Road Obstacle Detection Using YOLO Algorithm based on Attention Mechanism

Submitted in fulfillment of the academic requirements for the degree of Master of Science in the School of Mathematics, Statistics and Computer Science, University of KwaZulu-Natal, Durban

Bafokeng Lekola (223153061)

July 18, 2024

Declaration

I, Bafokeng Lekola, declare that:

- The research work reported in this dissertation, except where otherwise pointed out is my original work.
- This dissertation has not been for any degree or assessment at any other Institution.
- Any part of this dissertation where I reviewed the published work of others, I clearly state and reference them.
- In this dissertation, no other person's graphs, images, data, tables or information is used unless correctly stated as being derived from others.

.....

Bafokeng Lekola

Declaration-Supervisor

As the candidate's supervisor, I agree to the submission of this dissertation

Signed: [redacted]. Name: ...Prof. S. Viriri ... Date: (30/08/2024)

Abstract

Road obstacle detection is an important task in autonomous vehicles (AVs) and advanced driver assistance systems (ADAS) as they require real-time operation and high accuracy for safe operation. The mobile nature of the task means that it is carried out in a low-resourced environment where there is a need for an algorithm that achieves both high accuracy and high inference speed while meeting the requirement for lightweight. In this dissertation, an exploration of the effectiveness of the Attention-enhanced YOLO algorithm for the task of road obstacle detection is carried out. Several state-of-the-art attention modules that employ both channel and spatial attention are explored and fused with the YOLOv8 and YOLOv9 algorithms. These enhance feature maps of the network by suppressing non-distinctive features allowing the network to learn from highly distinctive features. The Attention-modified networks are trained and validated on the Kitti and BDD100k datasets which are publicly available. Comparisons are made between the models and the baseline. An improvement from the baseline is seen with the GAM attention achieving an accuracy rate of 93.3% on the Kitti dataset and 71.1% on the BDD100k dataset. The Attention modules generally achieved incremental improvements over the baseline.

Acknowledgements

Thank you world!

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Introduction

Changes in the current technology landscape have seen a significant increase in computation capabilities brought about by advancements in technology. One such technology is cloud computing, which has brought forth exponential growth in the AI space by providing computing resources once unavailable to many people. This has led to the emergence of multiple breakthrough technologies. We have seen state-of-the-art performances from DL and ML fields, such as the usage of GANs (Generative Adversarial Networks) in computer vision and LLMs (Large Language Models) in NLP, bringing attention to the AI field. These models have been able to achieve state-of-the-art results by leveraging the high computing capabilities offered by expensive cloud service offerings. One of the technologies that has seen a rise is that of advanced driver assistance systems (ADAS) and autonomous vehicles which, coupled with the advancements in perception technology, compute capabilities, machine learning and deep learning have constituted a popular research area due to their potential to improve road safety and reduce accidents. One of the critical tasks for ADAS and autonomous vehicles is obstacle detection on the road. Accurate and timely detection of obstacles on the road is essential for ensuring the safety of passengers and other road users. The usage of onboard sensors like cameras and lidar is very important in this regard and allows for the necessary data collection for obstacle detection.

Deep learning-based object detection algorithms have shown great promise in detecting obstacles in real time, with the You Only Look Once (YOLO) algorithm gaining widespread popularity due to its speed and accuracy. However, the YOLO algorithm still faces challenges in detecting small or occluded obstacles, which can limit its practical

application.

To address the limitations of the YOLO algorithm, this dissertation proposes a YOLO algorithm based on an attention mechanism to enhance the detection accuracy of small and occluded obstacles on the road. Multiple attention mechanisms are incorporated into the YOLO algorithm to improve the focus of the model on relevant regions of the image, which can improve the accuracy of obstacle detection.

In this dissertation, we want to propose an algorithm and evaluate its performance on the BDD100K and the Kitti datasets.

1.2 Motivation

The motivation for this dissertation is to improve the accuracy and robustness of road obstacle detection in autonomous vehicles and ADAS systems. Accurate detection of obstacles on the road is crucial for ensuring the safety of passengers and other road users. The YOLO algorithm has shown great promise in detecting objects in real-time, but it still faces challenges in detecting small or occluded obstacles. To address these challenges, this dissertation proposes a YOLO algorithm based on an attention mechanism that can enhance the detection accuracy of small and occluded obstacles on the road. The proposed algorithm has the potential to significantly improve the safety of autonomous vehicles and reduce the incidence of accidents caused by obstacles on the road. By improving the performance of road obstacle detection, this dissertation can contribute to the development of safer and more reliable autonomous driving technology.

1.3 Problem Statement

In this dissertation, we address the problem of accuracy and robustness of road obstacle detection in autonomous vehicles and ADAS systems. The detection of obstacles on the road is crucial for ensuring the safety of passengers and other road users. In recent years, the You Only Look Once (YOLO) algorithm has emerged as a popular method for real-time object detection. However, the performance of the YOLO algorithm and other single-stage

object detection algorithms is often limited when detecting small or occluded obstacles on the road, which are common in real-world scenarios.

The limitations of the YOLO algorithm can pose significant challenges to the practical application of autonomous vehicles and ADAS systems. Small or occluded obstacles on the road can go undetected, which can lead to accidents and put the safety of passengers and other road users at risk. Therefore, there is a need to develop an algorithm that can accurately detect small and occluded obstacles on the road in real-time.

We propose a YOLO algorithm based on an attention mechanism that can enhance network features for improvement of detection accuracy of small and occluded obstacles on the road and network robustness. The proposed algorithm aims to address the limitations of the original YOLO algorithm and improve the performance of road obstacle detection. By improving the accuracy and robustness of road obstacle detection, this dissertation can contribute to the development of safer and more reliable autonomous driving technology. The proposed algorithm can have significant implications for the practical application of autonomous vehicles and ADAS systems, enabling them to operate more safely and effectively on the road. The improvement of the YOLO algorithm can also be useful for other object detection applications.

1.4 Research Objectives

The research objectives for this dissertation are as follows:

- To investigate the state-of-the-art techniques for automatic road obstacle detection using the YOLO algorithms.
- To model an Attention Network-based YOLO framework for accurate road obstacle detection.
- To evaluate the proposed framework against the state-of-the-art techniques for road obstacle detection.

1.5 Contribution of the Dissertation

The contributions made in this research are as follows:

- An Attention Network-based YOLO framework for accurate road obstacle detection.
- Exploration of the various Attention mechanisms into the neck region of the new YOLOv8 model and training for road obstacle detection
- Implementation of various Attention modules into the head region of the YOLOv9 algorithm and training for road obstacle detection.
- Benchmarking the modified YOLO algorithms against state-of-the-art algorithms.
- Evaluation of the performance of the Attention Network-based YOLO framework against other models explored by research.

1.6 Dissertation outline

The remaining section of the dissertation is separated into the following sections:

- Chapter 2: Background and related work. A review of autonomous driving and methods used prior.
- Chapter 3: Methods and Techniques.A detailed description of the proposed methods and techniques used to carry out experiments in this dissertation.
- Chapter 4: A presentation and explanation of the results of the experiments.
- Chapter 5: Conclusion. Gives the conclusion of the dissertation, state the limitation of the experiments and suggest further work to be done.

1.7 Conclusion

In this chapter, a broad idea for the dissertation is given. The motivation for the research, the objectives and the contributions in made in the dissertation are stated. Meeting the stated objectives of this dissertation can help researchers and practitioners in the field of autonomous driving to develop more accurate and robust algorithms for road obstacle detection, thereby improving the safety and reliability of autonomous vehicles and ADAS systems.

Chapter 2

Background and Related Work

2.1 Introduction

Object detection is a crucial task for various applications, such as autonomous driving, robotics, and surveillance. Depending on the field in question, emphasis is either made to tailor the object detection model for high accuracy or low latency these two metrics usually being inversely proportional thus, trading one for the other. In the case of autonomous vehicles and ADAS systems, we seek to find a balance between real-time object detection and accurate detection. This is because the system running on the vehicle has access to limited resources capabilities in power and computing but also requires to high accuracy for the sake of safe operation.

Multiple models have been proposed for this task which seek to achieve this trade-off balance. This models have had a problem with road obstacle detection for small obstacles. In this work, we seek to perform object detection using the YOLO algorithm with the addition of attention. Among the various object detection algorithms, single-stage algorithms have been seen to be the best for real-time detection and particularly the You Only Look Once (YOLO) algorithm has emerged as the most popular method due to its speed and accuracy. However, the performance of the YOLO algorithm can be limited in detecting small or occluded objects, which are common in real-world scenarios, such as road obstacle detection.

2.2 Autonomous Vehicles and ADAS

It is hard to imagine life without the automobiles that we rely on daily. This allows us to travel distances long and short and also allows the transportation of goods just as easily. The domain of autonomous vehicles has seen a huge list this recent surge in technological advancement such that multiple automobile manufacturers are investing billions in R&D in this space. With the usage of hardware like cameras, lidars and multiple sensor technologies, data is captured and processed for use in assistive tasks such as collision avoidance, pedestrian detection, traffic sign recognition, lane departure, etc thus improving the elimination of errors that may be caused by human incompetence/shortcomings.

Assistive driver assistance technologies and thus autonomous vehicles have improved over the years as seen in table 2.1 below. Achievement of such results means that the data processing should be done to the best possible standard to avoid causing more harm than what was intended by the adoption of the technology in the first place. This is why deep learning techniques have been used in the data processing part as they have been seen to produce the best results.

Levels	Features
0	Fully driven. No automation at all
1	Driver assistance. cruise control, automatic braking, and lane keeping are included
2	Driver required to observe. Some automated functions like steering assist, automatic parking, collision braking are included.
3	Conditionally automated. The vehicle can self-drive with driver observing the environent but the driver can take control at any time.
4	High automation. Under certain conditions, the vehicle can self operate.
5	Fully automated. The vehicle can self drive fully under any condition.

Table 2.1: Different levels of vehicle automationn

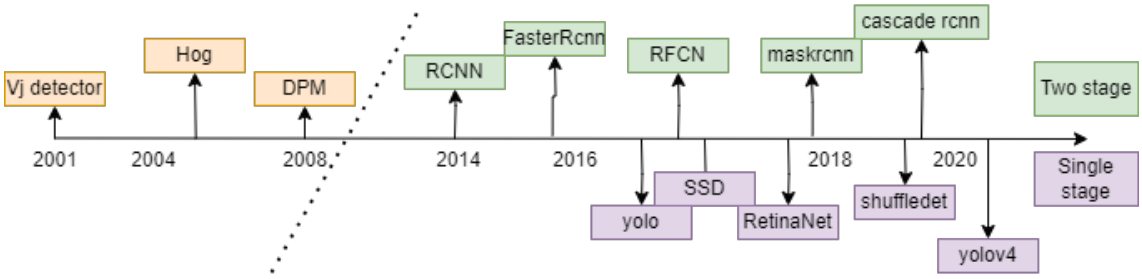


Figure 2.1: The progression of object detection algorithms [1]

2.3 Traditional Object detection

Early research in the field of object detection used multi-stage method of detection that included the proposal of regions in an image for processing, feature extraction from the proposed regions and then finally classification. These are seen in the first stages of figure 2.1 which shows the progression of object detection algorithms over time. The most commonly know of these feature extraction algorithms are the Scale Invariant Feature Transform (SIFT)[11], Local Binary Patterns (LBP)[12], Histogram of Oriented Gradients (HOG)[13] and the Haar wavelets [14]. These algorithms relied on identifying edges, key points and pattern matching in order to identify features present in image. These features were then passed through a classifier such as the support vector machine (SVM)[15]. The multiple stages included in this method and the number of operations performed means that this methods take a long time to process information and will require a lot of computing capabilities.

2.4 Deep Learning based Object detection

These are shown in the latter part of figure 2.1 where the arrival of GPU and large cloud computing platforms in deep learning research saw a rise in deep learning techniques used for object detection. Deep learning methods have proven to be more efficient for image processing tasks such as object detection, lane detection, image segmentation, etc, than the traditional methods.

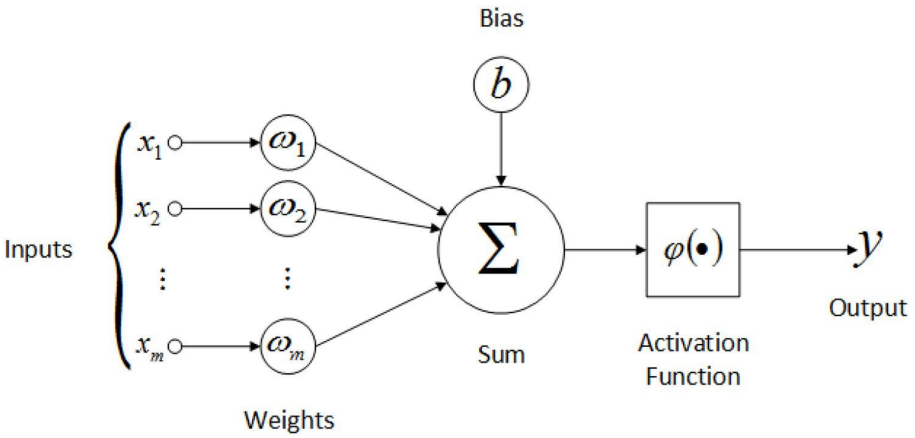


Figure 2.2: Artificial neuron

2.4.1 Artificial Neural Networks(ANN)

In deep learning, knowledge is drawn from the human nervous system to develop an artificial neural network structure. This consists of interconnected neurons that collectively learn the underlying structures from input data to determine the final output. These neurons are connected in layers that are stacked upon each other to form the structure of the ANN.

The artificial neuron, visually presented in figure 2.2 is mathematically presented such that, given an input feature vector (x), a weight vector (w) and bias term (b). Then equation 2.1, defines the weighted average of the features by the neuron:

$$y(x) = \varphi(wx + b) \tag{2.1}$$

Where σ is an activation function that ensures nonlinearity.

