processing needed to generate a new augmented dataset and to train and analyse the different YOLO object detection models along with all the software used. The main software used for this project was Visual Studio Code (VS Code). This was used for writing all the data augmentation algorithms, that allowed new images to be generated using an input dataset, and for the transfer learning and fine tuning of the yolo model [52]. The tests required to test the accuracy of the model were also writing in VS Code.

The initial dataset used for training one of the models, as well as producing the augmented dataset, is taken from [56]. The data set comes uses mixed traffic signs from around the world because there wasn't a large enough and easily available traffic sign dataset with only UK traffic signs. To test the model, the project splits the dataset into 80/20. 80% of the dataset was used to train a pretrained YOLO model, 20% was used for validation and to ensure good generalisation rather than overfitting to the training data. This split is commonly used by many machine learning models while training to provide enough data for the models to train on while keeping some unseen data to evaluate the model's performance [57]. Included in the same dataset there is already a testing folder with images ready to be used for evaluating the model's performance after training is complete. The testing dataset contains a good number of unseen images, around 2000, with a similar ratio to the training dataset.

3.2 Implementation

3.2.1 Augmented Dataset

The first thing for this study is to augment the dataset so that there are two different datasets the two models can train on. The project sets up a virtual python environment to keep all the libraries required for the project in one place separated from other global libraries that could be installed on the system.

One of the most important libraries the study makes use of is Albumentations [58]. The library allows for some good image augmentation pipelines and is easy to use after reading the documentation and understanding the parameters of each transform and what they mean in the context of altering the image. The study started off by sampling a few images from the dataset and creating a playground project. This is where a few of the images were loaded from disk and then saved as new images after they had some transforms applied to them with different parameters. This was done to see the change each parameter change would apply to the image and to understand the way the library works in a separate project before committing to creating a full pipeline and altering thousands of images. The only common parameter is probability p. Each transform was applied with probability p, where p = 0 means the transform is never applied and p = 1 means it's always applied.

Fig 15. shows the final augmentation pipeline which includes 5 different augmentation transforms, 2 of which the study has created independently by mixing existing transformations, Fig 16. The library did offer a fog transformation however, after different experimentation with its parameter the fog did not look right hence the implementation of fog_blur. Another transformation the library did not offer was a scenario where images would simulate a nighttime, which was needed to simulate harsher environments for the model to train on.

Fog_blur is made up of Random Brightness Contrast and Gaussian Blur. After experimenting with the parameters, Fig. 16, shows the ideal parameters for the study. In Random Brightness Contrast, the brightness_limit is the range in percentage the brightness is allowed to change, so the image can get between 20% darker and 70% brighter, while the contrast_limit defines the percentage range the contrast can change +-10. The Gaussian Blur has a blur_limit with a kernel size (or matrix) of between 5 and 10, where the larger the kernel the more blur is applied to the picture, Fig. 17.

Night, Fig. 16, is made up of Random Brightness Contrast where the parameters for the brightness are set to negative values. This is to make sure that the image always gets between 20% and 60% darker, as the study always wants a darker image, but it wants the features of the image to still be readable by the model. The contrast is also set to only be negative to make sure the colours are less vibrant. Along with the brightness shift, there

is also a RGB shift. The red and green channels are kept the same while the blue channel is slightly altered to add a blue tint to the image often found during nighttime, Fig. 18.

Random Sun Flare, Fig. 15, is part of the Albumentations experimental library that offers some realistic like scenarios which simulate weather events. The parameters in this case have been left as the default parameters. The flare_roi is the position where the sun flare will appear in the image and by default it is set to appear in the top half of the image. The src_radius is the radius of the sun flare in the image. Src_colour is the colour for the sun flare and by default it is set to be a white light. The angle range is the spread the flares have from the source flare, a [0, 0] range would make the rest of the flares be in a straight light, however, a [0, 1] allows for some random spread. The circle range is how many flares should appear on the image. The selected method is how the flare is applied to the image, overlay being harsher as it's applied on top of the image.

Random Rain, Fig. 15, is also used in the pipeline to simulate rainy weather which is an extremely common scenarios in which vehicles are driving under. The slant range is the random direction from which the raindrops are coming from. The drop length, width and colour set the size and the colour of each rain droplet. The blur value applies a blur effect to the rain droplets, and the brightness coefficient dims the image, so it appears darker.

Zoom Blur, Fig. 15, is the last transform used in the pipeline. Zoom Blur is used to make the traffic signs look like the camera has taken a picture of the traffic sign in motion. The max_factor parameter controls how much zoom should be applied to the image.

```
# augmentations array
augmentations = [
    A.RandomSunFlare(
    flare_roi = [0, 0, 1, 0.5],
    src_radius = 400,
    src_color = [255, 255, 255],
    angle_range = [0, 1],
    num_flare_circles_range = [6, 10],
    method = "overlay",
    p = 1
    ),
    A.RandomRain(
    slant_range=(-45, 45),
    drop_length = 50,
    drop_width = 1,
    drop_color = (200, 200, 200),
    blur_value = 10,
    brightness_coefficient = 0.7,
    rain_type = "default",
    p = 1
```

```
    ),
    A.ZoomBlur(
    max_factor = (1.1, 1.4),
    p = 1
    ),
    night,
    fog_blur
]
```

Fig. 15: Augmentation pipeline

```
night = A.Compose([
     A.RandomBrightnessContrast(
        brightness_limit=(-0.6, -0.2),
        contrast_limit=(-0.2, 0),
        p=1
    ),
    A.RGBShift(
        r_shift_limit=0,
        g_shift_limit=0,
        b_shift_limit=(20, 60),
        p=1
    ),
])

fog_blur = A.Compose([
    A.GaussianBlur(blur_limit=(5, 10), p=1),
    A.RandomBrightnessContrast(
        brightness_limit=(-0.2, 0.7),
        contrast_limit=(-0.1, 0.1),
        p=1
    )
])
```

Fig. 16: Created augmentation transforms

Fig. 17: gaussian blur + random brightness compared to sun_flare

Fig. 18: Night applied to an image, original (left) compared to augmented (right)

3.2.2 YOLO Model

To implement the YOLO models the study relies heavily on the Ultralytics library [59]. In a test project the study set up different YOLO model with different depths of the model in order to test the time it would take for a model to train and the accuracy a model would achieve. This was done to decide on a final model size that offered quick training and good accuracy. The model the study has settled on is a small YOLO model after testing a nano, small and a medium model. The nano was the quickest to train, taking around half an hour, the small one taking between 1 to 2 hours depending on the size of the dataset and the medium model taking over 5 hours on the non-augmented dataset of 6000 images. The medium model was not tested on the larger 10000 image dataset as it would have taken almost double the time and the accuracy provided between the small and medium model was looking very similar on both converging towards 1 after around 10 epochs.

