Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Mia Fornes

Monocular 3D Object Detection for Volumetric Identification in the Tolling Industry

Master's thesis in Computer Science
Supervisor: Frank Lindseth
Co-supervisor: Gabriel Kiss
June 2023

NTNU
Norwegian University of
Science and Technology

Mia Fornes

Monocular 3D Object Detection for Volumetric Identification in the Tolling Industry

Master's thesis in Computer Science
Supervisor: Frank Lindseth
Co-supervisor: Gabriel Kiss
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

Understanding the volumetric features of vehicles in the tolling domain is crucial for accurate identification and charging. Accurate measurement of these features can be achieved through various methods, such as laser scanners or stereo cameras. These methods are, however, not optimal in terms of cost and often require a lot of data processing. Monocular 3D object detection seeks to identify volumetric features using only a single image, making it a more practical and cost-effective solution.

This thesis investigates this particular field as an initiative from Q-Free, a global provider of tolling, traffic management, and C-ITS solutions. The monocular 3D object detection task is tackled using two different approaches. The first approach includes exploring the performance of monocular 3D object detection using different backbones. Convolutional Neural Networks (CNNs) are a dominant approach in the computer vision field, including monocular 3D object detection. However, with the introduction of the Vision Transformer in 2020, the CNN dominance may be coming to an end. This type of neural network employs self-attention mechanisms to process images and has demonstrated impressive results. Due to limited research on the topic in the monocular 3D object detection field, this thesis aims to investigate the performance of vision transformers compared to CNN-based approaches. The second approach involves utilizing data augmentation techniques, specifically Mixup, which has demonstrated success in various computer vision tasks, including the 3D object detection field. Inspired by previous studies, an enhanced Mixup technique is also implemented, which uses a threshold to determine when the technique should be applied.

The findings of this thesis suggest that CNNs remain superior in terms of feature extraction in the backbone, indicating that there is still room for improvement in the application of vision transformers for monocular 3D object detection. The results also indicate that implementing Mixup techniques may enhance the model's performance, particularly when using vision transformers as their backbone.

Sammendrag

Forståelse av volumetriske egenskaper er avgjørende for nøyaktig identifikasjon og korrekt beslatning av kjøretøy som passerer gjennom en bomstasjon. Presise målinger av disse egenskapene kan oppnås med forskjellige sensorer, eksempelvis laserskannere og stereokameraer. Disse metodene er i imidlertid ikke optimale med tanke på kostnad og behov for krevende databehandling. Monokulær 3D objektdeteksjon er en mer kostnadseffektiv og praktisk løsning, da det kun er behov for ett enkelt bilde for å identifisere volumetriske egenskaper.

Denne oppgaven undersøker nettopp dette feltet som et initiativ fra Q-Free, en global leverandør av løsninger innenfor bomstasjoner, trafikkstyring og C-ITS. To tilnærminger for monokulær 3D objektdeteksjon benyttes i denne oppgaven. Den første tilnærmingen innebærer å utforske ytelsen til monokulær 3D objekteksjonsmodeller ved å benytte forskjellige "backbones". Konvolusjonelle nevrale nettverk (CNN) er en dominerende tilnærming i feltet innenfor kunstig intelligens som omhandler datasyn. I nyere tid har imidlertig en ny tilnærming fått mye oppmerksomhet, nemlig Vision Transformers. Denne typen nevrale nettverk benytter "self-attention"-mekanismer for behandling av bilder, og har vist imponerende resultater. På grunn av begrenset forskning på denne tilnærmingen innenfor det monokulære 3D objektdeteksjonsfeltet, ønsker denne oppgaven å undersøke ytelsen til Vision Transformer-inspirerte metoder sammenlignet med CNN-baserte metoder. Den andre tilnærmingen omhandler bruken av dataøkningsteknikker, nærmere bestem Mixup, som har vist suksess i ulike datasynsopggaver, inkluder 3D objektdeteksjonsfeltet. Inspirert av tidligere studier, er i tillegg en forbedret versjon av Mixup implementert og testet. Denne versjonen drar nytte av en terskelverdi for å bestemme når teknikken skal benyttes.