For data to be input into the network, it has to be formed into a 1D vector in order to be propagated into the network such that if an input image of 28×28 pixels like in the MNIST dataset of [16] was to be the processed, the input to the ANN would be formed into a 1D vector of length 784. This would work for a small image resolution of this size but quickly becomes intractable with increasing resolution. This presents a Deep learning-based object detection algorithm in its simplest form.

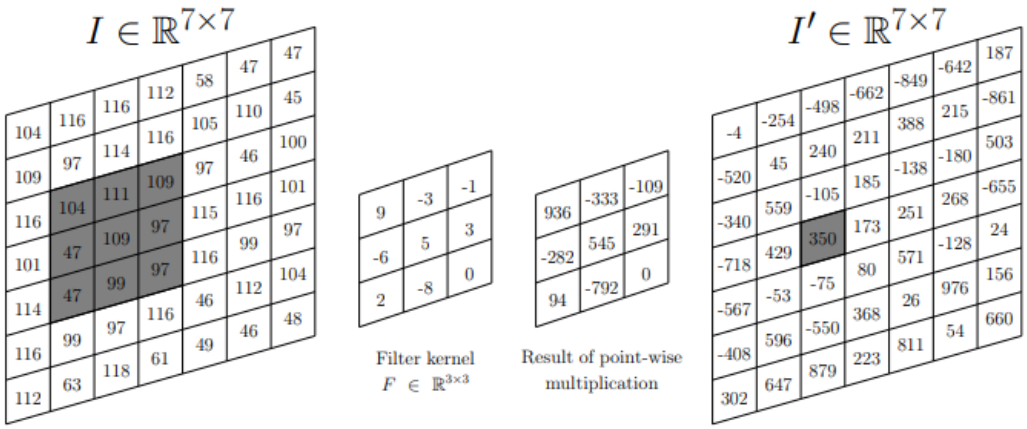


Figure 2.3: Visualization of the application of an image kernel [2]

2.4.2 The Convolutional Neural Network(CNN)

The CNN was introduced by Yan LeCun et al [16] on the task of handwritten digit recognition on the MNIST dataset in the 90s. This network differs from the normal ANN by the use of the convolution operation for feature processing of the image. The convolution operation involves the use of an image filter/kernel $F \in \mathbb{R}^{kw\times kh\times d}$, where kw represents the filter's width, kh the filter's height and d the number of input channels is passed over an image $I \in \mathbb{R}^{w\times h\times d}$ in a sliding window manner and dot product computed on all of the image positions to produce a new image I'. This operation is represented in a simplified form in figure 2.3 where only a 7×7 image and a 3×3 kernel are used. Equation 2.2 shows the convolution operation across the three dimensions.

$$I'(x,y) = \sum_{i_x=1-\lceil \frac{kw}{2} \rceil}^{\lfloor \frac{kw}{2} \rfloor} \sum_{i_x=1-\lceil \frac{kh}{2} \rceil}^{\lfloor \frac{kh}{2} \rfloor} \sum_{i_c}^{d} I(x+i_x, y+i_y, i_c) F(i_x, i_y, i_c) \tag{2.2}$$

The convolution operation makes it possible for features such as edges and corners making it possible to detect objects in images.

Traditional CNNs are made up of three basic block: Convolutional layers, the Pooling layer and the normalization layer.

Convolution Layer

The convolutional layer is constituted by n number of filters that are convolved with the input image to produce n number of output feature maps with the associated weights

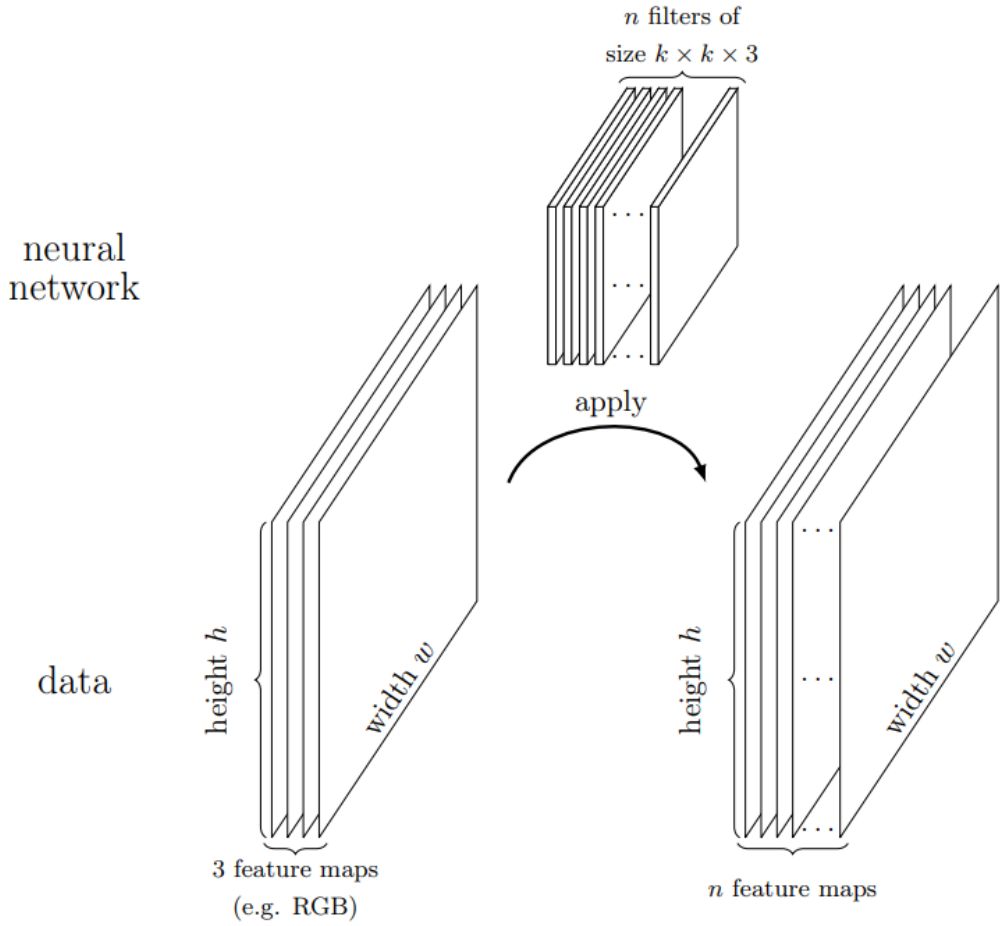


Figure 2.4: Visualization of the convolutional layer[2]

of the image filter being the parameters to be adapted during the training of the CNN. The convolutional layers is characterized by 4 hyperparameters which are the the number of image kernels in the layer n, the size of the image filter $k_w \times k_h$, the activation function and the stride $s = \mathbb{N}_{\geq}1$. Figure 2.4 shows an example visualization of the convolutional layer showing the input feature map and kernels to output feature maps.

Pooling Layer

In a convolutional neural network, after a convolutional layer, there is a need to add a pooling layer. This is because the convolutional layer records the precise location of features in the input. This means that small changes in the location of features in the input will create new feature maps. The result of using a pooling layer and creating a down-sampled feature map is a summarized version of the feature map of the detected features in the input. The reason for using the pooling layer is to enforce translation invariance

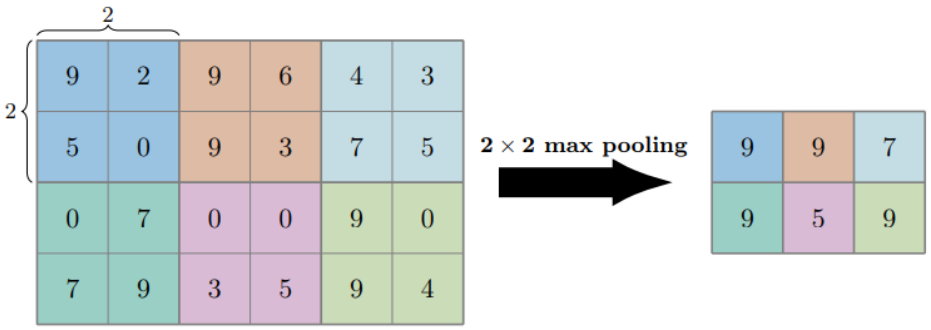


Figure 2.5: 2×2 max pooling operation of a input of 6×2 and stride $s = 2$

and by using a stride $s > 1$ pooling does a data reduction. The two most common ways in which pooling is done are max pooling and the average pooling functions. Max pooling is defined by the max function defined in 2.3 and the average pooling is defined by equation 2.4

$$Max\_pooling : \max a \in A \tag{2.3}$$

$$Average\_pooling : \frac{1}{|A|} \sum a \in A^a \tag{2.4}$$

An example of the max pooling operation can be seen in figure 2.5 the max of pixels around a kernel neighbourhood 2×2 is taken using a stride of $s = 2$.

Fully Connected Layer

This layer's design connects all the neurons of the previous layer to the neurons of the current layer as seen in figure 2.6. It is usually used to link the features extracted from the lower levels of the CNN to the output classes of the network. This layer constitutes a lot of the parameters of the network because of its full connectedness and will be responsible for the longer training time of the network. Stacking the convolutional layer, pooling layer and fully connected layers in certain optimal configurations yields CNN architectures that perform perfectly well for feature extraction purposes and therefore start-of-the-art performance on computer vision tasks.

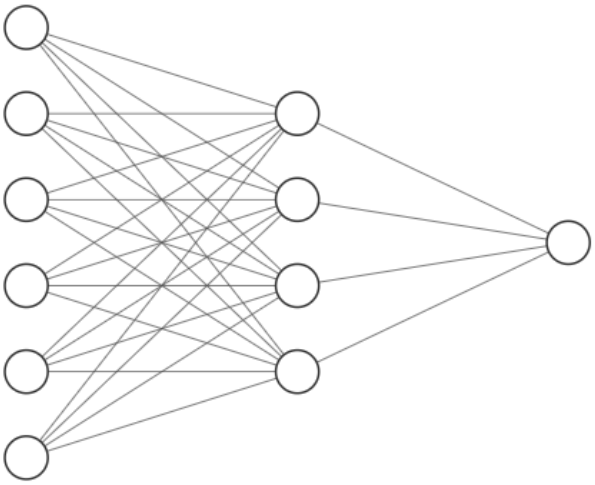


Figure 2.6: The fully connected layer

2.4.3 Nonlinear Activation functions

The activation functions of the convolutional layers are an integral part of the learning process of CNNs and allow them to learn non-linear relationships in data. This is important because the underlying structures in images cannot be approximated by linear functions. A few of the these activation functions are discussed below.

Sigmoid

A differentiable real function is defined for all real values, sometimes referred to as the logistic function. This implementation of this function drew inspiration from the firing of biological neurons and is mathematically presented in equation 2.5. The output of this function tends to saturate for higher and lower inputs.

$$\sigma(x) = \frac{1}{1 - e^x} \tag{2.5}$$

The sigmoid is usually used at the end of CNNs in order to predict class probabilities for multiclass classification tasks.

Rectified Linear Unit

This function solves the computational complexity problem presented by the sigmoid function. As presented by equation 2.6 it can be seen that x remains unchanged for

positive values and 0 for negative values.

$$ReLU(x) = max(0, x) = \begin{cases} x, x \geq 0 \\ 0, otherwise \end{cases} \tag{2.6}$$

The ReLU function presents a problem of the vanishing gradient for negative values which are set to 0. The activation function has been extensively used regardless of this with positive results. To mitigate against the vanishing gradient problem, the Leaky ReLU function has been utilized by allowing the use of negative values as is presented by equation 2.7.

$$ReLU(x) = max(0, x) = \begin{cases} x, x \geq 0 \\ 0.01 \times x, x < 0 \end{cases} \tag{2.7}$$

Softmax

Using a vector (logits) of real values, the softmax activation function calculates the probability distribution across several classes. Its outputs are in the range $sigma(x) \in [0, 1]$, and a cumulative sum of 1. It is mathematically defined as:

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{2.8}$$

2.4.4 Training the Convolutional Neural Network

Successfully training a convolutional neural network requires the network's predicted output to be the same as the ground-truth label defined for the input. This task requires the minimization of an objective function that is defined for the network. This object function is called the loss function is essential to deep learning. There exist multiple loss functions many of which are useful in some tasks but not other. A few of the common loss function are discussed below.

Mean squared error(MSE)

The Mean Squared Error (MSE) is the average sum of the squared distances of a network's predicted output and the ground truth values. The function is defined in equation

2.9 below:

$$MSE = \frac{1}{n}\sum(y_{pred} - y_{true})^2 \tag{2.9}$$

MSE is very sensitive to outliers because of the squaring of the error. This is a loss function that is used for regression purposes and, therefore, used for predicting continuous values.

Mean absolute error(MAE)

Calculates the average of the absolute value of the difference between predicted labels and ground truth this means it is less sensitive to outliers that the MSE. It is defined in equation 2.10 below:

$$MAE = \frac{1}{n}\sum|y_{pred} - y_{true}| \tag{2.10}$$

Cross entropy loss

Measures the performance of a classification model whose output is a probability distribution and is defined in equation]2.11 below.

$$\text{Cross-entropy} = -\sum_{i=1}^{d}(y_{true} \times \log(y_{pred})) \tag{2.11}$$

Where y_{true} is a one-hot vector of the true label, and y_{pred} is the output vector of the softmax activation function. The one-hot vector is encoded as:

$$y_i(x) = \begin{cases} 1\ if y_i\ = j \\ 0\ if y_i\ \neq j \end{cases} \tag{2.12}$$

Where j is the index position of the true label in the one-hot vector.