Fig. 19, shows the code used to train the augmented model. The study makes use of transfer learning, which uses an already trained model and fine tuning it's parameters to account for a new dataset. The first line loads a small pre-trained YOLOv8 model that has been trained on a ImageNet dataset with a size of over 14 million images [60] [61]. The

model then takes what it has learned from the previous training and applies it to the new dataset being fed into the training step. This allows it to train faster, and it as it only has to change a few layers to account for the number of new classes the new dataset has. The model has trained for 30 epochs, however after about 10 to 15 epochs the accuracy line on the validation dataset has converged and it showed to be around 0.99. The imagesz field is the size of the image that the model is being trained on. Based on the given size of 128, the images are rescaled appropriately making them closer to the target size and then padding is added if needed to make the image a square. The size of 128 was chosen as the dataset has a varied number of image sizes and most of them being small images, leaving it to the default 640 pixel would make some of the images too pixelated for the model to learn any valuable feature from them. The augment argument had to be turned off in order to test the study's hypothesis. This is something that is turned on by default for the model to improve its accuracy and have a more varied set of images. However, since the hypothesis involved comparing a model trained on a non-augmented dataset against one that's trained on an augmented dataset, having the library interfere with what augmentation it would add to the data would invalidate the experiment.

```
model = YOLO("yolov8s-cls.pt")

model.train(
    data="data/augmented_dataset/data",
    epochs=30,
    imgsz=128,
    augment=False,
)
```

Fig. 19: Augmented model code.

To evaluate the YOLO models the study makes use of another library, scikit-learn [62], to analyse and come up with different measurable outcomes to compare the models. The four evaluation metrics used are accuracy_score, precision_score, recall_score, f1_score. They take in two arguments, a y_true and a y_pred, which are the true value of the model and the predicted value by the model, respectively. To calculate the y_true and y_pred so it could be used by the library, the study created a function calculate y_values, Fig. 20. The function takes in a YOLO model, the model it will calculate the values for, and a Boolean flag, whether to use the augmented test dataset or the non-augmented dataset. Inside the function, there is a loop over the generated results to find the true class and the predicted class for a given classification. To find the true class, there is a helper function, get_true_class, that loops over a dictionary, names_map, which contains all the classes the model has learned and then compares the class within the dictionary with the filename in the test dataset to see whether they match. If no class matches, then it returns None.

```python
def calculate_y_values(model: YOLO, use_aug: bool):
    # Select test dataset
    test_folder = ""

    if use_aug:
        test_folder = "data/test_dataset_aug"
    else:
        test_folder = "data/test_dataset"


    names_map = model.names
    y_true = []
    y_pred = []

    results = model(test_folder, stream=True, verbose=False)

    for r in results:
        pred_class = r.probs.top1

        filename = os.path.basename(r.path)
        true_class = get_true_class(filename, names_map)

        if true_class != None:
            y_true.append(true_class)
            y_pred.append(pred_class)

    return y_true, y_pred
```

Fig. 20: Calculate y values function to be used by scikit-learn library

4 Results & Discussion

To evaluate the two models thoroughly different metrics were used, at different points in the training. The metrics used are a confusion matrix taken on the validation dataset during training, accuracy metrics taken on the unseen dataset as well as, precision, recall and an f1 score evaluated during testing on the unseen dataset. These wide range of different metrics are used to check whether the hypothesis of the study was correct, and whether a model trained on an augmented dataset will perform better at detecting partially blocked traffic signs, common in real life scenarios.

4.1 Results

The model that's been trained on the original dataset with no augmentations, will be called the non-augmented model, whilst the model that's been trained on the augmented dataset will be called the augmented model. The non-augmented model has achieved an accuracy of **86.26%** when being tested on the unseen original, non-augmented test dataset, as seen from Fig. 21. The same model has achieved a significantly lower accuracy of only **65.94%** when being tested on the augmented test dataset, which includes partially blocked signs with realistic weather scenarios. The augmented model has managed to achieve higher accuracies on both datasets with **91.78%** accuracy on the original test dataset and **83.63%** accuracy on the augmented test dataset. This shows a great improvement overall, but most noticeably, almost **20% improvement** on detecting partially obstructed traffic signs.

```
Non-augmented model accuracy on original dataset:  0.8626
Non-augmented model accuracy on augmented dataset: 0.6594
Augmented model accuracy on original dataset: 0.9178
Augmented model accuracy on augmented dataset: 0.8363
```

Fig. 21: A comparison between the model's accuracies

A similar trend to the accuracy is present in the precision results, Fig. 22. Precision is how many of the model's predictions were correct out of all the positive predictions, for one class, and the average is then taken across all classes. The non-augmented model achieved an **80.77%** precision on the non-augmented test dataset and a lower **62.85%** precision on the augmented test dataset. The augmented model performed better at on both datasets, achieved a higher, **87.27%** precision on the original test dataset and a **74.17%** precision on the augmented test dataset. The precision score for both decreases when tested on the augmented test dataset, however while the non-augmented model decreased around **18%** the augmented model only decreased around **13%**.

```
Non-augmented model precision on original dataset:  0.8077
Non-augmented model precision on augmented dataset: 0.6285
Augmented model precision on original dataset: 0.8727
Augmented model precision on augmented dataset: 0.7417
```

Fig. 22: A comparison between the model's precision

The same trend of the augmented model outperforming the non-augmented model continues in the recall results, Fig. 23. Recall can be seen as a per class accuracy, of the total traffic signs in one class, how many did the model correctly identify. In this experiment, recall is first calculated for each class and then the average is taken across all classes to calculate the result. The non augmented model achieved a score of **83.7%** recall on the original test dataset and a score of **62.64%** on the augmented tests dataset. The augmented model has achieved higher scores for both values, with an **87.69%** recall

on the original test dataset and **77.83%** on the augmented test dataset. While the non-augmented model has a **20%** loss when comparing the results of the same model on the two separate datasets, the augmented model halves the loss to only **10%**.

```
/Users/tudorhaba/University/7. Final Project/object-detection/venv/lib/python3.13/site-packages/sklearn/metrics/_classification.p
y:1833: UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division` pa
rameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
Non-augmented model recall on original dataset:  0.8370
Non-augmented model recall on augmented dataset: 0.6264
Augmented model recall on original dataset: 0.8769
Augmented model recall on augmented dataset: 0.7783
```

Fig. 23: A comparison between the model's recall

The augmented model outperforming the non-augmented model trend continues when looking at the f1 score values produced by evaluating both models, Fig. 24. The f1 score means to balance both the precision and recall. A high f1score meaning the model is good at correctly identifying the traffic sign without making too many mistakes. The non augmented model achieved a f1 score of **78.73%** on the original test dataset, while achieving a much lower score of **57.51%** on the augmented dataset. The f1 score for the augmented model achieved a higher **84.55%** score on the original test dataset and a **72.96%** on the augmented test dataset. The difference between the results for the first model is around **20%** and the difference between the augmented model being much lower at around **10%** whilst still being above a good score of **70%**.

```
Non-augmented model f1 score on original dataset:  0.7873
Non-augmented model f1 score on augmented dataset: 0.5751
Augmented model f1 score on original dataset: 0.8455
Augmented model f1 score on augmented dataset: 0.7296
```

Fig. 24: A comparison between the model's f1 score

Fig. 25 shows the normalized confusion matrices for the non-augmented model (left) and the augmented model (right), this can be seen in more details in the appendix. The matrices were generated using the Ultralytics library and evaluated on the validation set obtained from an automatic dataset split, 80% of the data being used for training and 20% for validation. Both models show a strong performance with most of the prediction being on the diagonal line, indicating a high rate of classification (closest to 1). However, looking at the matrices, the non-augmented model has fewer off diagonal predictions compared to the augmented model, indicating a better performance on the validation set.