Funnene i denne oppgaven tyder på at konvolusjonelle nevrale nettverk forblir overlegne som "backbones", og indikerer at det forsatt er forbedringspotensiale for "vision transformers" for monokulær 3D objektdeteksjon. Resultatene indikerer også at bruken av Mixup-teknikker kan forbedre modellens ytelse, spesielt når "vision transformers" brukes som "backbone".

Preface

This thesis was written in collaboration with Q-Free as part of an Msc. in Computer Science at Norwegian University of Science and Technology (NTNU). It investigates the use of monocular 3D object detection for identifying the volumetric features of vehicles, as well as strategies that could potentially improve performance, such as the use of Vision Transformers.

Many people have contributed to the completion of this work and therefore deserve recognition. I would like to thank my supervisor Frank Lindseth and co-supervisor Gabriel Kiss for their guidance and support during the work of this thesis. A special thank goes to Henriette Berg, Truls Gulbrandsen, and Torstein Malvik at Q-Free for their excellent support and contribution to this thesis, providing valuable guidance, expertise, data, and other resources that have made the work easier. An additional thank you goes to those at Q-Free who have shown their interest and support throughout this period. Finally, I would like to thank family and friends for their support during this period.

Mia Fornes
June, 2023

Contents

Figures

Tables

Code Listings

Acronyms

AI Artificial Intelligence. 2

ANN Artificial Neural Network. 4, 5

AP Average Precision. xi, xii, 20, 21, 27–29, 31, 32, 43, 44, 47, 52–54, 57–59, 65, 68, 78–81

C-ITS Cooperative Intelligent Transport Systems. 2

CDIoU Control Distance Intersection over Union. 18

CIoU Complete Intersection over Union. 18, 43

CNN Convolutional Neural Network. iii, 1, 2, 7–9, 13, 15, 20, 23, 26, 28–32, 38, 41–43, 47, 48, 52–54, 58, 62, 65–69

CVIS Cooperative Vehicle Infrastructure Systems. 19

DIoU Distance Intersection over Union. 17, 18

FN False Negative. 20, 48, 66

FP False Positive. 20, 48, 66

FPN Feature Pyramid Network. ix, 15, 16, 23, 27–29, 31, 32, 43

FPS Frames Per Second. 48, 53, 57, 66

GIoU Generalized Intersection over Union. 17

GPU Graphical Processing Unit. 44

IoU Intersection over Union. x, xi, 16–18, 20, 26, 39, 43, 44, 47, 53, 57–62, 64, 65, 68–70

ITS Intelligent Transportation Systems. 19

LiDAR Light Detection and Ranging. 1, 19

MAE Masked Autoencoders. 27

mAP mean Average Precision. xi, xii, 20, 21, 23, 26, 28, 43, 44, 47–49, 52, 53, 57, 58, 61, 62, 65–68, 78–81

MLP Multilayer Perceptron. viii, 5, 14

MSE Mean Squared Error. 16

NTNU Norwegian University of Science and Technology. v, 44

ReLU Rectified Linear Unit. 4, 12

ROI Region of Interest. ix, 20, 35, 37

RQ research question. 2

SLURM Simple Linux Utility for Resource Management. 44

SOTA State of The Art. 13, 26, 29–31

SSD Single Shot Detector. 15

SSH Secure Shell. 44

SVLD-3D Surveillance Vehicle Localization Dataset. 32, 35, 38, 40, 46

TP True Positive. 20, 48, 53, 58, 66

VM Virtual Machine. 44

VSCode Visual Studio Code. 44

YOLO You Only Look Once. 15, 25, 39

Chapter 1

Introduction

1.1 Motivation

Computer vision has made significant progress in recent years, revolutionizing machines' ability to perceive and interpret visual information. 3D object detection is an important challenge with real-world applications that have caught the attention of many researchers. Traditional 3D object detection approaches often involve combining data from multiple sensors, like stereo cameras or Light Detection and Ranging (LiDAR), to determine the necessary depth information for precise detection. However, the application and adoption of these sensors are limited due to the need for complex and expensive sensor configurations.