Focal loss

Addresses the problem of imbalanced classes by assigning different weights to different classes. The aim is to force the model to focus on learning the harder class rather than overfitting on the easy samples. The weighting reduces the influence of the easier sample on the loss function allowing for the focus on the harder sample. A modulating factor

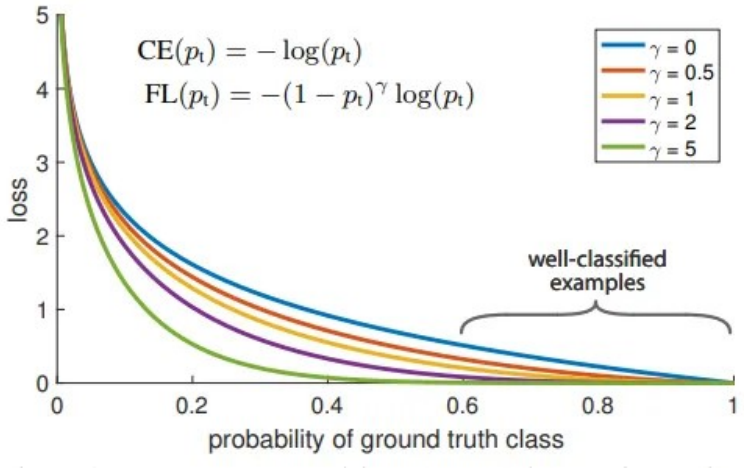


Figure 2.7: Graphs of varying γvalues [3]

$(1 - Pt)^\gamma$ [3]is added to the cross entropy as seen in 2.13 .

$$FL(Pt) = -\alpha_t(1 - Pt)^\gamma \log(Pt) \tag{2.13}$$

Where γ is the modulating parameter to be adjusted and α is the weighting parameter usually set to 0.25. A $\gamma > 0$ value reduces the relative loss of the easy samples. Figure 2.7 shows the effects of varying γ.

Gradient Descent

This is the method with which network weights are iteratively updated in the network during learning. After the computation of the losses, negative gradients are computed over the training data and parameter updated as per the equation 2.14.

$$\theta = \theta - \eta * \nabla_\theta L(\theta) \tag{2.14}$$

Where, η is the learning rate, θ is the vector of network parameters(weights and biases), $\nabla_\theta L(\theta)$ represents the gradient computed with respect to the network parameters θ.

Stochastic Gradient Descent

Extends the Gradient Descent algorithm by computing gradients on mini-batches picked at random rather than on single inputs. This allows for training to be done quicker

than in the ordinary gradient descent. The randomness of samples pulled per mini-batch also helps the model to avoid getting stuck in a local minima. Due to noisy gradients or a lot of curvature in the search space, SGD with mini-batches can occasionally show sluggish convergence when estimating parameter updates on tiny batches [17]. Utilizing momentum in conjunction with SGD decreases the impact of noise and expedites convergence by adding a hyper-parameter that maintains momentum in the direction of search. It is stated as:

$$v = \alpha v - \eta * \nabla_\theta L(\theta; x^{(i:i+n)}; y^{(i:i+n)}) \tag{2.15}$$

α is the momentum hyper-parameter, which determines the impact of previous gradients, and v is the accumulation of gradients. The greater the *alpha* value in relation to the η learning rate, the greater the impact of previous gradients on the direction of search

Backpropagation

Works together with the gradient descent algorithm by computing gradients. The chain rule is used to compute the partial derivatives with respect to the network parameters. The chain rule to compute the partial derivative for a single weight in the network in figure 2.8 is presented in equation 2.16. To minimize a cost function C with respect to weights

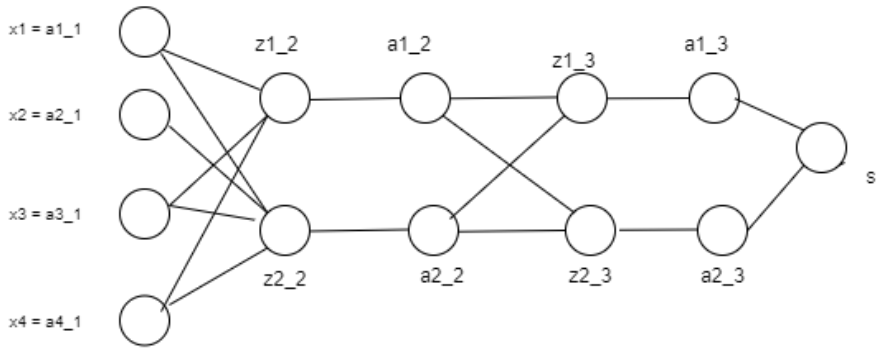


Figure 2.8: Basic neural network for demonstration of backpropagation

w_{22} connecting $z2_2$ and $a2_2$ the chain rule can be written as follows:

$$\frac{\partial C}{\partial w_{22}} = \frac{\partial C}{\partial z2\_3}\frac{\partial z2\_3}{\partial w_{22}} = \frac{\partial C}{\partial a2\_3}\frac{\partial a2\_3}{\partial z2\_3} \tag{2.16}$$

2.4.5 Novel architectures for object detection networks

Deep convolutional networks feature multiple layers stacked together which are important for the detection task to happen. These are work perfectly well for the task of object detection.

Non maximal impression(NMS)

It is possible to have an object detection network generating multiple bounding boxes for the same object because of noise or overlap. NMS removes these extra bounding boxes by eliminating the bounding boxes that have a lower confidence score. The NMS algorithm checks for the bounding box with the highest confidence and checks the IOU overlap between that and the other bounding boxes. An overlap over a certain threshold leads to the other bonding box being eliminated. The algorithm for NMS is seen in figure 2.9. Figure 2.10 visualizes the bounding boxes before and after non-max supression. NMS

Algorithm 1 Non-Max Suppression

1: **procedure** NMS(B,c)
2: $B_{nms} \leftarrow \emptyset$ Initialize empty set
3: **for** $b_i \in B$ **do** => Iterate over all the boxes
4: $discard \leftarrow$ False Take boolean variable and set it as false. This variable indicates whether b(i) should be kept or discarded
5: **for** $b_j \in B$ **do** Start another loop to compare with b(i)
6: **if** same$(b_i, b_j) > \boldsymbol{\lambda_{nms}}$ **then** If both boxes having same IOU
7: **if** score$(c, b_j) >$ score(c, b_i) **then**
8: $discard \leftarrow$ True Compare the scores. If score of b(i) is less than that of b(j), b(i) should be discarded, so set the flag to True.
9: **if not** $discard$ **then** Once b(i) is compared with all other boxes and still the discarded flag is False, then b(i) should be considered. So add it to the final list.
10: $B_{nms} \leftarrow B_{nms} \cup b_i$
11: **return** B_{nms} Do the same procedure for remaining boxes and return the final list

Figure 2.9: The non-maximal impression algorithm

is an important algorithm that has been used extensively in the field of object detection.

Spatial Pyramid Pooling

The SPP layer removes the requirement of neural networks only being able to take only fixed-size inputs. Usually, the later layers of the network that perform classification only take a fixed input size. The SPP layer is placed at the end of the feature extraction network to pool features of varying input size as shown in figure 2.11 to generate a fixed-length

Figure 2.10: Before and after application of NMS

feature representation. The SPP layer is seen in figure 2.11

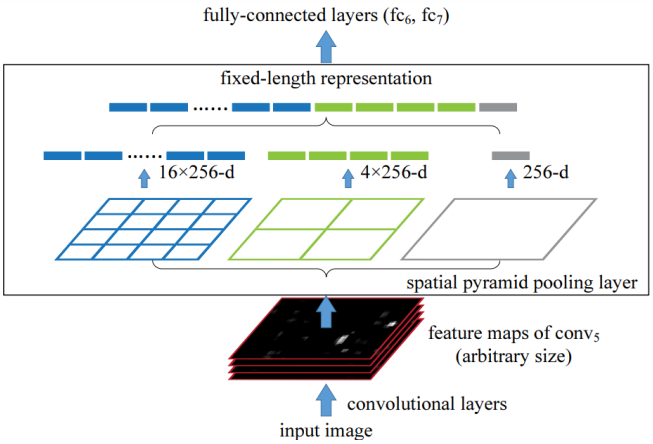


Figure 2.11: A network structure with a spatial pyramid pooling layer [4]

The SPP is very important and is key to the operation of most state-of-the-art object detection networks.

2.4.6 Deep learning-based object detection algorithms

These are methods that utilize the CNN architecture for feature extraction instead of using the traditional methods of section 2.3 above. These usually have a predefined convolutional network termed the backbone of the network that performs the feature extraction and then classification and localization done on the extracted features in the final stages of the network. Some of the CNNs commonly used as network backbones are as seen in table 2.2. These networks usually have varying parameter sizes and network depth which directly influence their performance in feature extraction. Several algorithms have emerged in the field of object detection that utilize these backbone networks and

Model	#Parameter(M)	ImageNet-1k Top-1 Accuracy (%)	Year
Alexnet[18]	62.36	63.32	2012
VGG[19]	138.37	73.01	2014
GoogLeNet[20]	25.56	76.13	2014
EfficientNetV2-S[21]	21.5	84.23	2021
ShuffleNetV2[22]	7.4	76.23	2018
MobileNetV3-Large[23]	5.5	75.27	2021

Table 2.2: Popular Backbone/feature extraction networks

have been seen to achieve state-of-the-art results in their respective time of release. These can be classified into two categories mainly the single-stage detectors and the two-stage detectors.

2.4.7 Two Stage detectors

These utilize a two-stage detection pipeline. Firstly performing proposal of regions of interest before performing object detection. This allows the network to better focus on only those parts where an object is thought to be. This makes them a lot more accurate but with the drawback of being computationally expensive because in addition to the ROI proposal, detection is made on multiple images (regions of interest) instead of one image.

Faster-RCNN

The Faster-RCNN [5] algorithm builds on top of the fast RCNN [24] algorithm by sharing the convolutional layers for the two regions: the Region Proposal Network(RPN) and the Fast-RCNN detector as can be seen by the separation in the two paths after the conv layers in figure 2.12. This works such that, the RPN points the detector in the right direction for detection. Because of the separate regions of the network, there is a need to define custom training strategies. Multiple techniques are implemented for training the network with the common one and the one used in the original paper being the alternating method. Training using this method is done by first training the RPN. The resulting regions are then used to train the detector and thus, simultaneously fine-tuning the shared convolutional network. The resulting network is then used to initialize the training for the RPN region again. This steps are repeated until desirable results are

achieved.

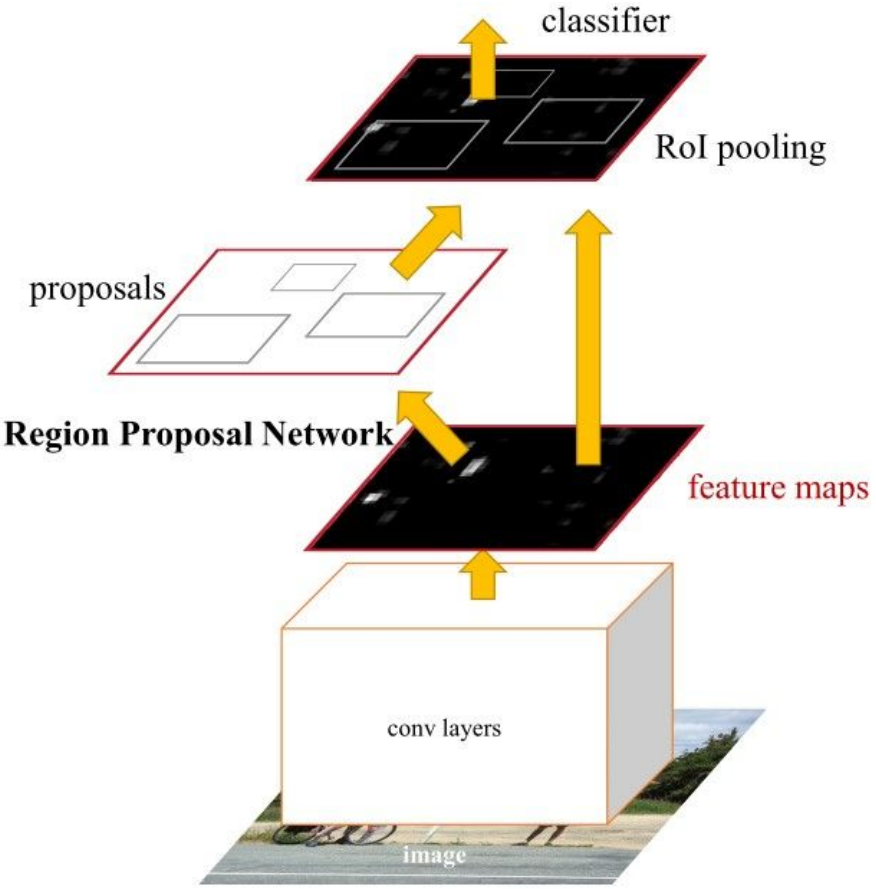


Figure 2.12: The faster RCNN network[5]

RFCN

Builds on the Faster RCNN method by introducing position-sensitive score maps using the convolutional neural network. This aims to address the problem that comes with the need for position sensitivity in object detection in an image and the translation invariance required for the classification of the image in the shared network. Figure 2.13 shows the architecture of the RFCN network showing the positioning if the backbone, position-sensitive score maps and the region proposal network regions.