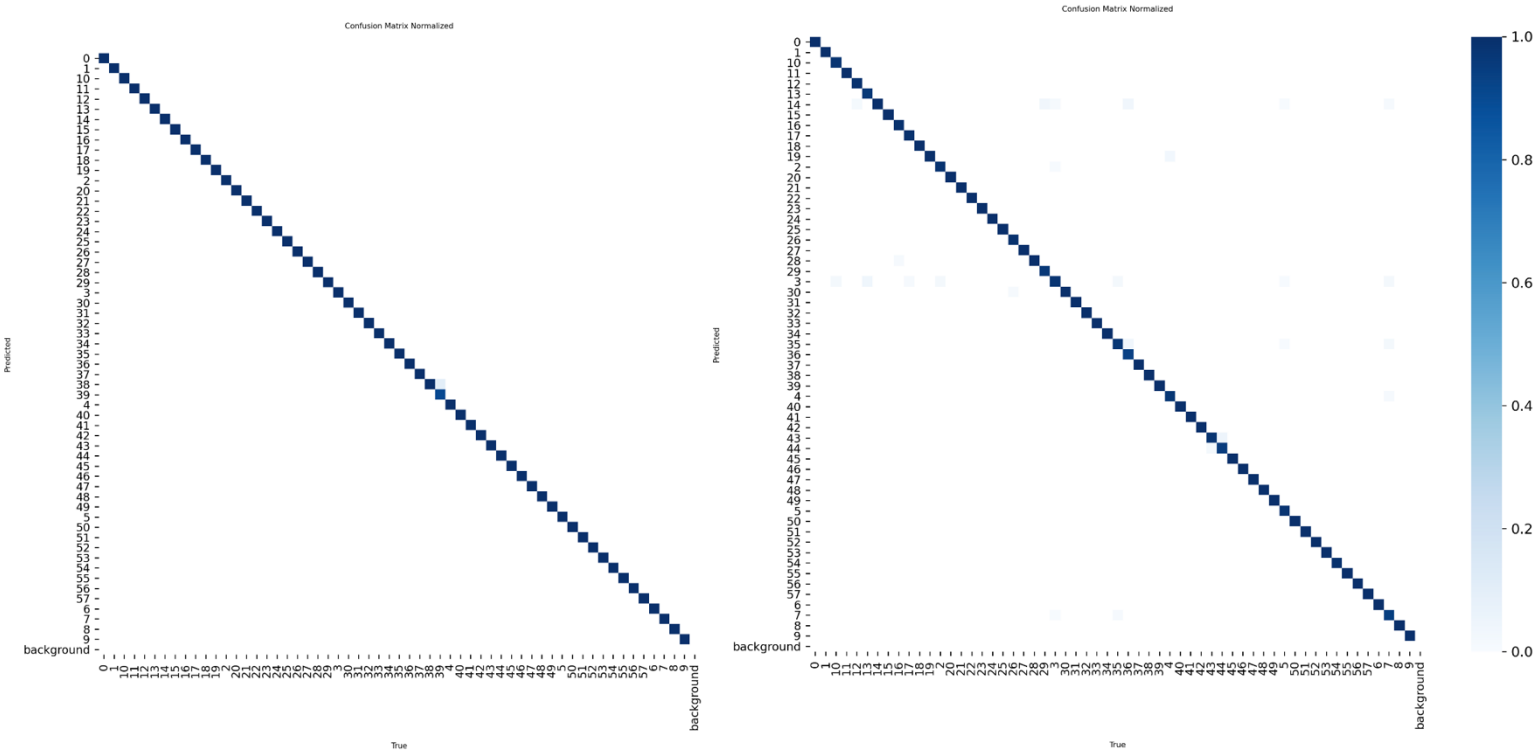


Fig. 25 Normalized confusion matrix of the non-augmented model (left) and augmented model (right) – see appendix for labels and larger images

4.2 Discussion

When evaluating the accuracy, the non-augmented model performed well on the original dataset with it being 86.26% accurate, meaning that 9 out of 10 times the model predicts the correct traffic sign in perfect weather conditions. However, the massive 20% difference achieved on the augmented dataset, has real life implications especially in autonomous vehicles. Only being able to identify 6 out of 10 traffic signs during adverse weather or during nigh time would be very dangerous for everybody on the road. Adverse weather is a common occurrence, and many people drive at night so any partial obstructions such as rain, rain, shadows or low light affecting the model's prediction capabilities affects the reliability of the model. A poorly reliable model increases the risk to accidents. The non augmented model has learned some features from the training dataset, however, by not having a bigger and more diverse dataset the model only focused on clear features of images and fails to identify slight variations. The augmented model outperformed the non-augmented model, with 91% accuracy in perfect weather condition, and 83% accuracy for the augmented dataset. Being able to identify 8 out of 10 traffic signs during poor driving conditions is a massive difference compared to the non-augmented model. It is a more reliable model, and it suggests that augmentation and training on a bigger and more varied dataset have helped the model generalise to

more difficult scenarios that are similar to real life scenarios. The non-augmented model's performance aligns with the gap described in [45], which highlights a performance drop in traffic signs detection systems when exposed to real-world conditions and challenges such as low light. The augmented model's performance also dropped when tested against the augmented dataset, however, the drop is significantly smaller and has kept the accuracy to a similar level the original model was when evaluated on perfect data. This shows augmenting the dataset has improved the robustness of the model in real-world scenarios. While the 83% accuracy is a good result for the augmented model and it is a great improvement compared to the non-augmented model, it is still lower than the 91% it achieved on the original dataset. The 8% difference is significant for a critical system and shows that work is still needed to improve the accuracy in real life scenarios. In countries like the United Kingdom where rain is common the model is more likely to perform around the 80% accuracy range identified during testing. Further work on the augmented dataset, such as more difficult augmentations or a larger dataset, could improve the model's accuracy and use in such countries.

The results obtained from the precision test show that the augmented model has produced more reliable results compared to the non-augmented model. The model has achieved a higher percentage in during both tests which means the model produces less false positives, especially when evaluated against the more challenging dataset model with a 12% increase in its precision. Since precision means how many times a model has correctly identified the class of an image correctly, a decrease in the result means the model is falsely classifying more images. The data shows a decrease in both model's precision levels when given more challenging data from the augmented dataset, however, there is less of a decrease from the augmented model than on the non-augmented model. This smaller decrease in performance indicates the augmented model is more robust and has relied on less weak features that could be easily altered by introducing weather like augmentations to a scenario. This increase in the precision of the augmented model suggest that the model is able to generalise more effective to unseen data. With this, and the model's ability to learn and transfer more complex features from images, the model is better suited for use in real life scenarios. However, while the augmented model's precision is better compared to the non-augmented model, its score is nowhere near perfect and shows there is room for improvement. To ensure a safer use in real world scenarios the model could benefit from additional training data, or additional more complex augmentations being applied to the dataset.