Monocular 3D object detection, on the other hand, aims to obtain three-dimensional properties of an object using a single image. This area of research has attracted major interest due to its focus on creating easily accessible and cost-effective autonomous systems, including autonomous vehicles, traffic surveillance, and robotics. Accurately determining an object's position, orientation, and size is crucial for the perception system to understand and interact with the surroundings.

Convolutional Neural Networks (CNNs) have become dominant in the computer vision field, primarily due to their ability to extract hierarchical features from input images [1]. This is also evident in the field of monocular 3D object detection, where many models use CNNs as their foundation. In recent years, the Vision Transformer has gained more attention in the computer vision field. The fundamental idea behind the Vision Transformer is to represent an image as a sequence of patches, treating them as tokens analogous to words in natural language. The patches are subsequently fed into a transformer-based architecture that employs self-attention mechanisms to allow the model focus on relevant image features. Using this method allows the Vision Transformer to capture both local and global relationships, which has resulted in improved performance in a variety of computer vision tasks, such as serving as a backbone in 2D object detectors. However, the utilization of Vision Transformers as backbones for monocular 3D object detectors is rather limited.

Motivated by these concepts, this thesis aims to investigate their applicability in the tolling industry. Q-Free is a global provider of Tolling, Traffic Management, and Cooperative Intelligent Transport Systems (C-ITS) solutions. One of their areas of specialization is vehicle detection and characterization based on various sensor systems, with a particular focus on the application of Artificial Intelligence (AI) on images. An area they want to explore further explore is the identification of vehicles and their volumetric features. Today's approach involves using a laser scanner to measure the vehicle's volumetric features, such as length, width, and height. However, to be more efficient in terms of cost and potentially performance, Q-Free wants to explore the possibility of using AI- based technologies to identify the volumetric features from images.

1.2 Goal and Research Questions

The main objective of this thesis is to explore the use of a monocular 3D object detection model for detecting vehicles in Q-Free's gantries, including investigating different approaches to enhance the performance of a monocular 3D object detection model. This includes changing the model by investigating how Vision Transformers can be integrated into existing 3D detection pipelines and their impact on performance, as well as applying techniques during training to increase the model's precision without. The following research questions (RQs) are proposed to address these goals:

RQ1. How accurate is volumetric detection of vehicles using monocular images?

RQ2. How do transformer-based architectures compare to CNNs as feature extractors for monocular 3D object detection?

RQ3. Can training strategies from 2D object detectors be utilized to enhance the precision of monocular 3D object detectors?

1.3 Method

The proposed research questions in this thesis will be addressed through experimentation inspired by a review of relevant literature. The experimental outcomes will be evaluated using quantitative and qualitative methods. The quantitative methods include standard object detection metrics and metrics specifically designed to assess the accuracy of 3D bounding boxes, such as size and localization. These metrics are particularly crucial for precise detection of vehicle volumes.

1.4 Contributions

This thesis makes a valuable contribution to the application of computer vision in the tolling industry. Specifically, this involves examining the potential of a monocular 3D object detection model and exploring various approaches to improve

its accuracy. The monocular 3D object detection area using Vision Transformers has not been thoroughly investigated. Hence, this thesis aims to make a valuable contribution to this field.

1.5 Thesis Outline

Chapter 1 Introduction gives an introduction to the problem and research questions this thesis seeks to address.

Chapter 2 Background and Related Work presents relevant theory central to understanding this thesis, such as an introduction to neural networks, and object detection, as well as presenting related work relevant for this thesis.

Chapter 3 Method describes the process which has been followed in this thesis. A justification of the selected architecture and other configurations are described, as well as an overview of the approach from dataset preprocessing to training and evaluation configurations.

Chapter 4 Experiments and Results presents the results from the different experiments conducted in this thesis.

Chapter 5 Discussion discusses the obtained results with regard to the proposed research questions.

Chapter 6 Conclusion and Further Work concludes the work and identifies potential future research areas.