2.4.8 Single Stage Detectors

These methods utilize a single pass through the network to perform detection from the whole image without region proposal. By sampling the image at different scales, multiple objects can be detected in the image. By utilizing many different strategies, these methods

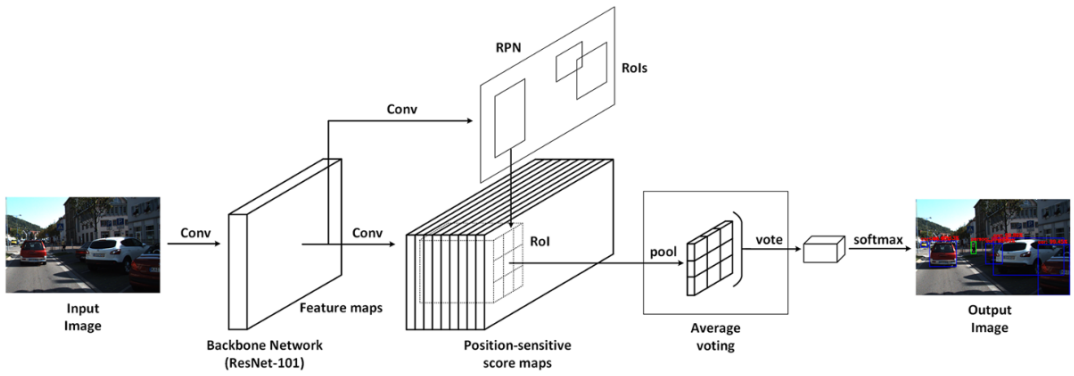


Figure 2.13: R-FCN architecture[6]

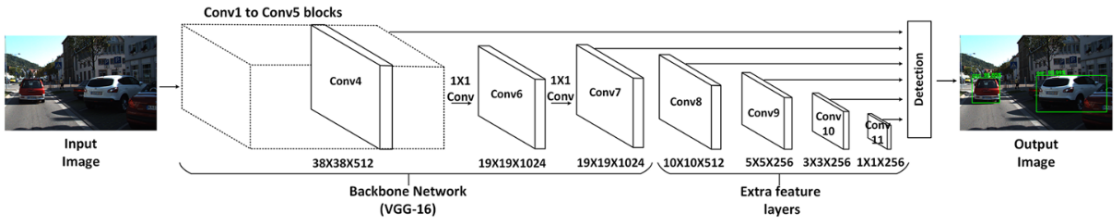


Figure 2.14: SSD architecture [7]

can achieve good accuracy while also having the added advantage of performing fast real-time inference.

SSD

Anchor boxes are generated at different scales/ratios for the different size feature map locations. The network creates scores for each object category present in each default box during prediction time and modifies the box to better fit the object's shape. To handle objects of varied sizes naturally, the network additionally incorporates predictions from numerous feature maps with different resolutions. The SSD network uses the VGG 16[19] network for feature extraction ontop of which 6 more convolutional layers are added for detection. Since SSD wraps all computing in a single network, it is simpler than approaches that require object proposals since it does away with the need for proposal production and any following stages of pixel or feature resampling. Figure 2.14 shows the architecture of the SSD network showing the linear nature of the single-stage object detectors showing the sufficiency of the one stage. This SSD algorithm operates faster while also achieving comparable accuracy with the slower more accurate 2-stage detection networks.

RetinaNet

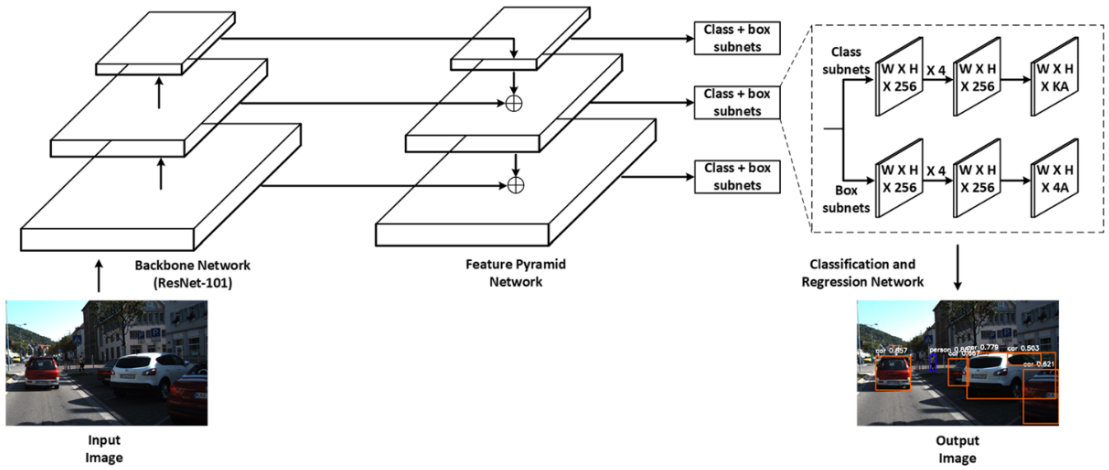


Figure 2.15: Architecture of RetinaNet[8]

Proposes the use of the focal loss [8] function to replace the binary cross entropy function that was commonly used before for object detection. This loss function adds a weighting on the detection loss of difficult sample on the easier samples, offsetting the loss of one from the other in order to counter the issue of the class imbalance problem usually encountered from the foreground background classes. The weights for the easy samples should be decreased relative to the increase of the difficult samples to achieve better classification. The RetinaNet uses the Resnet [25]as the backbone and the FPN[26] as the feature extraction network. This network achieved state-of-the-art results and outperformed two-stage detectors and the SSD in detection accuracy on the COCO dataset. Figure 2.15 shows the architecture of the RetinaNet network.

2.4.9 The YOLO Algorithm

This is the most popular single-stage detection algorithm that has seen adoption in many applications since its inception. There have been multiple improvements to the original YOLO algorithm that have seen the emergence of what we term the YOLO series of algorithms. These algorithms are a progressive improvement and introduce multiple novel methods to increase the accuracy and efficiency of the original algorithm. For example, the YOLOv2 [27] algorithm introduces batch normalization, and high resolution and introduces the Darknet-19 backbone for feature extraction to name a few improvements. Other improvements include adding a spatial pyramid pooling network (SPP) [28] to gather fea-

tures from multiple scales, allowing for a larger receptive field, introducing anchor boxes and then introducing an anchor-free approach in the latest YOLOv8 algorithm, different loss functions, etc. Outside of the YOLO series, attention mechanisms have been proposed to enhance the detection accuracy of small and occluded objects.

2.4.10 Attention Mechanism

Attention mechanisms have been widely used in natural language processing and computer vision to selectively focus on relevant features or regions of interest. In object detection, attention mechanisms can help the model to selectively attend to the most informative features and suppress the irrelevant ones, which can improve the accuracy of detection [29]. Several recent studies have explored the use of attention mechanisms in object detection, including in YOLO-based methods.

Several studies have applied attention mechanisms to the YOLO algorithm to improve its performance in detecting small objects. [30] proposed an attention-based YOLOv5 algorithm for multispectral pedestrian detection to provide robust and accurate detection under different illumination conditions, which achieved state-of-the-art performance on benchmark datasets. Similarly, Chen et al. [31] proposed an attention-guided Tiny-YOLOv4 algorithm for vehicle and pedestrian detection to solve the problem of rapidly changing detection scales due to a fast-moving vehicle, which also outperformed other state-of-the-art object detection algorithms. These studies and more show that attention is useful in improving the YOLO models.

Similarly, [32] proposed an attention-based YOLOv5 algorithm for traffic sign recognition. The algorithm used an attention mechanism to focus on the regions of interest and improve the detection accuracy of small traffic signs on the traffic sign dataset named TT100K. The study showed that the speed of the attention-based model had significant improvements in accuracy on the small target objects and the model can meet the requirements for a vehicle-mounted detection system.

Overall, these studies suggest that attention mechanisms can be effective in improving the performance of the YOLO algorithm for road obstacle detection. This paper aims to

contribute to this research by proposing a YOLO algorithm based on an attention mechanism that can accurately detect small and occluded obstacles on the road in real time. The proposed algorithm will be evaluated using benchmark datasets for road obstacle detection, and compared with other state-of-the-art object detection algorithms. The study aims to provide insights into the impact of the attention mechanism on the detection accuracy and processing speed of the YOLO algorithm, and the robustness of the algorithm in different lighting and weather conditions.

2.5 Conclusion

In this section, sufficient deep-learning literature was introduced outlining the traditional methods and their drawbacks and then deep neural networks. The convolutional neural network and its structures were studied thoroughly to give a good understanding of its importance in computer vision. This forms a good base for the next chapter that presents the methods to be used in this dissertation.

Chapter 3

Methods and Techniques

3.1 Introduction

Road obstacle detection for ADAS and autonomous vehicles is a task that requires real-time and accurate performance because of the urgency in the domain of operation. In this paper, road obstacle detection using the YOLO algorithm performed.This algorithm has been proven to be one of the algorithms that has the best balance of the two requirements of speed and accuracy. A drawback of the previous iterations of the YOLO architectures that has been well noted is that the detection of small or occluded objects is shallow and confidence in the algorithm in this regard is very low. We propose the addition of attention to two of the most recent iterations of the YOLO algorithms, the YOLOV8 and the YOLOv9 and to compare the performance of the models. The attention mechanism selectively weights the features from different layers of the backbone network to focus on the regions of interest, which are more likely to contain small or occluded obstacles. The multi-scale features extracted by the attention mechanism are then used by the detection head of YOLO to predict the bounding boxes and class labels of the objects in the image. Usage of mutliple attention mechanisms is explored and compared.

3.2 YOLO algorithm

Since its inception, the YOLO algorithm has proven to be one of the best algorithms for real-time object detection. It has proven to be lightweight and fast while also achieving state-of-the-art results. Over the year there have been multiple Yolo algorithms developed by different people across different research labs and figure 3.1 shows the progression over

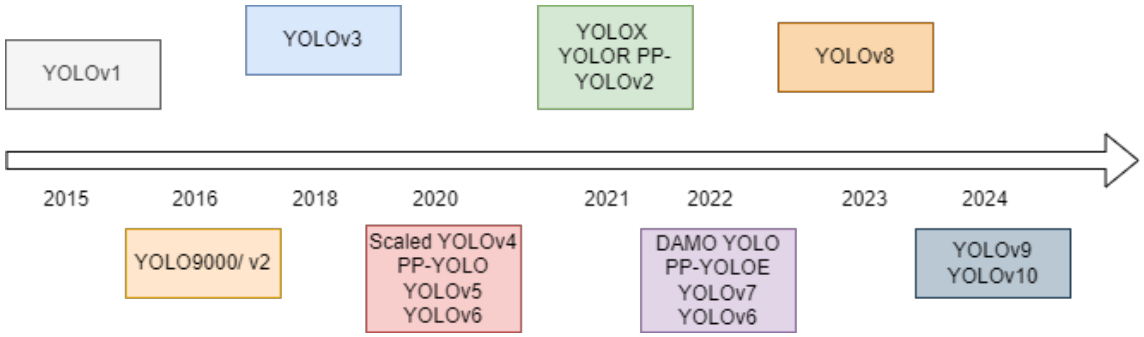


Figure 3.1: Progression of the YOLO algorithm

the years. The different yolo algorithms will be discussed below.

Since its inception, the YOLO algorithm has seen multiple improvements. It is a single-stage detection algorithm that uses a single pass of the network to perform object detection and as such, has been seen to achieve high inference time as compared to other algorithms. The YOLO architecture is composed of a backbone network typically an already existing and pre-trained network like the VGG16 or EfficientNet[33] to extract the low-level features in the images, the neck which would usually use a feature pyramid such as the FPN[26] to aggregate features from different scales to detect object at multiple scales and then the detection head which consists of convolutional layers that predict the bounding boxes and the class probabilities in the image.

The YOLOv1[34] algorithm pioneered real-time object detection by utilizing a single-stage detection pipeline to predict bounding boxed and class probabilities directly from an image grid while achieving real-time operation and reasonable accuracy. YOLOv2[35] can as the successor to YOLOv1 and introduced batch-normalization for faster training convergence, high-resolution feature maps and anchor boxes for better prediction of bounding boxes. YOLOv3[36] added a new deeper backbone network for better feature extraction and also incorporated the spatial pyramid for the utilization of features at different scales. From YOLOv4[37] advanced modifications were done to the backbone and spp architectures to achieve superior performance over previous models. A modified spatial attention module (SAM)[10] was incorporated and generic algorithms were employed to find optimal hyperparameters during training.

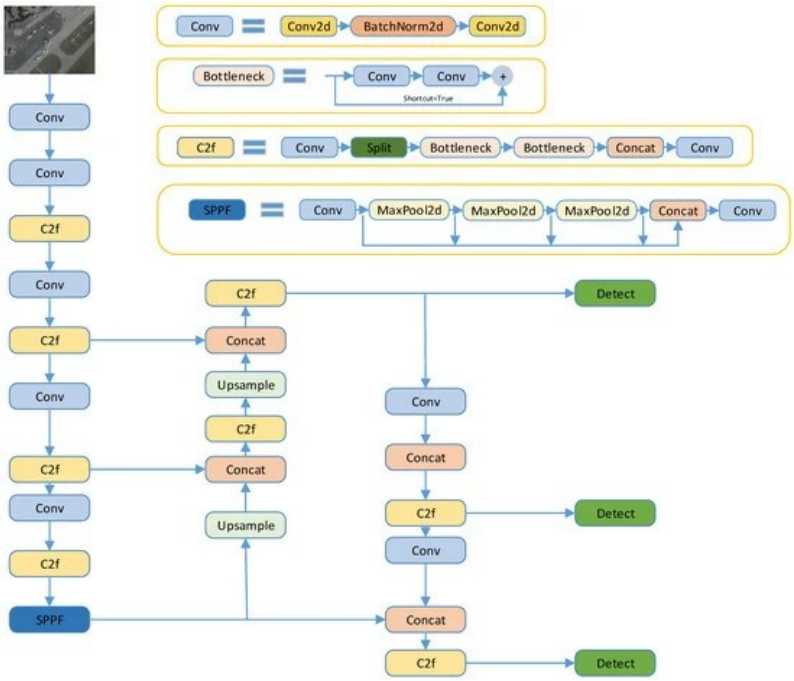


Figure 3.2: The YOLOv8 architecture

3.3 Network structure of the YOLOv8

The YOLOV8 algorithm is one of the latest iterations of the YOLO algorithms which was presented in 2023 by Glen Jowcher at Ultralytics whose structure is visualized in figure 3.2. This algorithm has been to proven to achieve better performance than all previous versions.