The recall results demonstrate a consistent pattern between the augmented and non-augmented model. While both models perform well being tested on the original non-augmented test data set, showing there is potential that both models have a high chance of performing well under perfect conditions, the augmented model performs a lot better on the augmented dataset. This shows that data augmentation helps to improve how

robust a model is and indicates the model is missing less traffic signs, aligning with the hypothesis. The recall formula is

$$\boldsymbol{Recall} = \frac{\boldsymbol{True\ Positives}}{\boldsymbol{True\ Positives} + \boldsymbol{False\ Negatives}}$$

in other words, out of all the traffic signs belong to a class, how many were correctly identified and how many were missed. This is especially import for traffic signs since means a sign completely has more of an impact that miss interpreting it. If the car misinterprets a sign, it knows there was a sign there and however, if it misses it completely the car will have no idea there was supposed to be a sign at all and could result in worse consequences. The 20% drop in the recall score for the non-augmented model means the model is very sensitive to variations in the data. This also means that the model has learned features from the training data and does not generalise well to unseen variations for the same data. Compared to the non-augmented model, the augmented model has halved the difference it loses between the original and augmented test dataset. This suggests the model was able to make assumptions based on the features it learned during training and better generalise to unseen data. This also means that the model misses substantially less traffic signs, a 15% difference between the augmented and non-augmented model, results in a model that's better suited for experiments in less than perfect real-world scenarios. However, the score achieved for both models is an average taken between all the classes of the dataset. This is a problem as the average has been affected because of the dataset. As seen from the warning in, Fig. 23, some of the classes had no true positives, they were completely missing from the test dataset, better seen in the report from the appendix, and as a result the calculation was set to 0. This would be a problem if the recall was calculated differently for the model, however, because the study uses the same datasets for calculating the recall for the model, this has equally affected both model's recall score. While, this is not perfect, it shows there is class imbalance in the test datasets, and it shows there are missing classes in the test datasets.

The F1 score is a balance between precision and recall scores achieved by the models. A high F1 score would suggest that the models are good at both reliably predicting the traffic signs and avoiding too many false positives. While a high f1 score is visible for both models on the non-augmented test dataset, the non-augmented model is drastically affected when tested against the augmented test dataset. This is the first time a model has performed rather poorly compared to the other estimates, with a score below 60%. This score suggest that the model's macro average was affected by the unbalanced dataset and the model is not very reliably classifying images (precision) or the model is missing some of the traffic signs completely (recall). Looking at the augmented model, the drop between the model's performance on the original test dataset and augmented dataset is smaller when compared to the other model, 10% drop, compared to 20% drop.

This suggest that the model is more robust due to being trained on the augmented dataset and is a safer model to real world applications. Even though the drop is halved, the drop is still affecting the model's performance, only achieving just above 70%. This result is good for a vehicle under harsh weather conditions but there is still room for improving the score to make it more reliable and robust in those scenarios. What has probably affected the score the most is the lack of some classes in the test data, as seen with the recall score and warning. The score would see an increase if the test dataset would be larger and include all the classes the model's been trained on.

The normalized confusion matrix from Fig. 25, shows that both models achieve a high rate of classification, noted by proportion of classifications along the diagonal line. This suggest that during validation both models were able to extract useful features out of the images. However, when comparing the two matrices there is a very clear difference between the off diagonal results. While the non-augmented model has a few off diagonal classifications, the augmented model has more, meaning the augmented model misclassifies a larger number of traffic sings. This indicates a stronger performance for the non-augmented model on the validation set. The validation dataset comes from a 80/20 split within the training dataset. It has to be noted that the confusion matrices come from different datasets, since the split comes directly from the training dataset, thus, the non-augmented model is validated against non-augmented images while the augmented model is validated against augmented images. Looking at the confusion matrix for the non-augmented model, there are very few misclassifications. Since the 20% validation data comes from the same dataset as the training, this could indicate a very strong potential of model overfitting. While the non-augmented model is performing very well almost close to 100% accuracy on the validation dataset and then drops to 86% on a non-augmented test dataset it could indicate the model has overfitted and learned more than features from the training dataset. In comparison, the augmented model shows more misclassifications on the confusion matrix, as seen by the increased number of off diagonal classifications. This indicates the model being less tightly fitted to the training dataset and is also backed up by the other results it achieved against the augmented test dataset. These results show that a strong performance during validation does not translate to a strong performance during testing.

Based on the accuracy, precision, recall and F1 score, where the augmented model has outperformed the non-augmented model, the augmented model is the choice to take forward in a critical system such as autonomous vehicles, see appendix for a detailed report table comparing the augmented model's performance on the unseen augmented test dataset. This indicates that the augmented model is more accurate and more robust in identifying obstructed traffic signs in challenging conditions. This demonstrated that data augmentation has made an impact and has helped the model generalise better on unseen data proving the hypothesis.

Overall, based on the results found during evaluation, the research question was answered, and the results have shown that without any data augmentation applied to the images, the model will struggle to accuracy recognise traffic signs. The results also supported the hypothesis of the study, showing that applying data augmentation during training has improved the model's generalisation on unseen, obstructed traffic data. With the hypothesis proven, this also proves the initial research question, and it demonstrates that a model can maintain accurate sign recognition when the signs are partially obstructed by weather events, but the models need help during training to understand what features to extract in such scenarios. While some performance decline can be observed in the augmented model's performance on the more challenging images, it has still outperformed the non-augmented model. This shows there is potential for future work to expand on the augmentation methods used to improve the model's robustness.

5 Limitations & Future Work

5.1 Limitations

This study was affected by several problems, with the biggest problem caused by the dataset. Trying to find a good-sized dataset with many examples of traffic signs that was good for training a YOLO model to detect objects proved challenging and resulted into the study scaling back and working with YOLO for classification instead. Even for classification it took a long time to try and find a decent sized dataset. This is where the second problem came into effect, the size of the dataset. The dataset found is of relatively small to train a model on. When splitting a 6000-image dataset 80/20 between training and validation the dataset becomes even smaller and leaves just 4800 images for training. As a direct result of this the non-augmented model suffered as seen from the results always being lower than the augmented model. This was the case even when classifying the non-augmented images during testing where that is exactly the type of data the model has trained on. Data imbalance within the dataset is the other problem caused by the found dataset. Whilst the dataset offered a lot of classes which is good for the model's training, it lacked a good balance between all the classes. Some of the classes would have hundreds of images whilst other will only contain a few images, with some classes contain as few as 5. This has definitely had an impact on the model's ability to detect some of the classes as there just wasn't enough data for the model to train on. This problem was carried over into testing the model, the testing dataset was lacking some of the classes all together as seen from the warning in coming on during the recall evaluation and seen in the appendix in the detailed classification report, where multiple classes have 0 images. These limitations would suggest that future research could benefit from a larger, more balanced dataset to improve the model performance and generalisability.

5.2 Future Work

Looking at the results and the limitations working on the dataset would be a good starting step to improve the models. This could mean looking for a different dataset that keeps a similar number of classes and increases the overall number of training images or improving the current dataset by merging or manually adding similar images from other datasets. This would also result in a larger and more varied dataset that could be improving the overall performance of the models.

Identifying a different dataset that would be better suited for object detection rather than object classification would also be something worth considering in the future. As this experiment has proved, augmenting the data helps improving the model's generalisability on unseen images, therefore the same would be expected while doing the experiment on object detection. Doing object detection in the future would prove whether the same hypothesis can be applied there and it would be more suited for autonomous vehicles. The object classification experiment proves that the model generalises well on unseen traffic signs only, but more research is needed to see whether an autonomous vehicle is also able to generalise on unseen data when there are more objects in an image.