Chapter 2

Background and Related Work

This chapter gives an introduction to the relevant concepts utilized in this thesis. Certain material elements are derived from the author's previous work [2] and thus contain similarities. However, most of the content has been revised to better align with this thesis.

2.1 Artificial Neural Networks

Artificial Neural Network (ANN) is a subfield of machine learning that draws inspiration from the human brain. ANNs seek to imitate how neurons in the human brain process information [3]. The networks are comprised of three layers: input-, hidden-, and output layer. In its simplest form, an ANN is commonly referred to as a feedforward network. This is due to the unidirectional flow of information, specifically from input to output. See Figure 2.1. Each layer consists of multiple neurons that are connected with a neuron from the previous layer. At the connection point (output/input), equivalent to the synapse in a biological brain, a signal can be transmitted from one node to the other. The neurons usually also have a threshold, i.e. determining whether or not the node should be activated and thus sending the signal to the connected nodes. Additionally, each connection has an associated weight, which will be adjusted during the training process. The weight determines the significance of the input in the output.

$$z = b + \sum_{i}^{n} x_i w_i \tag{2.1}$$

z in Equation (2.1) represents the aggregated sum of a neuron's input. This value is further passed through an activation function f that determines the neuron's (perceptron's) final output. See Equation (2.2). Rectified Linear Unit (ReLU) [4] is a widely used activation function in the hidden layers in a neural network. It is defined as $f(x) = \max(0, x)$, where the function output is 0 for any negative input and the input value itself for any positive input x.

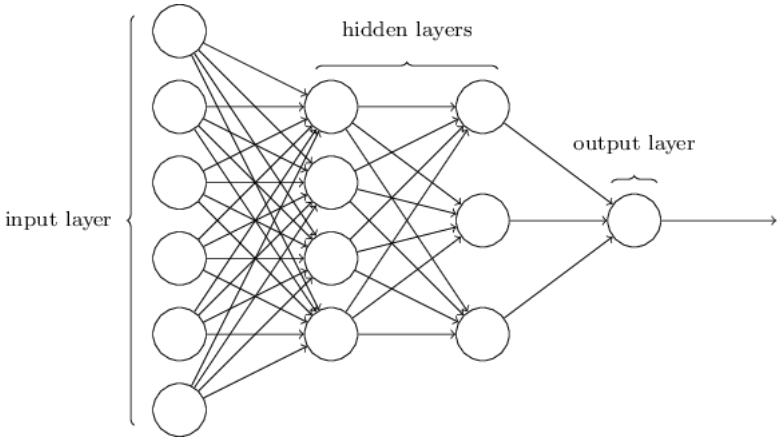


Figure 2.1: Illustration of an Artificial Neural Network (ANN) with and its layers. The design of this neural network is also known as a feedforward network, wherein information is passed on from the input layer to the output layer. These multi-layer networks are commonly referred to as Multilayer Perceptrons (MLPs). Adapted from [5].

$$output = f(z) = f(b + \sum_{i}^{n} x_i w_i) \tag{2.2}$$

An illustration of a single neuron (perceptron) can be seen in Figure 2.2.

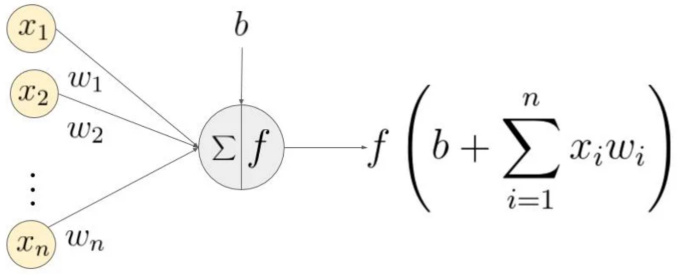


Figure 2.2: Illustration of a neuron (perceptron). Input values x_i, w_i, and b are aggregated and further passed through the activation function f. Adapted from [6].

The weights of an ANN are initialized randomly. The objective of training an ANN is to obtain a set of weights that can accurately describe patterns within the given data. The output of the forward pass, which is the predicted value of the network, will be evaluated against the target value. This is accomplished by utilizing a cost (or loss) function, which serves as a metric for the accuracy of the network. The objective of the training procedure is to determine the weights and biases that minimize the loss.