The key modifications added to the YOLOv8 include:

CSPDarknet53, a Darknet architecture variation, as its feature extractor. SiLU activation functions, batch normalization, and convolutional layers make up this component. Notably, for better feature extraction, YOLOv8 substitutes a 3x3 convolutional layer for the original 6x6 convolutional layer.

Cross-Stage Partial Bottleneck (C2f) Module: YOLOv8 presents the C2f module, which efficiently combines contextual data with high-level features. The bottleneck blocks' outputs, which are two 3x3 convolutions with residual connections, are concatenated to accomplish this. The goal of this architectural modification is to improve feature representation and to balance between lightweight implementation and good performance. The inspiration for this module is drawn from the C3 module of the YOLOv5 architecture.

Spatial Pyramid Pooling Fast (SPPF), to capture feature at multiple scales enhancing the performance of the algorithm at different object sizes.

Neck and Head A Path Aggregation Network (PANet)[38] is introduced as the neck structure of YOLOv8. By facilitating information flow across various spatial resolutions, PANet helps the model effectively capture features at several scales. By using an anchor-free detection approach, YOLOv8 predicts object centers without the need of predefined anchor boxes. It makes use of an altered YOLO head that has a new IoU (Intersection over Union) loss function and dynamic anchor assignment. These enhancements help handle overlapping items better and forecast bounding boxes with more accuracy. It comprises of multiple detection heads that make up the complete head structure that are responsible for projecting bounding boxes, class probabilities, and objectness ratings on various scales. The model comes in multiple sizes and depending on the speed to accuracy tradeoff that is acceptable, anyone of the models can be chosen. In this paper we chose to use the large model because of the comparison to be done with the YOLOv9 model which comes with the number of parameters equal to that of the YOLOv8l.

3.4 Network structure of the YOLOv9

This algorithm addresses the non-negligible information loss encountered during the feedforward process which leads to diminished information being used to upgrade the weights of the network. This information loss is detrimental in the usage of deep CNN for small object detection as small objects occupy a small portion of the network and thus are very susceptible to being lost due to diminished gradients. The

Programable Gradient Information (PGI):In order to enhance the model's detection capacity, PGI provides an auxiliary supervision framework intended to control the transmission of gradient information across multiple semantic levels. The primary branch, auxiliary reversible branch, and multi-level auxiliary information make up PGI. The main branch manages both forward and back propagation during the inference phase. An information bottleneck could develop as the network gets deeper, resulting in loss functions that are unable to provide helpful gradients. Reversible functions are then used by the auxiliary branch to feed back the necessary information back into the main branch to

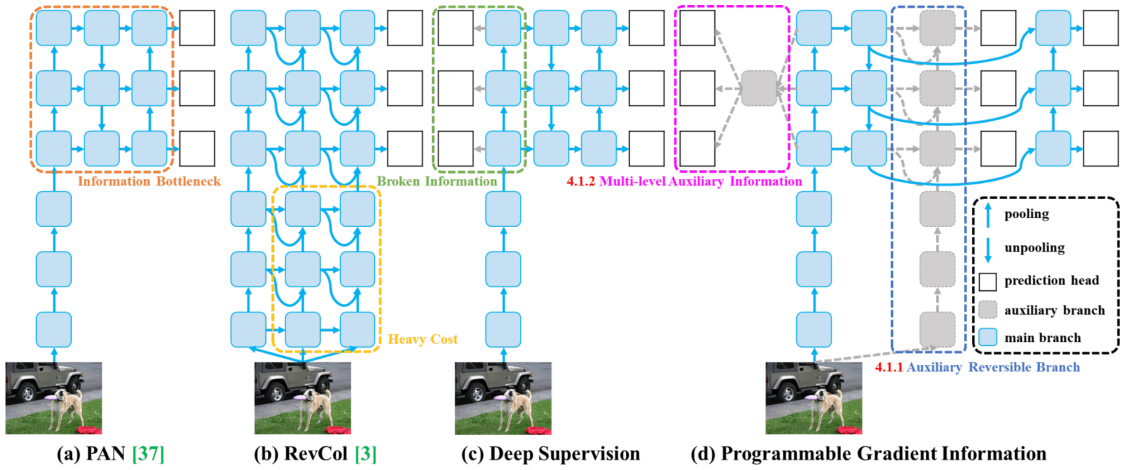


Figure 3.3: Structure of the YOLOv9 PGI[9]

protect information integrity and lessen information loss. Since the auxiliary branch is only used for the propagation of gradients, it can be removed after training and inference performed only by the main branch. The multi-level auxiliary information is then used to aggregate returned gradients from several prediction heads by inserting an integration network between the main branch and the feature pyramid hierarchy tiers of auxiliary supervision to tackle the problem of error accumulation from the deep supervision mechanism, enhancing the model's learning ability by adding additional information at different levels. the structure of the PGI is as seen in fig 3.3, showing the structure of branches employed.

Generalized Efficient Layer Aggregation Network Combines concepts from the CSPNet[39] architecture and the ELAN[40] which are designed for gradient path planning. Provides a lightweight architecture that helps the network to more efficiently use information across layers and helps the network to avoid the information bottleneck problem leading to increased accuracy and speed.

3.5 Attention Mechanisms

3.5.1 The Convolution Block Attention Module (CBAM)

This attention block module includes the usage of the channel[41] and spatial attention[10] modules applied sequentially as illustrated in figure 3.4. This allows the network to compute attention maps across the spatial and channel dimensions which are then multiplied with the input feature maps for feature refinement. The structures of the attention modules are shown in figure 3.5. The CBAM module outputs the two attention maps as, a 1D channel attention map $S_c \in \mathbb{R}^{C\times1\times1}$ and a 2D spatial attention map $S_s \in \mathbb{R}^{1\times H\times W}$ a process which is represented by the equations:

$$F' = S_c(F) \odot F \tag{3.1}$$

$$F'' = S_s(F') \odot F' \tag{3.2}$$

With $\odot$ denoting an element-wise multiplication operation. Both attention maps are then used together in a sequential manner such that F'' is a result of the application of both attention maps.

The channel attention performs both a global average pooling and a max pooling on the input feature map along the spatial dimension to aggregate spatial information across the feature map. The authors propose this to be the best way to get a good representation of features in the network rather than utilizing only one of them. These feature descriptors are then forwarded to a shared multi-layer perceptron to output two feature maps after which an element-wise summation is performed to produce the final output channel attention map $S_c \in \mathbb{R}^{C\times1\times1}$. The equation for this can be written as :

$$S_c = \sigma(MLP(AvgPool(F)) + MLP(MaxPool(F))) \tag{3.3}$$

The channel attention map is meant to give information about what is important in the image. The spatial attention module will then perform global average pooling and global max pooling along the channel dimension in order to aggregate inter-channel interactions. This is important to define where in the image the important features are in the feature

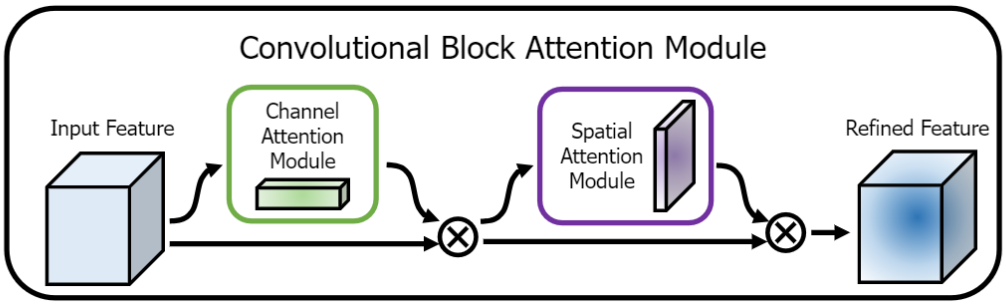


Figure 3.4: Structure of the CBAM[10]

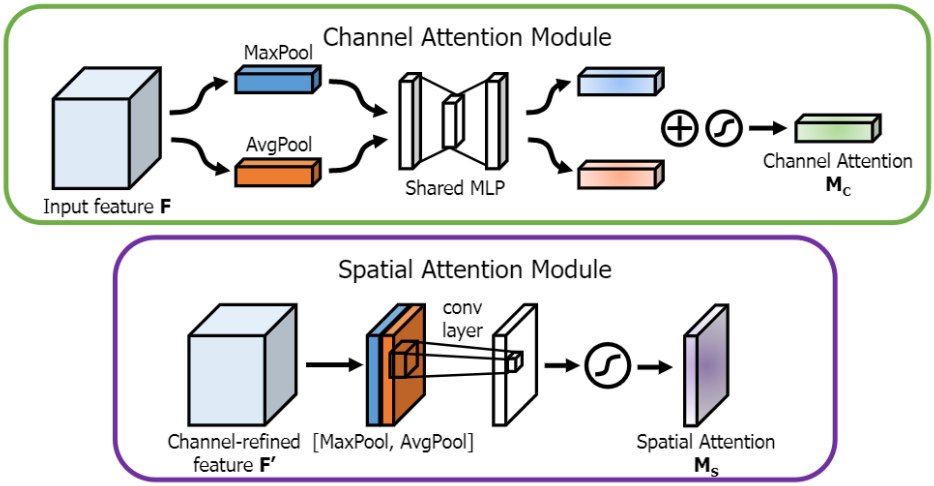


Figure 3.5: The structure of the channel and spatial attention modules[10]

map. The two feature descriptors produced, $F^s_{avg} \in \mathbb{R}^{1\times H\times W}$ and $F^s_{max} \in \mathbb{R}^{1\times H\times W}$ are then passed through a convolution layer to produce the final output spatial attention map:

$$S_z(F) = \sigma(f^{(7\times 7)}(F^s_{avg}; F^s_{max})) \tag{3.4}$$

3.5.2 The Efficient channel attention module

This method is drawn from the channel attention method of SENet [42] which is seen to perform a dimensionality reduction such that inter-channel interactions are decreased. Avoiding dimensionality reduction is seen to be important for effective channel attention learning and thus by using fewer parameters to capture local channel interactions, the ECA guarantees efficiency [43].

A band matrix w_k for learning attention is used for learning channel attention such that

w_k has $k \times C$ parameters with k being an adaptively determined neighbourhood. This implementation avoids the usage of groupings formerly done for learning channel attention and thus avoids the independence of individual groups and encourages interchannel interactions. The weights corresponding to channel y_i can then be calculated using the interaction of y_i and its k neighbors:

$$w_i = \sigma(\sum_{j=1}^{k} w^j y_i^j), y_i \in \omega_i^k \tag{3.5}$$

With ω_i^k being a set of the k adjacent channels of y_i. To increase efficiency, channels are made to share the same learning parameters. The operation in (3.5) can be implemented using a 1D convolution of size k such that:

$$w_i = \sigma(C1D_k(y)) \tag{3.6}$$

Adapative determination of this parameter k would be very costly if done via fine tuning hence a mapping between the k and C is made such that:

$$c = \Phi(k) \tag{3.7}$$

A non-linearity is then imposed on the mapping so that:

$$c = \Phi(k) = 2^{\rho \times k - b} \tag{3.8}$$

The kernel size k can then be adaptively determined by :

$$k = \Psi(C) = |\frac{\log_2(c)}{\rho} + \frac{b}{\rho}|, \tag{3.9}$$

where $|x|_o dd$ indicates the nearest odd number of x and $\rho = 2, b = 1$.

3.5.3 Global attention module

Adopts the sequential structure of the CBAM module and redesigns the channel and attention submodules. This architecture aims to preserve as much information as possible while increasing cross-dimensional feature interactions [44]. The channel attention module employs 3D permutations to preserve information across dimensions. Using a two-layer encoder-decoder type MLP the channel attention module amplifies the cross-dimensional channel-spatial dependencies. The spatial attention submodule then uses two convolutional layers for feature fusion along the spatial dimension. Max pooling is then removed from the spatial attention submodule because it tends to reduce information and negatively affect the network and is replaced with a grouped convolution and a channel shuffle to mitigate against the increase in parameters.

3.5.4 SimAM

This architecture moves from the traditional methods of attention and draws directly from neuroscience knowledge. It is known from the literature that neurons that are firing in the brain show distance patterns or behaviour which can suppress the activity of the surrounding neurons. These are the important neurons to consider when assigning priority for attention. A way that is proposed is to measure the linear separability between a neuron and the surrounding neurons. This is done with the energy function that follows:

$$e_t(w_t, b_t, y, x_i) = (y_i - \hat{t})^2 + \frac{1}{M-1}\sum_{i=1}^{M-1}(y_0 - \hat{x_i})^2 \tag{3.10}$$

with $\hat{t} = w_t t + b_t$ and $\hat{x_i} = w_t x_i + b_t$ as linear transforms of t and x_i and t and x_i as the target neurons in a single channel. i is the index over the spatial dimension and $M = H \times W$ are the number of neurons in a channel. w_t and b_t and the weights and the biases of the transform. The total number of energy functions to be computed would then be equal to the number of neurons per channel which would be computationally expensive to do using normal stochastic gradient descent(SDG). Computing the mean and variance

over all the neurons except the in question neuron t, a closed-form solution is obtained as:

$$w_t = -\frac{2(t-\mu_t)}{(t-\mu_t)^2 + 2\sigma_t^2 + 2\lambda}, \tag{3.11}$$

$$b_t = -\frac{1}{2}(t-\mu_t)w_t \tag{3.12}$$

Where $\mu_t = \frac{1}{M-1}\sum_{i=1}^{M-1} x_i$ and $\sigma_t^2 = \frac{1}{M-1}\sum_{i=1}^{M-1}(x_i - \mu_t)^2$ the mean and variance over the input channel. An assumption is made that, over the channel, all pixels follow the same distribution hence the mean and variant are the same for all and can be reused, reducing computation and as a result, the minimal energy can be computed as :

$$e_t^* = \frac{4(\hat{\sigma}^2 + \lambda)}{(t-\hat{\mu})^2 + 2\hat{\sigma}^2 + 2\lambda} \tag{3.13}$$

where $\hat{\mu}_t = \frac{1}{M}\sum_{i=1}^{M-1} x_i$ and $\hat{\sigma}_t^2 = \frac{1}{M-1}\sum_{i=1}^{M-1}(x_i - \mu_t)^2$. The importance of a neuron can then be computed by using $\frac{1}{e_t^*}$. To do feature refinement using this importance metric, a scaling operation is performed instead of the usual addition such that.

$$F^1 = sigmoid(\frac{1}{E}) \odot F, \tag{3.14}$$

Where E is the group of all e_t^* across the channel and spatial dimensions and F is the input feature map.