Something that could be considered for the future would be a bigger range of experiments with the augmentations. Due to time constraints, a limited number of combinations and experiments were tried but there are many other weather events that could potentially be simulated by augmenting data. Some of the augmentations to be considered could include, snow buildup or snowing, icy conditions and heatwave distortion. While rain was already part of the augmented dataset, further experiments could be tried with different variations of rain, drizzle and torrential. A combination of more than one augmentation, such as rain and night, on the same image would also push the model's limits and would create even more challenging data to test the model on.

6 Conclusion

This project taught me a lot of valuable information about the world of autonomous vehicles and how image detection in a computer vision system works. I was able to apply my existing CNN knowledge and expand it further into more specialised deep learning models like YOLO. While I ended up choosing YOLOv8 for the final model, the prior research into all YOLO versions and similar deep learning models such as ResNet and DenseNet, has strengthen my understanding of modern object detection architectures. One aspect that went really well and exceeded my initial expectations were the results I archived training the models. Initially I was expecting the models to achieve around 70% accuracy, however, both models were able to achieve a much higher accuracy and even have achieved a higher accuracy on the augmented test dataset. I am happy to have

learned how to thoroughly evaluate a model by learning about different metrics such as precision, recall and the f1 score. Although I would have wanted to put the models under more tests, such as a T-test, I was able to thoroughly assess the models under the given circumstances and timeframe. While the timeframe was tight and it posed a challenge, I believe it taught me how to manage more than one complex projects at the same time, balancing this project and work projects.

However, there were some challenges that I wasn't expecting and I wasn't prepared for. I was not expecting finding a good dataset would prove to be this difficult. I spend longer than expected looking for a good dataset made for object detection, but I ended up having to scale back the project slightly and use YOLO for classification instead. While the outcome is similar and I don't think the results would be too different, it would have been an interesting experiment to see if other objects would have had an impact on the model. Class imbalance was the other challenge introduced by the dataset, it made it so the model would have less experience with some of the classes and made it more prone to errors.

Overall, I think this has been a successful project that has strengthened my technical knowledge in machine learning and improved my research skills.

7 Appendix

Class Labels

ClassId	Name
0	Speed limit (5km/h)
1	Speed limit (15km/h)
2	Speed limit (30km/h)
3	Speed limit (40km/h)
4	Speed limit (50km/h)
5	Speed limit (60km/h)
6	Speed limit (70km/h)
7	speed limit (80km/h)
8	Dont Go straight or left
9	Dont Go straight or Right
10	Dont Go straight
11	Dont Go Left
12	Dont Go Left or Right
13	Dont Go Right
14	Dont overtake from Left
15	No Uturn
16	No Car
17	No horn
18	Speed limit (40km/h)
19	Speed limit (50km/h)
20	Go straight or right
21	Go straight
22	Go Left
23	Go Left or right
24	Go Right
25	keep Left
26	keep Right
27	Roundabout mandatory
28	watch out for cars
29	Horn
30	Bicycles crossing
31	Uturn
32	Road Divider
33	Traffic signals
34	Danger Ahead
35	Zebra Crossing
36	Bicycles crossing
37	Children crossing
38	Dangerous curve to the left
39	Dangerous curve to the right
40	Unknown1
41	Unknown2

42	Unknown3
43	Go right or straight
44	Go left or straight
45	Unknown4
46	ZigZag Curve
47	Train Crossing
48	Under Construction
49	Unknown5
50	Fences
51	Heavy Vehicle Accidents
52	Unknown6
53	Give Way
54	No stopping
55	No entry
56	Unknown7
57	Unknown8

Classification Report

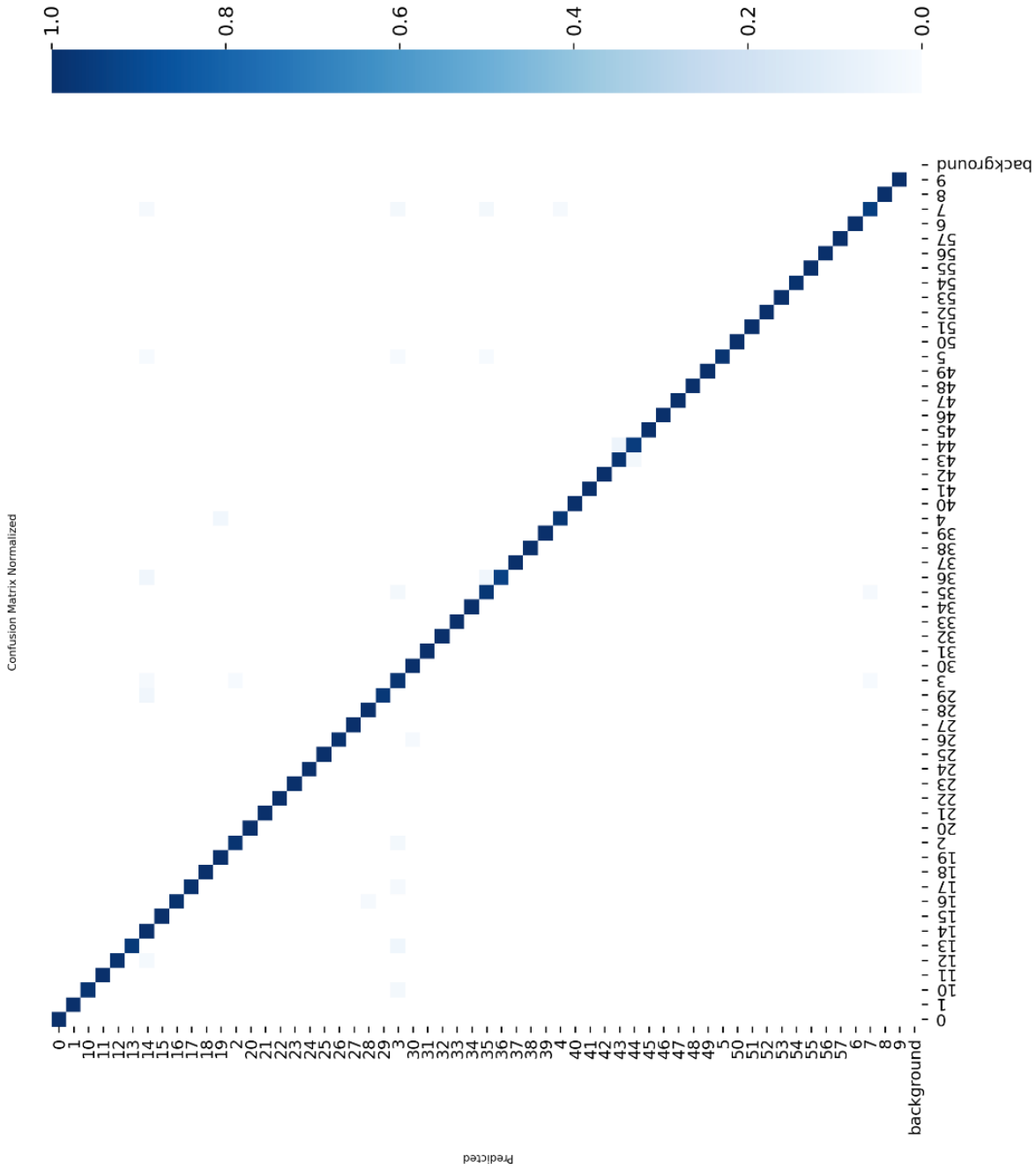
Class	Precision	Recall	F1-Score	Support
0	0.750	0.923	0.828	13
1	0.769	0.833	0.800	12
2	0.968	1.000	0.984	60
3	0.975	0.891	0.931	129
4	0.656	0.955	0.778	22
5	0.948	0.802	0.869	91
6	0.706	1.000	0.828	12
7	0.926	0.694	0.794	36
8	0.947	0.947	0.947	76
9	1.000	0.929	0.963	84
10	0.000	0.000	0.000	0