Upon completion of the loss calculation, the weights are updated and corrected. Backpropagation and gradient descent is essential for this part. The backpropagation algorithm involves the computation of the gradient for each weight in the neural network, which is then propagated backward through the network. The gradient descent algorithm is an optimization technique used to determine the parameters that minimize the loss function.

2.1.1 Regularization

The objective of machine learning is to have an algorithm that performs well on both the training and test data. Regularization is a technique utilized to address this issue, and there exist numerous regularization techniques that can be applied in the context of deep learning. [7]

A common problem with neural networks is when the model has high accuracy during training, but poor performance on the test data. The phenomenon referred to as overfitting is a result of the model's lack of generalization. This implies that the model has become too reliant on the training data and is unable to identify features in the test data. The contrary problem is referred to as underfitting. It occurs when the model fails to learn from the training data, resulting in insufficient performance on the training data. Consequently, the model is unable to generalize on new data, leading to unreliable predictions on the test data.

Weight Decay

Weight decay is a regularization technique that involves applying a parameter norm penalty to improve the generalizability of the model [7]. The technique involves including a penalty to the loss function, usually in the form of the L_2 norm. This approach helps to keep the weights small by encouraging the learning of features less likely to cause overfitting on the training data [8, 9].

Data Augmentation

One effective approach to enhance the generalization of a model is to train it using a larger dataset. Data augmentation is a technique that can be utilized to expand the quantity of data available, particularly when it is limited. This technique expands the dataset by adding modified copies of the original data into the training set [7]. Some common techniques for data augmentation include flipping, scaling, translation, and adjustment of brightness and contrast [10].

Early Stopping

Early stopping is a common form of regularization and, thus, another strategy for avoiding overfitting. This technique involves monitoring the training process and creating a copy of the model every time there is an improvement in the validation

loss, thus saving the best model. The training process is terminated when the metric stops improving, indicating overfitting of the model. In contrast to other regularization strategies, such as weight decay, where the network could get trapped in a bad local minimum, early stopping can be implemented without damaging the learning dynamics. [7].

Transfer Learning

Another technique for preventing overfitting is transfer learning, where knowledge, i.e. weights, from a source task is used to improve learning in the target task. Transfer learning can help the training process converge more quickly and is particularly useful when the dataset is small, making it take longer for the model to overfit. [11].

Fine-tuning is a common approach for transfer learning, where the target model uses a pre-trained model trained on the source tasks to train for further training. Although the source and target tasks may have different problem-solving goals, the former can offer valuable information that the latter can build upon, such as general features like edges, shapes, and textures [12].

2.2 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a type of feed-forward neural network that is especially suitable for computer vision tasks such as image processing. This is due to its ability to extract features/patterns from the data. The key component in a CNN is the convolutional layer [13]. In these layers, a matrix called the kernel slides through the input, multiplying each element in the input matrix with the corresponding position in the kernel. The result from this element-wise product is then saved in an output matrix. This process continues until the whole input matrix has been processed. As a kernel can identify different types of features in the input, the output of this operation will be a matrix with reduced dimensions, wherein the most significant features have been extracted. This matrix is commonly referred to as the feature map.

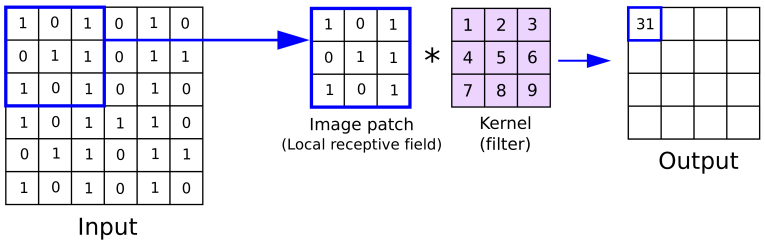


Figure 2.3: An example of the convolution operation. A 3×3 kernel slides through the input matrix, performing an element-wise product between the input and kernel matrix. The result is further summed, and results in a matrix with reduced dimensions called a feature map. Adapted from [14].