3.6 Proposed Framework

In this work, different implementations of attention mechanisms are incorporated into the neck region of the two YOLO models, v8 and v9. This is to enhance the features extracted from the lower layers of the network, making them ripe for use in the detection head parts of the network. The proposed schematic of the proposed system is as in 3.6. The red sections represent the location in which the GAM attention, ECA attention, CBAM attention and simAM attention will be added. Since the attention module performs feature enhancement, these areas should benefit and hence the network should perform better.

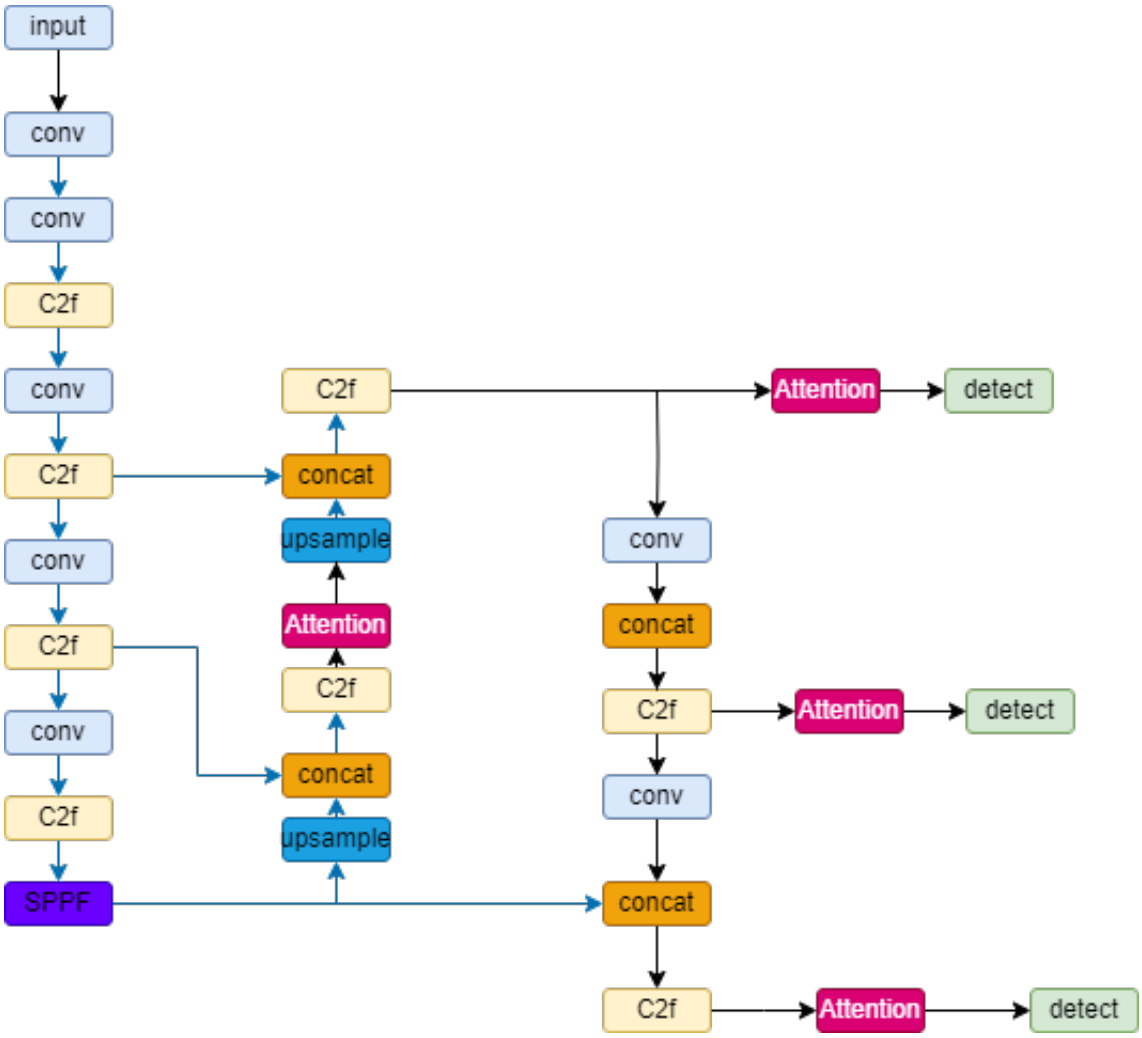


Figure 3.6: Proposed YOLOv8 with Attention Mechanism

3.7 Conclusion

In this section, a comprehensive review of the Yolo algorithms and attention modules was done to get an intuition for the proposed model and why it can be used for the proposed task. From this section, it can be seen just how advanced and complex the latest Yolo models are as compared to the earlier models. It can also be seen from this section how much research is being done on the YOLO algorithm.

Chapter 4

Experimental Results and Discussion

4.1 Introduction

In this section, the experimental setup is explained and the resulting outputs are discussed as outlined in chapter 1. The datasets and Evaluation metrics are also explained in detail.

4.2 Datasets

We utilize two of the most popular publicly available datasets for this research in Kitti[45] and the BDD100k[46] datasets are used for autonomous vehicles. The difference between the two datasets is mainly in the number of images and differing weather conditions found in the BDD100k dataset.

4.2.1 Kitti Dataset

This is one of the most popular datasets which consists of 14999 images with half of them with image annotations. The labelled images comprise 51865 object annotations of which the object distribution can be seen in fig 4.1. We merge the original classes into just six classes down from nine. These images included many different driving and traffic scenarios, driving in the city, highways, and rural areas which gives a good variation in learning real-world driving conditions. For this research, we use the labelled part of the dataset and split that into training, validation and test sets using the usual 7:2:1 ratio. The training split is done so that the model avoids overfitting the training data. Providing a

larger portion of the remaining data for validating makes sure that the network is exposed to a larger sample of "unseen" data during training. The test data will thus, help us evaluate the models after training has been completed.

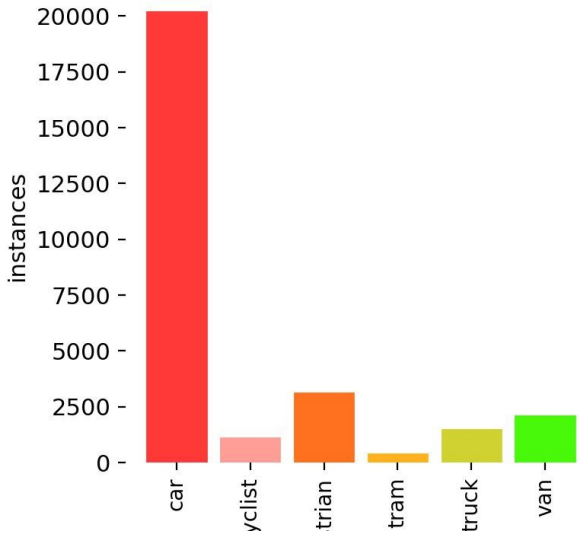


Figure 4.1: Distribution of object classes in the kitti dataset

4.2.2 BDD100k Dataset

This dataset provides 2221128 bounding box annotations across 10 different categories on 100k images sampled from multiple crowd-sourced image sequences. The dataset includes data from different driving scenes including city streets, highways and residential areas across multiple day and night weather conditions. This diversity gives the dataset an advantage over other popular datasets allowing for the evaluation of different domains for object detection. The dataset is split three ways into training(70000), validation(10000) and testing(20000). This data split avoids the network overfitting on the training data providing unseen reserved data for validation and testing of the models. We merge and drop some of the categories from the dataset to form just the four categories (vehicle, pedestrian, traffic sign, traffic light) for road obstacle detection with the distribution of objects in the dataset as seen in figure 4.2.

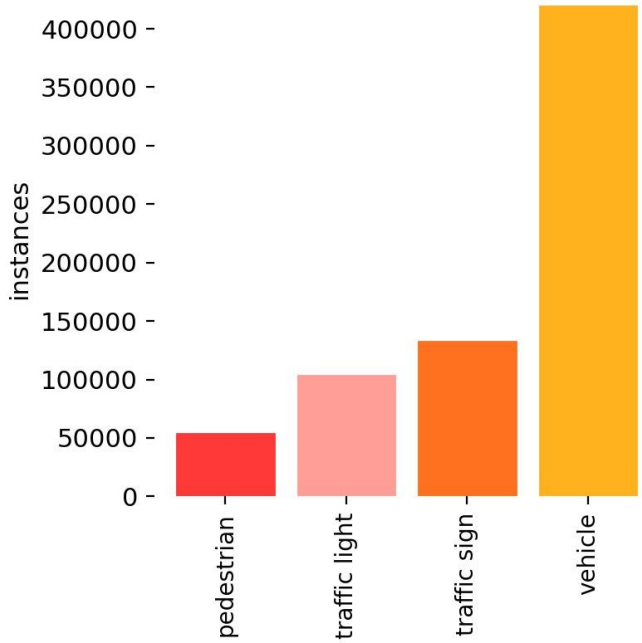


Figure 4.2: Distribution of object classes in the BDD100k dataset

4.3 Evaluation metrics

In this research, mean average precision(mAP) is used as the method to evaluate the model's accuracy. This is the most common method used for evaluating object detection models and provides a very good measure of model performance. The mAP is calculated by using two other matrics which are the precision(P) and recall(R) and is obtained by computing the area under the precision/recall curve.

Precision is defined as the percentage of samples detected as correctly positive. It is calculated by equation 4.1. At the same time, recall is the percentage of samples detected as correctly positive for all the ground-truth samples and is calculated by equation 4.2. The parameters for this calculation are False positives(FP), false negatives(FN), true positives(TP) and true negatives(TN).

True positives are the correct detection, false positives are the samples detected as correct but are not, false negatives are samples detected as false but are correct and true negatives are correctly detected as false.

$$Precision = \frac{TP}{TP + FP} \tag{4.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

The TPs are computed at object bounding box intersection-over-union(IOU) of 0.5. The usage of these two metrics is ideal for object detection as they focus only on positive detection negating the true negatives which may overwhelmingly affect the detection performance as there maybe be infinitely many TN [47].The calculation for average precision(AP) is given by the equation 4.3 which computes the area under the precision/recall curve mAP will then be the averaging of the APs for the different classes given by equation 4.4.

$$AP = \int_0^1 P(R)dR \tag{4.3}$$

$$mAP = \frac{\sum_{i=1}^{M} AP_i}{M} \tag{4.4}$$

4.4 Preprocessing

For training purposes, the images are processed using the HSV augmentation, image scaling, horizontal flip and mosaic augmentation. These methods provide a lot of variation in the training data which allows the models to learn more and be more robust to variations that may be encountered in the real world [48].

HSV augmentation: varying the HSV values simulates different lighting conditions, brightness and colours which allows for enough variation for the network to learn. This could be useful in training the network to produce robust detection results regardless of day or nighttime lighting conditions. This is very important for vehicle obstacle detection.

Image scale allows for the network to learn across varying image scales allowing the model to detect small and large objects. This makes the network robust to changes in image size in the real world. This is useful for the detection of traffic lights, and traffic signs and also to detect obstacles close and far from view.

Horizontal flip, by flipping images horizontally allows the network to detect object from different view points.

Mosaic augmentation Several photos are combined to produce a single training sample that has a mosaic-like appearance as in fig 4.3. This was introduced in the YOLOv4 paper which included images from different contexts in one image [37]. This aids the

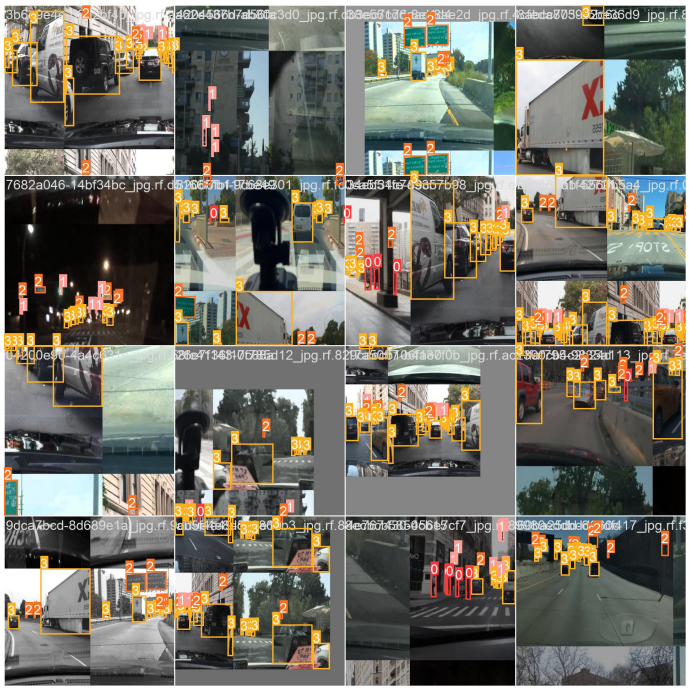


Figure 4.3: Mosaic Augmented images

YOLO model in learning how to identify things in complex scenarios with potentially overlapping objects or a busy environment. The model learns to handle scenarios where items are partially covered or mixed together when it is trained using mosaic-augmented images. Even in difficult situations, the model's accuracy in object detection is enhanced by this augmentation strategy.

4.5 Experimental environment and training parameters

Experimentation was done on a cluster environment of Tesla V100 16GB GPUS accessed individually for training. All code was implemented on pytorch using the ultralytics code base[48].