11	0.000	0.000	0.000	0
12	0.725	0.877	0.794	57
13	0.500	1.000	0.667	2
14	0.900	0.750	0.818	12
15	0.375	0.750	0.500	8
16	0.857	0.600	0.706	10
17	0.737	0.538	0.622	26
18	0.333	0.500	0.400	2
19	0.974	0.841	0.902	132
20	0.885	0.958	0.920	24
21	0.970	0.941	0.955	68
22	0.806	0.962	0.877	26
23	0.852	0.893	0.872	84
24	0.744	0.941	0.831	34
25	0.621	1.000	0.766	18
26	0.250	1.000	0.400	2
28	1.000	0.625	0.769	8
29	1.000	0.957	0.978	46
30	1.000	0.917	0.957	12
31	0.958	0.885	0.920	26

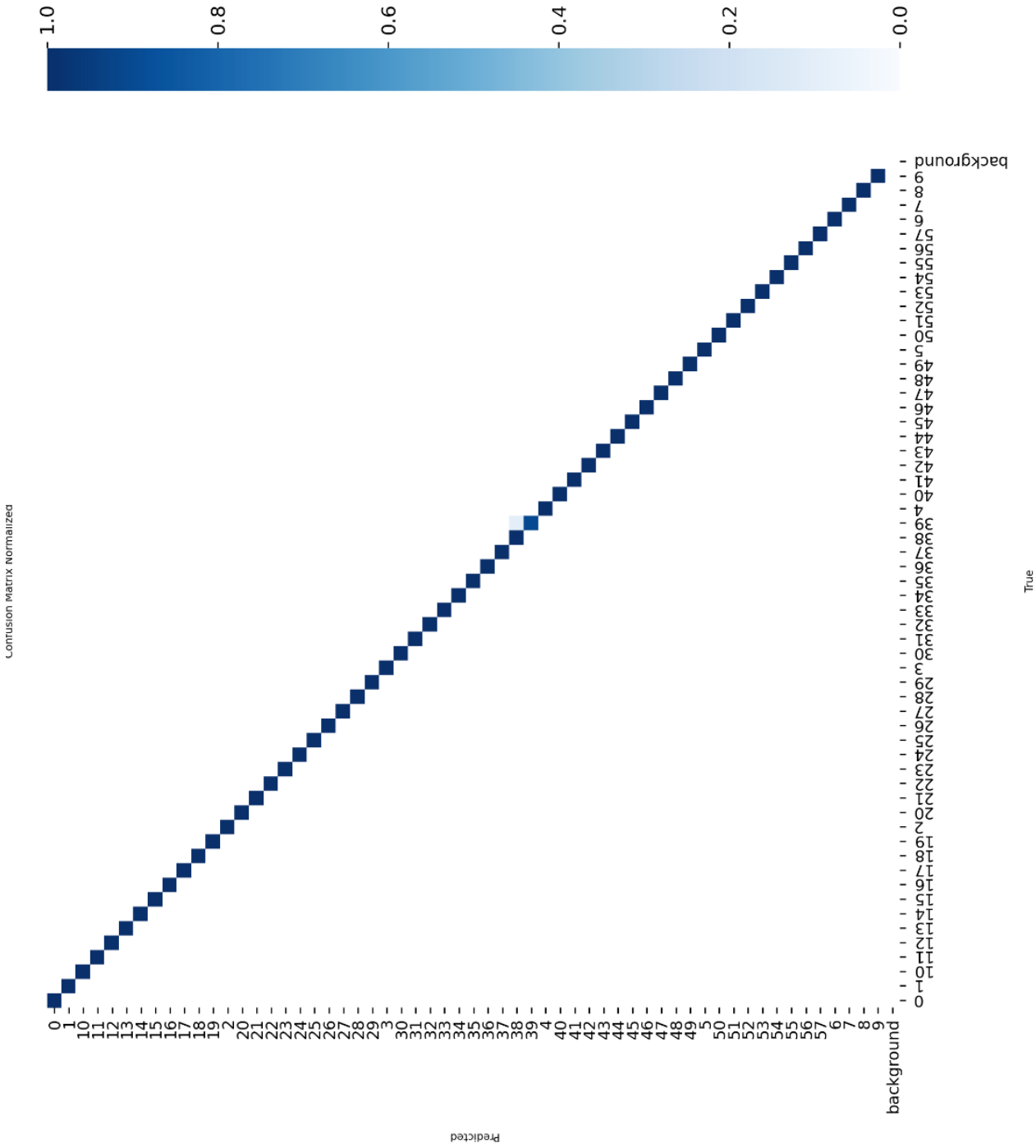
32	0.667	0.050	0.093	40
33	0.373	0.633	0.469	30
34	0.808	0.724	0.764	58
35	0.533	1.000	0.696	8
36	1.000	0.375	0.545	8
37	0.938	0.833	0.882	18
38	0.793	0.629	0.702	116
39	0.138	0.167	0.151	24
40	0.105	1.000	0.190	2
41	0.333	0.857	0.480	14
42	0.875	0.700	0.778	10
43	1.000	0.833	0.909	6
44	0.950	0.905	0.927	42
45	0.865	0.900	0.882	50
46	0.778	0.700	0.737	20
47	0.800	1.000	0.889	4
48	0.903	0.933	0.918	30
49	1.000	1.000	1.000	2
50	0.966	0.966	0.966	175
51	0.964	0.931	0.947	58

52	0.854	0.875	0.864	40
53	0.667	1.000	0.800	4
54	1.000	0.767	0.868	30
55	0.911	0.820	0.863	50
56	0.923	0.857	0.889	14
57	0.000	0.000	0.000	0
---	---	---	---	---
Accuracy			**0.836**	**1985**
Macro Avg	0.742	0.778	0.730	1985
Weighted Avg	0.873	0.836	0.842	1985

Normalised confusion matrix of the augmented model.



Normalised confusion matrix of the non-augmented model.



8 Bibliography

[1] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, 'Convolutional neural networks: an overview and application in radiology', *Insights Imaging*, vol. 9, no. 4, pp. 611–629, Jun. 2018, doi: 10.1007/s13244-018-0639-9.

[2] M. Q. Kheder and A. A. Mohammed, 'Improved traffic sign recognition system (itsrs) for autonomous vehicle based on deep convolutional neural network', *Multimed. Tools Appl.*, vol. 83, no. 22, pp. 61821–61841, Jul. 2024, doi: 10.1007/s11042-023-15898-6.