4.6 Experimental Results on the Kitti dataset

This section presents the results of the different models on the Kitti dataset.

Table 4.1: Experimental Settings and Hyperparameters

Setting	**Hyperparameters**
Epochs	190
Optimizer	SGD with Momentum
Learning Rate	0.01
Mometum	0.937
LR Scheduler	11, 15
LR Decay	0.0005
Weight Decay	0.001
Batch Size	16
NMS: IOU Threshold	0.7
Resolution	640×640
Weight Initialization	from scratch

4.6.1 Comparisons of models across accuracy

In this section, the models are compared across the accuracy achieved. The models are trained on 6 classes of the kitti dataset. We compare using the 4 metrics precision, recall, mAP_{50} and the $mAP50-95$. For the Yolov8 variants, the attention modules were added to the neck region in the sections after the C2f module. There are four attention modules added to the neck section of the yolov8 network. We can see that the GAM attention achieves the best performance increase amongst all the attention implementations of 1%. It can be seen that there is an increment in both precision metrics and mAP_{50} across the models but not so much on the recall metric.

Yolov8 model	Precision	Recall	mAP_{50}	mAP_{50-95}
Original	91.1	86.8	92.3	71.9
ECA	92	86.4	92.8	71.4
simAM	92.8	86.3	92.7	71.3
GAM	**93.6**	82.7	**93.3**	**72.6**
CBAM	92.3	**87.4**	92.7	70.8

Table 4.2: Results of modified YOLOv8 models on the kitti dataset

In table 4.3 the comparison of YOLOv9 models with attention is made. The attention modules are added to the head section of the network to enhance features for detection. The ECA attention network is seen to give a performance enhancement over the baseline network that is better than other networks. It can also be seen that recall and $mAP50-95$ scores across the network have seen a slight increase over the baseline. This shows that addition of attention into the YOLO algorithm does provide feature enhancements that

do provide performance improvements.

Yolov9 model	Precision	Recall	mAP_{50}	mAP_{50-95}
Original	89.7	83.7	89.3	67.8
ECA	**90.2**	83.8	**90.8**	**68.8**
simAM	89.4	83.6	90	68.1
GAM	88.9	84.3	89.4	68.1
CBAM	88.7	**84.5**	89	67.4

Table 4.3: Results of modified YOLOv9 models on the kitti dataset

4.6.2 Comparison of models across Categories

This section compares the performance of models across the different categories. From table 4.4 it can be seen that, across categories, the GAM attention network sees an improvement over multiple categories hence the greater improvement in overall mAP_{50} value. The car category is the most prominent one as with almost if not all autonomous driving datasets having a class imbalance that favours it, it can be seen that all the variants present the same value which has very little room for improvement.

Class	Original	ECA	CBAM	simAM	GAM
Car	98	98	98	98	98
Cyclist	85.4	86.1	85.5	**86.2**	86.3
Pedestrian	85.2	**86.4**	85.6	85.2	86.2
Tram	94.5	95	94.8	94.8	**96.8**
Truck	94	94.3	95.2	94.9	**95.5**
Van	96.9	96.7	97.2	97.1	**97.3**

Table 4.4: Results of modified YOLOv8 models on the kitti dataset categories

In table 4.5, it can be seen that the ECA attention network achieves performance increases on multiple categories that are higher than the other networks over the baseline network.

Class	Original	ECA	CBAM	simAM	GAM
Car	97.7	97.6	97.7	97.5	97.6
Cyclist	81.6	82.3	**82.8**	82	80.9
Pedestrian	79.1	80.3	**80.6**	80.1	79.9
Tram	89	**93.1**	82.7	90.8	87.8
Truck	93.3	**94.4**	93.9	94.2	94.2
Van	95.4	**96.8**	96.1	95.8	95.9

Table 4.5: Results of modified YOLOv9 models on the kitti dataset categories

4.7 Results on the BDD100k dataset

In this section results on the BDD100k dataset are presented. As can be seen from the results of the Kitti dataset above, the YOLOv9 has a lower performance on the task of road obstacle detection than the YOLOv8 algorithm. Experiments on the datasets are done using modified YOLOv8 models of different sizes on the BDD100k dataset.

4.7.1 Comparisons of models across accuracy

In this section, the models are compared across the accuracy achieved. We compare using the 4 metrics precision, recall, mAP_{50} and the $mAP50-95$. Table 4.6 below presents accuracy results on the yolov8 variations on the BDD100k dataset. The results show that there are some performance increments on some of the models such as the nano and small versions of the CBAM and the GAM attention models. This could be because training on an extended dataset such as the BDD100k dataset provides a large enough sample size such that an advanced model such as the YOLOv8 is able to learn as much as possible. The attention models that produce performance increases also have a notable increase in parameter count over the baseline network which could be the reason why they give a performance boost. An increase in parameter count and complexity can also be seen in the increasing accuracy metrics as the model sizes increase though also at the tradeoff in inference speed of the networks. YOLOv9-tiny and small baseline models are trained in the same manner as the YOLOv8 models to give a fair comparison between models that have the same parameter count. It can be seen from the results that, even though the YOLOv9 algorithm is the latest between the two and also draws inspiration from the YOLOv8 algorithm does not perform better. The YOLOv9 does not perform as well as the YOLOv8 algorithm even though it trains twice as long.

4.7.2 Comparison of models across Categories

Performance across object categories is presented and compared between the different models. The results presented in 4.7 for the models across categories show that the GAM attention and CBAM attention modules achieve a performance boost over the baseline more than the other attention mechanisms. This shows that attention has the potential

Yolov8 model	Precision	Recall	mAP_{50}	mAP_{50-95}
Original(n)	68.7	53.6	59	28.7
Original(s)	72.8	58.7	64.7	32.1
Original(l)	75.1	63.9	69.9	35.9
ECA(n)	68.7	53.7	59	28.7
ECA(s)	72.1	58.7	64.6	32.1
ECA(l)	**75.7**	63.9	70	36
simAM(n)	68.2	53.6	58.8	28.6
simAM(s)	72.1	58.4	64.4	32
simAM(l)	75.2	63.9	69.8	35.9
GAM(n)	69.8	54.4	60.3	29.5
GAM(s)	72.1	58.7	64.6	32.1
GAM(l)	75.8	65.1	71.1	36.5
CBAM(n)	69.3	54.4	59.9	29.3
CBAM(s)	72.8	58.9	65	32.4
CBAM(l)	74.8	62.3	68.6	34.9
YOLOv9-t	66.1	49.9	55.6	27.6
YOLO9-s	70.7	56.4	62.6	31.7

Table 4.6: Results of modified YOLOv8n and YOLOV8s models on the BDD 100k dataset

for feature enhancement depending on the dataset used and the type of attention used.

4.8 Comparison with literature

In this section, we compare the results of our algorithms with results found in the literature. The BDD100k dataset provides us with a sizable amount of literature to compare against since it has been widely used for this task. The models are compared on the same metrics as the YOLOP[49] which combines the four classes bus, car, truck, train into a single class "vehicle". Table 4.8 presents the results of the vehicle class alone for comparison. Table 4.9 presents the comparison of proposed models with respect to literature. It can be seen that the results achieved are on par with the results achieved by state-of-the-art models. These models are trained as multitasking models hence the performance they achieve over single-task models like ours may be the result of training strategies that may have an implicit influence on the individual tasks[50]. For this comparison, the best-performing model is used for each model size.

Yolov8 model	Vehicle	Pedestrian	Traffic light	Traffic sign
Original(n)	75.8	51.4	51	57.8
Original(s)	78.9	58.5	57.3	64
Original(l)	81.7	65.4	62.8	69.7
ECA(n)	76	51.1	51.1	57.9
ECA(s)	79	58.2	57.5	64.2
ECA(l)	81.7	65.3	63.2	69.8
simAM(n)	75.8	51.2	50.8	57.5
simAM(s)	78.8	57.9	56.7	64.1
simAM(l)	81.7	65.2	62.8	69.5
GAM(n)	76.6	53.4	52.3	58.8
GAM(s)	78.8	58.5	57.2	64.2
GAM(l)	82.4	66.9	64.1	70.9
CBAM(n)	76.4	53.1	52	58.1
CBAM(s)	79.2	58.8	57.7	64.2
CBAM(l)	81.2	63.5	61.3	65.8
YOLOV9-t	74.9	47.2	45.3	54.8
YOLOV9-s	78.7	55.2	53.3	63.1

Table 4.7: Results of modified YOLOv8n and YOLOV8s models on the BDD 100k dataset

Yolov8 model	Precision	Recall	mAP_{50}	mAP_{50-95}
Original(n)	76.4	69.1	75.8	45.9
Original(s)	80.2	71.2	78.9	48.5
Original(l)	75.1	63.9	69.9	35.9
ECA(n)	76.1	69.3	76	46
ECA(s)	80	71.7	79	48.5
ECA(l)	75.7	63.9	70	36
simAM(n)	76.1	69.3	75.8	45.9
simAM(s)	79.4	71.8	78.8	48.4
simAM(l)	75.2	63.9	69.8	35.9
GAM(n)	77.2	69.8	76.6	46.5
GAM(s)	79.3	72	78.8	48.4
GAM(l)	82.9	75.6	82.4	51.7
CBAM(n)	77.1	69.8	76.4	46.5
CBAM(s)	80	72.1	79.2	48.8
CBAM(l)	74.8	62.3	68.6	34.9
YOLOv9-t	73.6	68.3	74.9	27.6
YOLOv9-s	78	71.4	78.7	49

Table 4.8: Results on the single class vehicle

4.9 Example Visualizations of Attention heatmaps

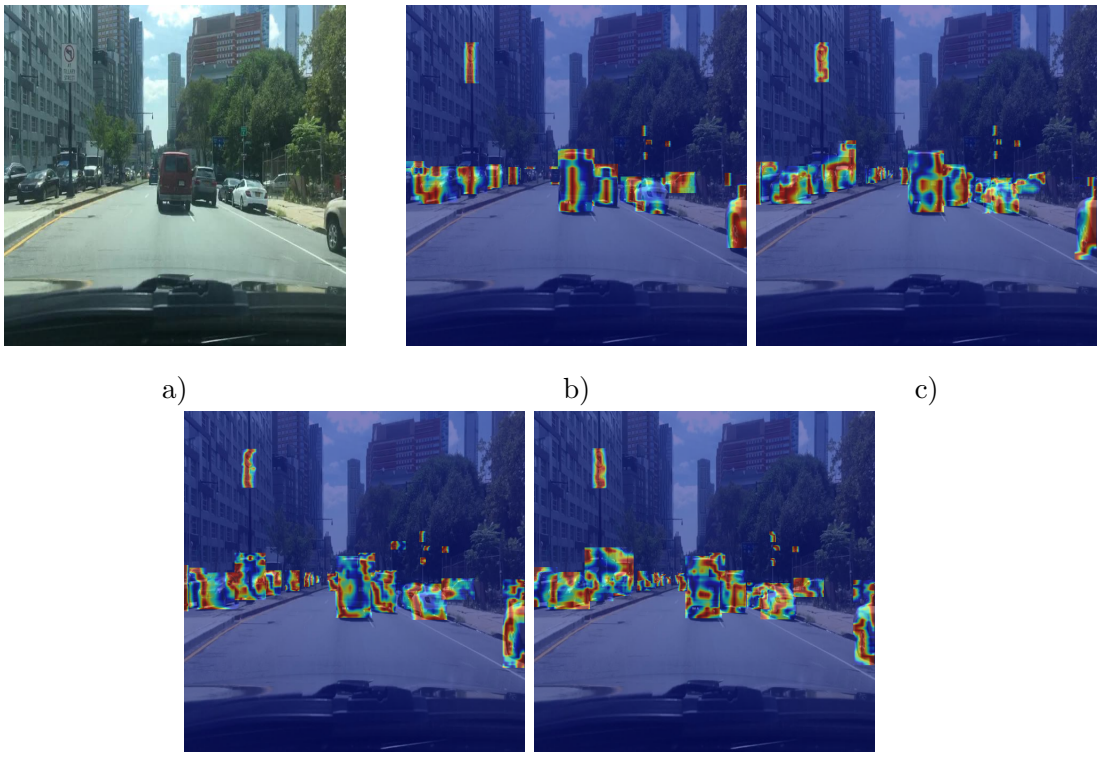
The attention heatmap of the network helps to visualize the outputs of parts of the network and see the different parts of the image that assist in the determination of the final output detection. This also helps to understand the performance of the network

Model model	Recall	mAP_{50}	Parameters
HybridNets	92.8	77.3	12.8M
YOLOP	88.6	76.5	7.9M
YOLOPV2	91.1	83.4	38.9M
YOLOv8(n)	69.1	75.8	3M
A-YOLOM(n)	85.3	78.0	4.43M
A-YOLOM(s)	86.9	81.1	13.61M
YOLOv8(n)-GAM	69.8	76.6	3.69M
YOLOv8-CBAM(s)	72.1	79.2	16.05M
YOLOv8(l)-GAM	75.6	82.9	3.69M

Table 4.9: Comparing models with state of the art models

model and how it operates. A close look at figure 4.4 shows that the attention models do have a big impact in the computed feature maps of the network. In comparison with the original yolov8, though most of the same parts of the image are targeted, the attention networks seem to have accentuated the focus region more and in some instances provided a focus on parts of the image that were not seen by the original network. In images c), d) and e), there's parts that have been isolated on the mid-left of the image that contain instances of vehicles not seen by the original network. d) Also identifies a traffic sign not seen by the other networks

[h]



a) b) c)