[3] C. Isidore, 'Elon Musk says his next product will be the most important Tesla has ever produced. And he says it's (finally) almost here | CNN Business', CNN. Accessed: Jun. 25, 2025. [Online]. Available: https://www.cnn.com/2025/06/12/business/musk-tesla-robotaxi-rollout-austin

[4] 'Uber brings forward trialling driverless taxis in UK', BBC News. Accessed: Jun. 25, 2025. [Online]. Available: https://www.bbc.com/news/articles/cwynm4wl1j7o

[5] A. Dineley, F. Natalia, and S. Sudirman, 'Data Augmentation for Occlusion-Robust Traffic Sign Recognition Using Deep Learning', 2024, *ICIC International 学会*: 04. doi: 10.24507/icicelb.15.04.381.

[6] D. Parekh *et al.*, 'A Review on Autonomous Vehicles: Progress, Methods and Challenges', *Electronics*, vol. 11, no. 14, p. 2162, Jan. 2022, doi: 10.3390/electronics11142162.

[7] X. Dong and M. L. Cappuccio, 'Applications of Computer Vision in Autonomous Vehicles: Methods, Challenges and Future Directions'.

[8] 'SAE Levels of Driving Automation™ Refined for Clarity and International Audience - SAE International'. Accessed: Nov. 13, 2025. [Online]. Available: https://www.sae.org/news/blog/sae-levels-driving-automation-clarity-refinements

[9] 'The SAE Levels of Driving Automation Fully Explained (Levels 0–5)', Zego. Accessed: Jan. 23, 2026. [Online]. Available: https://www.zego.com/blog/the-sae-levels-of-driving-automation-explained-levels-0-5/undefined/the-sae-levels-of-driving-automation-explained-levels-0-5

[10] N. Manakitsa, G. S. Maraslidis, L. Moysis, and G. F. Fragulis, 'A Review of Machine Learning and Deep Learning for Object Detection, Semantic Segmentation, and Human Action Recognition in Machine and Robotic Vision', *Technologies*, vol. 12, no. 2, p. 15, 2024, doi: 10.3390/technologies12020015.

[11] D. U. Iqbal, 'AUTONOMOUS VEHICLES: CHALLENGES AND OPPORTUNITIES IN COMPUTER VISION AND AI', *Comput. Sci. Bull.*, vol. 4, no. 02, pp. 146–161, Dec. 2021.

[12] B. Bala and S. Behal, 'A Brief Survey of Data Preprocessing in Machine Learning and Deep Learning Techniques', in *2024 8th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, Oct. 2024, pp. 1755–1762. doi: 10.1109/I-SMAC61858.2024.10714767.

[13] 'What Is Data Labeling? | IBM'. Accessed: Jan. 21, 2026. [Online]. Available: https://www.ibm.com/think/topics/data-labeling

[14] Y. Wu and J. Feng, 'Development and Application of Artificial Neural Network', *Wirel. Pers. Commun.*, vol. 102, no. 2, pp. 1645–1656, Sep. 2018, doi: 10.1007/s11277-017-5224-x.

[15] 'What is a Neural Network? | IBM'. Accessed: Jun. 14, 2025. [Online]. Available: https://www.ibm.com/think/topics/neural-networks

[16] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, 'A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects', *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 6999–7019, Dec. 2022, doi: 10.1109/TNNLS.2021.3084827.

[17] A. Ajit, K. Acharya, and A. Samanta, 'A Review of Convolutional Neural Networks', in *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, Feb. 2020, pp. 1–5. doi: 10.1109/ic-ETITE47903.2020.049.

[18] A. Zafar *et al.*, 'A Comparison of Pooling Methods for Convolutional Neural Networks', *Appl. Sci.*, vol. 12, no. 17, Art. no. 17, Jan. 2022, doi: 10.3390/app12178643.

[19] X. Ying, 'An Overview of Overfitting and its Solutions', *J. Phys. Conf. Ser.*, vol. 1168, no. 2, p. 022022, Feb. 2019, doi: 10.1088/1742-6596/1168/2/022022.

[20] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, 'A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS', *Mach. Learn. Knowl. Extr.*, vol. 5, no. 4, pp. 1680–1716, Dec. 2023, doi: 10.3390/make5040083.

[21] A. Landge, 'A Review on Object Detection for Autonomous Vehicles', *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 13, no. 1, pp. 348–351, Jan. 2025, doi: 10.22214/ijraset.2025.66191.

[22] T. Diwan, G. Anirudh, and J. V. Tembhurne, 'Object detection using YOLO: challenges, architectural successors, datasets and applications', *Multimed. Tools Appl.*, vol. 82, no. 6, pp. 9243–9275, Mar. 2023, doi: 10.1007/s11042-022-13644-y.

[23] S. Pogrebivsky, 'Precision, Recall and F1 Explained (In Plain English)', DataGroomr, LLC. Accessed: Oct. 18, 2025. [Online]. Available: https://datagroomr.com/precision-recall-and-f1-explained-in-plain-english/

[24] 'Figure 4: Visualization of the AP metric, estimated by the area below...', ResearchGate. Accessed: Oct. 18, 2025. [Online]. Available: https://www.researchgate.net/figure/sualization-of-the-AP-metric-estimated-by-the-area-below-the-Precision-Recall-curve-at_fig2_367251324

[25] M. Hussain, 'YOLOv1 to v8: Unveiling Each Variant–A Comprehensive Review of YOLO', *IEEE Access*, vol. 12, pp. 42816–42833, 2024, doi: 10.1109/ACCESS.2024.3378568.

[26] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, 'A Review of Yolo Algorithm Developments', *Procedia Comput. Sci.*, vol. 199, pp. 1066–1073, 2022, doi: 10.1016/j.procs.2022.01.135.

[27] 'Understanding Intersection over Union for Model Accuracy', viso.ai. Accessed: Jan. 18, 2026. [Online]. Available: https://viso.ai/computer-vision/intersection-over-union-iou/

[28] DeepBean, *How YOLO Object Detection Works*, (Mar. 12, 2023). Accessed: Jan. 19, 2026. [Online Video]. Available: https://www.youtube.com/watch?v=svn9-xV7wjk

[29] 'Leaky ReLU Activation Function Explained | Ultralytics'. Accessed: Jan. 19, 2026. [Online]. Available: https://www.ultralytics.com/glossary/leaky-relu

[30] J. Redmon and A. Farhadi, 'YOLOv3: An Incremental Improvement'.

[31] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, 'YOLOv4: Optimal Speed and Accuracy of Object Detection', Apr. 23, 2020, *arXiv*: arXiv:2004.10934. doi: 10.48550/arXiv.2004.10934.

[32] C. Li *et al.*, 'YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications', Sep. 07, 2022, *arXiv*: arXiv:2209.02976. doi: 10.48550/arXiv.2209.02976.

[33] R. Sapkota and M. Karkee, 'Ultralytics YOLO Evolution: An Overview of YOLO26, YOLO11, YOLOv8 and YOLOv5 Object Detectors for Computer Vision and Pattern Recognition', Oct. 15, 2025, *arXiv*: arXiv:2510.09653. doi: 10.48550/arXiv.2510.09653.