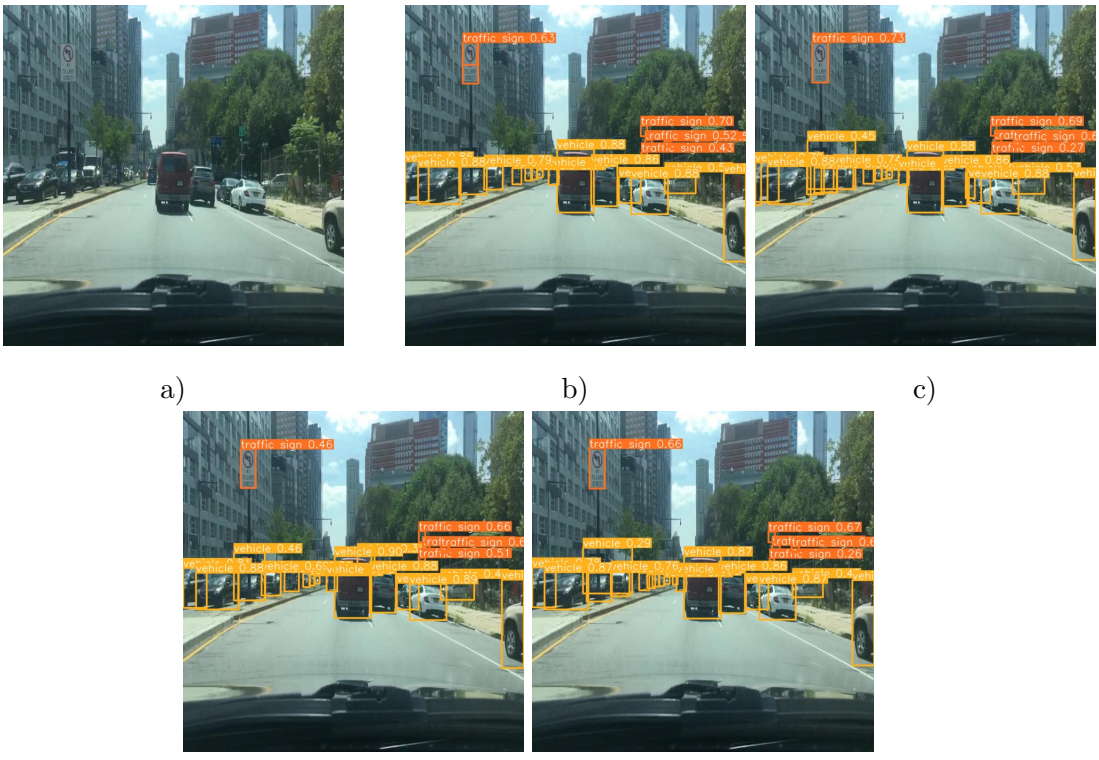
d) e)

Figure 4.4: a)image, b)heat map of original yolov8, c)Heatmap of the GAM attention, d) Heatmap of CBAM attention, e)Heatmap of ECA attention

4.10 Detection comparison of models

Detection results from the different networks are analysed visually in this section. It can be seen from figure 4.5 that the models have varying detection accuracies on the image and the attention models are more or less on par with the original yolov8, but have detected some instances not seen by the original. On the left side of the image, there are vehicle instances that have been picked up by the attention-enhanced models. It can also be seen that the original network sees the street light on the left as two instances whereas there is only one. These results though not overwhelmingly so, show that attention added to the yolov8 can provide some performance enhancements to the network.

[h]



a) b) c)

d) e)

Figure 4.5: a)image, b)yolov8 detections, c)GAM attention detection, d)CBAM attention detections, e)ECA attention detections

4.11 Comparison of detection across different weather conditions

Example prediction results are plotted and visual observations are made on the results of the different models. From these results, it can be seen that even though most of the detections made are similar for all the models, it can be seen that the attention models achieve higher confidence in detection than the original network. It can be seen in figure 4.6 that the GAM attention achieves a confidence score of 90% whereas the original yolo achieves 85%. It can also be observed that the original yolo also seems to miss some instance of vehicle on the left which can be detected by all the other models. In figure 4.7 it can be seen that the models performer in a similar manner and achieve confidence scores that are lower. This shows that, detection in the dark is not as easy as during the day. This is true even for the human driver and the models perform as expected in this case. CBAM seems to be the models the misses instances that have been detected by all the models in this case.

For figure 4.8, the models seem to be performing on par though the original network does detect a traffic sign not detected by the other models. The confidence scores of the models on the different objects is more or less the same. This shows that in the rain, the models perform equally. This could be the result of fewer samples on such conditions than normal daylight samples. It can be seen that the vehicle class is detected at a higher confidence score than the other classes. This could be caused by the area covered by the vehicle class in each image and the general class imbalance of the dataset.

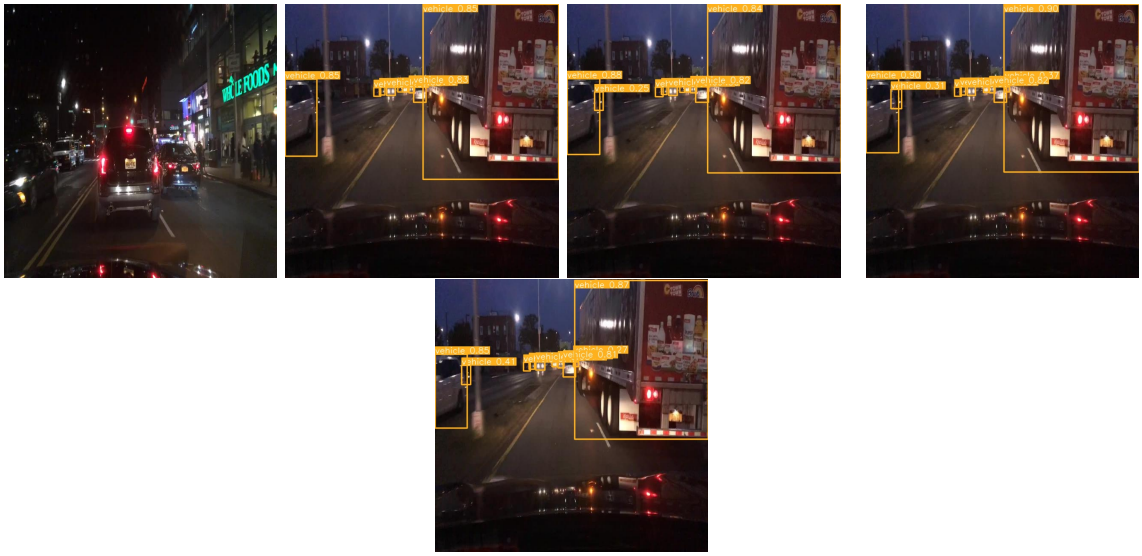


Figure 4.6: Detection results at night frame2:yolov8, frame3:GAM, frame4:CBAM, Frame5:ECA

Figure 4.7: Detection results at night frame2:yolov8, frame3:GAM, frame4:CBAM, Frame5:ECA

Figure 4.8: Detection results in the rain frame2:yolov8, frame3:GAM, frame4:CBAM, Frame5:ECA

4.12 Conclusion

In this section results are presented and comparisons made between models. Models modified with different attention models are compared across the overall accuracy metric and across categories. Observations made show that attention and parameter count do affect the accuracy of models. Visual comparisons done from section 4.9 to section 4.11 show the difference in the detection difference in small objects(traffic lights) and occluded objects(vehicles) between the attention-enhanced models and the original models showing the ability of the attention model to enhance detection capabilities.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Attention Network proved to be effective in improving neural network performances in multiple fields for object detection. This dissertation explores the effectiveness of the fusion of Attention Network with the latest YOLO algorithms for the task of road obstacle detection. This was done by incorporating Attention modules into parts of the YOLO models for the sake of feature enhancement. The networks were trained on two datasets, the Kitti dataset and the BDD100k dataset. From the attention modules that were implemented, the GAM attention module was found to be the best-performing model achieving mAP_{50} of 93.3% on the Kitti dataset and 71.1% on the BDD100k dataset. It was also seen that the YOLOv8 outperforms the YOLOv9 with the same number of parameters. The trained models were then compared with state-of-the-art models and were seen to perform on a level close to them. Visual inspection of the results showed as concluded in section 4.12 that the attention-enhanced models were able to detect instances of occluded vehicles that were not detected by the original network. Some instances of traffic signs were also detected by the enhanced models but not the original. This shows that the attention was able to improve the detection accuracy of the network as expected.

5.2 Limitations of the study

The YOLO algorithm is one of the most popular object detection networks with a huge community of people working on it daily. Even though this is the case, our results show that there are still limitations to what the YOLO algorithm can do.

- The YOLO algorithms seek to balance the tradeoff between real-time inference and high accuracy. This makes it difficult to improve performance while maintaining real-time operation because increasing the number of parameters is what usually improves performance.

- Weather and light conditions can prove to be a problem for object detection algorithms and the YOLO algorithm is not exempt from this. Rain, fog, snow, etc can still be a problem for the algorithm as was seen in the results section.

- Because most of the datasets used for road obstacle detection have the vehicle class overwhelmingly dominating, this skews the network training.

5.3 Recommendations and Future work

- Changing the YOLO backbone to a lighter network designed with attention can allow an improvement in accuracy while also maintaining high inference speed. Investigating techniques to design networks that can produce efficient architectures that compare with the original YOLO backbone accuracy or even better.

- Exploring transfer learning to optimize the model performance. Training on multiple sets of datasets that have varying characteristics on varying domains might bring better insights into the models.in order

- Exploring the usage of synthetic datasets

- Using or developing new evaluation methods that can mainly focus on the detection of small and occluded objects which is still a challenging task for the YOLO algorithm.

Bibliography

[1] B. Mahaur, N. Singh, and K. Mishra, "Road object detection: a comparative study of deep learning-based algorithms," *Multimedia Tools and Applications*, vol. 81, no. 10, pp. 14 247–14 282, 2022.

[2] M. Thoma, "Analysis and optimization of convolutional neural network architectures," *arXiv preprint arXiv:1707.09725*, 2017.

[3] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[4] K. He, X. Zhang, S. Ren, and J. Sun, *Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition*. Springer International Publishing, 2014, p. 346–361. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-10578-9$_2$3

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[6] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," *Advances in neural information processing systems*, vol. 29, 2016.

[7] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, 2016, pp. 21–37.

[8] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[9] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "Yolov9: Learning what you want to learn using programmable gradient information," *arXiv preprint arXiv:2402.13616*, 2024.

[10] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.

[11] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[12] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[13] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 886–893.

[14] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," in *Proceedings. international conference on image processing*, vol. 1. IEEE, 2002, pp. I–I.

[15] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their applications*, vol. 13, no. 4, pp. 18–28, 1998.

[16] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Handwritten digit recognition with a back-propagation network," *Advances in neural information processing systems*, vol. 2, 1989.

[17] S.-i. Amari, "Backpropagation and stochastic gradient descent method," *Neurocomputing*, vol. 5, no. 4-5, pp. 185–196, 1993.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[21] M. Tan and Q. Le, "Efficientnetv2: Smaller models and faster training," in *International conference on machine learning*. PMLR, 2021, pp. 10 096–10 106.

[22] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 116–131.

[23] S. Qian, C. Ning, and Y. Hu, "Mobilenetv3 for image classification," in *2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*. IEEE, 2021, pp. 490–497.

[24] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[25] B. Koonce and B. Koonce, "Resnet 50," *Convolutional neural networks with swift for tensorflow: image recognition and dataset categorization*, pp. 63–72, 2021.

[26] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[27] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016. [Online]. Available: http://arxiv.org/abs/1612.08242

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional

networks for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[29] S. Chaudhari, V. Mithal, G. Polatkan, and R. Ramanath, "An attentive survey of attention models," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 12, no. 5, pp. 1–32, 2021.

[30] Q. Jiang, J. Dai, T. Rui, F. Shao, J. Wang, and G. Lu, "Attention-based cross-modality feature complementation for multispectral pedestrian detection," *IEEE Access*, vol. 10, pp. 53 797–53 809, 2022.

[31] C. B. Murthy, M. F. Hashmi, and A. G. Keskar, "Efficientlitedet: a real-time pedestrian and vehicle detection algorithm," *Machine Vision and Applications*, vol. 33, no. 3, p. 47, 2022.

[32] Y. Chen, J. Wang, Z. Dong, Y. Yang, Q. Luo, and M. Gao, "An attention based yolov5 network for small traffic sign recognition," in *2022 IEEE 31st International Symposium on Industrial Electronics (ISIE)*, 2022, pp. 1158–1164.

[33] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[34] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[35] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[36] ——, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[37] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[38] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8759–8768.

[39] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "Csp-net: A new backbone that can enhance learning capability of cnn," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 390–391.

[40] C.-Y. Wang, H.-Y. M. Liao, and I.-H. Yeh, "Designing network design strategies through gradient path analysis," *arXiv preprint arXiv:2211.04800*, 2022.

[41] A. A. Bastidas and H. Tang, "Channel attention networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2019, pp. 0–0.

[42] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[43] Q. Wang, B. Wu, P. Zhu, P. Li, W. Zuo, and Q. Hu, "Eca-net: Efficient channel attention for deep convolutional neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 534–11 542.

[44] Y. Liu, Z. Shao, and N. Hoffmann, "Global attention mechanism: Retain information to enhance channel-spatial interactions," *arXiv preprint arXiv:2112.05561*, 2021.

[45] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[46] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, T. Darrell *et al.*, "Bdd100k: A diverse driving video database with scalable annotation tooling," *arXiv preprint arXiv:1805.04687*, vol. 2, no. 5, p. 6, 2018.

[47] R. Padilla, S. L. Netto, and E. A. B. da Silva, "A survey on performance metrics for object-detection algorithms," in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2020, pp. 237–242.

[48] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLO," Jan. 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[49] D. Wu, M.-W. Liao, W.-T. Zhang, X.-G. Wang, X. Bai, W.-Q. Cheng, and W.-Y. Liu, "Yolop: You only look once for panoptic driving perception," *Machine Intelligence Research*, vol. 19, no. 6, pp. 550–562, 2022.

[50] J. Wang, Q. J. Wu, and N. Zhang, "You only look at once for real-time and generic multi-task," *IEEE Transactions on Vehicular Technology*, 2024.