[34] Z. Tian, C. Shen, H. Chen, and T. He, 'FCOS: Fully Convolutional One-Stage Object Detection', Aug. 20, 2019, *arXiv*: arXiv:1904.01355. doi: 10.48550/arXiv.1904.01355.

[35] X.-Y. Jiang, K. H. B. Abas, W. Cui, A. R. B. Husain, and Y.-F. Zhao, 'A LaMa-based Panoramic Image Restoration Framework for Enhanced Omnidirectional Obstacle Detection in Low-Altitude UAVs', *J. Comput.*, pp. 159–178, Aug. 2025, doi: 10.63367/199115992025083604011.

[36] T. S. Prajwal and I. A K, 'A Comparative Study Of RESNET-Pretrained Models For Computer Vision', in *Proceedings of the 2023 Fifteenth International Conference on Contemporary Computing*, Noida India: ACM, Aug. 2023, pp. 419–425. doi: 10.1145/3607947.3608042.

[37] K. He, X. Zhang, S. Ren, and J. Sun, 'Deep Residual Learning for Image Recognition', Dec. 10, 2015, *arXiv*: arXiv:1512.03385. doi: 10.48550/arXiv.1512.03385.

[38] A. Jafar and M. Lee, 'High-speed hyperparameter optimization for deep ResNet models in image recognition', *Clust. Comput.*, vol. 26, no. 5, pp. 2605–2613, Oct. 2023, doi: 10.1007/s10586-021-03284-6.

[39] W. Xu, Y.-L. Fu, and D. Zhu, 'ResNet and its application to medical image processing: Research progress and challenges', *Comput. Methods Programs Biomed.*, vol. 240, p. 107660, Oct. 2023, doi: 10.1016/j.cmpb.2023.107660.

[40] Y. Hou, Z. Wu, X. Cai, and T. Zhu, 'The application of improved densenet algorithm in accurate image recognition', *Sci. Rep. Nat. Publ. Group*, vol. 14, no. 1, p. 8645, 2024, doi: 10.1038/s41598-024-58421-z.

[41] S. A. Samad and J. Gitanjali, 'Augmenting DenseNet: Leveraging Multi-Scale Skip Connections for Effective Early-Layer Information Incorporation', *IEEE Access*, vol. 12, pp. 141344–141360, 2024, doi: 10.1109/ACCESS.2024.3460830.

[42] K. Nanthini, D. Sivabalaselvamani, K. Chitra, P. Gokul, S. KavinKumar, and S. Kishore, 'A Survey on Data Augmentation Techniques', in *2023 7th International Conference on Computing Methodologies and Communication (ICCMC)*, Feb. 2023, pp. 913–920. doi: 10.1109/ICCMC56507.2023.10084010.

[43] 'What is Data Augmentation? - Data Augmentation Techniques Explained - AWS', Amazon Web Services, Inc. Accessed: Jun. 20, 2025. [Online]. Available: https://aws.amazon.com/what-is/data-augmentation/

[44] K. Maharana, S. Mondal, and B. Nemade, 'A review: Data pre-processing and data augmentation techniques', *Glob. Transit. Proc.*, vol. 3, no. 1, pp. 91–99, Jun. 2022, doi: 10.1016/j.gltp.2022.04.020.

[45] S. B. Wali *et al.*, 'Vision-Based Traffic Sign Detection and Recognition Systems: Current Trends and Challenges', *Sensors*, vol. 19, no. 9, Art. no. 9, Jan. 2019, doi: 10.3390/s19092093.
[46] 'Guide to Understanding the Research Onion | 15 Writers'. Accessed: Nov. 27, 2025. [Online]. Available: https://15writers.com/research-onion/
[47] K. E. Howell, *An Introduction to the Philosophy of Methodology*. SAGE, 2012.
[48] C. Maretha, 'Positivism in Philosophical Studies', *J. Innov. Teach. Instr. Media*, vol. 3, no. 3, pp. 124–138, May 2023, doi: 10.52690/jitim.v3i3.716.
[49] C. Bruce, 'Modeling the environmental collaboration process: A deductive approach', *Ecol. Econ.*, vol. 59, no. 3, pp. 275–286, Sep. 2006, doi: 10.1016/j.ecolecon.2005.10.015.
[50] N. F. Elias, 'Using a deductive approach in validating a measurement model', *Aust. J. Basic Appl. Sci.*, vol. 6, pp. 329–336, Aug. 2012.
[51] S. Sirisilla, 'Experimental Research Designs: Types, Examples & Advantages', Enago Academy. Accessed: Dec. 05, 2025. [Online]. Available: https://www.enago.com/academy/experimental-research-design/
[52] H. E. Kim, A. Cosa-Linan, N. Santhanam, M. Jannesari, M. E. Maros, and T. Ganslandt, 'Transfer learning for medical image classification: a literature review', *BMC Med. Imaging*, vol. 22, no. 1, p. 69, Apr. 2022, doi: 10.1186/s12880-022-00793-7.
[53] I. Kandel and M. Castelli, 'Transfer Learning with Convolutional Neural Networks for Diabetic Retinopathy Image Classification. A Review', *Appl. Sci.*, vol. 10, no. 6, p. 2021, Jan. 2020, doi: 10.3390/app10062021.
[54] S. R. Lenka, S. K. Bisoy, R. Priyadarshini, and M. Sain, 'Empirical Analysis of Ensemble Learning for Imbalanced Credit Scoring Datasets: A Systematic Review', *Wirel. Commun. Mob. Comput.*, vol. 2022, no. 1, p. 6584352, 2022, doi: 10.1155/2022/6584352.
[55] X. Wang and Z. Cheng, 'Cross-Sectional Studies: Strengths, Weaknesses, and Recommendations', *Chest*, vol. 158, no. 1S, pp. S65–S71, Jul. 2020, doi: 10.1016/j.chest.2020.03.012.
[56] 'Traffic Sign Dataset - Classification'. Accessed: Dec. 03, 2025. [Online]. Available: https://www.kaggle.com/datasets/ahemateja19bec1025/traffic-sign-dataset-classification
[57] V. R. Joseph, 'Optimal Ratio for Data Splitting', *Stat. Anal. Data Min. ASA Data Sci. J.*, vol. 15, no. 4, pp. 531–538, Aug. 2022, doi: 10.1002/sam.11583.
[58] 'Albumentations: fast and flexible image augmentations', Albumentations. Accessed: Dec. 06, 2025. [Online]. Available: https://albumentations.ai/
[59] Ultralytics, 'Image Classification'. Accessed: Jan. 13, 2026. [Online]. Available: https://docs.ultralytics.com/tasks/classify/
[60] 'Transfer Learning: Definition, Benefits & Applications | Ultralytics'. Accessed: Jan. 13, 2026. [Online]. Available: https://www.ultralytics.com/glossary/transfer-learning
[61] 'ImageNet Dataset - Ultralytics YOLO Docs'. Accessed: Jan. 13, 2026. [Online]. Available: https://docs.ultralytics.com/datasets/classify/imagenet/
[62] 'scikit-learn: machine learning in Python — scikit-learn 1.8.0 documentation'. Accessed: Jan. 13, 2026. [Online]. Available: https://scikit-learn.org/stable/