Usually, the convolutional layer is followed by a pooling layer that reduces the dimensionality of the convolution layer's feature map. The most common type of pooling utilized is max pool. The pooling operation is executed in a similar way as the convolution operation. Using maxpool as an example, a filter with dimensions of $n \times n$ is applied to the matrix (feature map). The maximum value from each of the receptive fields is then chosen. The filter is associated with a stride, which determines the number of positions (pixels) the kernel will move after each computation. The max pool layer's output is a matrix comprising the most prominent features, as the less significant ones are ignored. Figure 2.4 illustrates an example of the max pool operation.

The last part of a CNN is a fully connected layer. It is essentially a feed-forward network similar to the one shown in Figure 2.1. The final output of either the convolutional or pooling layer is flattened into a one-dimensional vector and further inputted into the feed-forward network. It is responsible for the actual prediction of the network. The output of the final hidden layers undergoes a softmax activation function to produce a probability distribution that determines the class to which an image belongs.

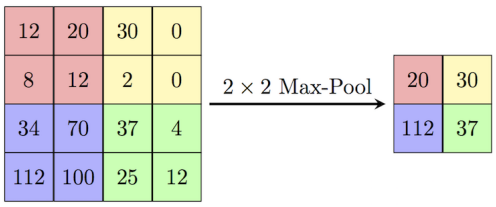
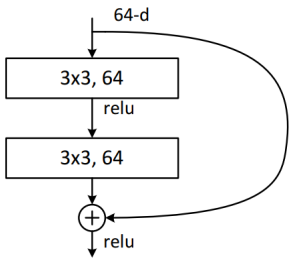


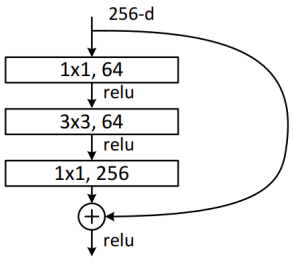
Figure 2.4: An example of the max pool operation using a 2 × 2 kernel and a stride of 2, i.e. the kernel moves two positions to the right after each operation. Adapted from [15].

2.2.1 ResNet

Increasing the depth of CNNs, i.e. creating deeper neural networks, has been a common approach for tackling more complex problems, which also has shown promising results. However, adding more layers to the model has made it difficult to train and has resulted in decreasing performance, which is known as the degradation problem. ResNet [16] was designed to overcome this problem using a residual block, hence the name Residual Neural Network. Illustrations of different residuals block used in ResNet can be seen in Figure 2.5. The residual block makes use of the concept shortcut connections, i.e. skipping one or more layers. In the ResNet residual block, these shortcut connections perform identity mapping. This means that the input gets mapped forward in the network, forcing the deep layers to retain information learned in the early layers of the network.



(a) Basic building block consisting of two convolutional layers. Used in ResNet-18/34



(b) Bottleneck building block consisting of three convolutional layers. Used in ResNet-50/101/152.

Figure 2.5: Illustrations of different residual blocks used in different ResNet models. As seen in each example, the block consists of stacked (sequential) layers and the shortcut connection. An input x goes through both routes, and the output from the stacked layers ($\mathcal{F}(x)$) and the shortcut connection are further added ($\mathcal{F}(x) + x$). Adapted from [16].

2.2.2 ResNext

The introduction of the residual block allowed the ResNet architecture to go much deeper without running into the vanishing gradient problem. Another study [17] also explored widening the ResNet block and found that increasing its width is a more efficient method for improving residual network performance than increasing its depth. However, the ResNext [18] architecture introduced a new dimension C known as cardinality, which was considered to be a more significant factor than the depth and width of the network.

Similar to ResNet, ResNext consists of residual blocks. However, the ResNext residual block follows a split-transform-merge approach inspired by the Inception [19] module. Each block applies a set of transformations, determined by the cardinality C, to a low-dimensional embedding and further aggregates the results. As can be seen in Figure 2.6, the transformation scheme is shared along all

paths, making it flexible for any value of C. With the introduction of cardinality, experiments showed that increasing the cardinality is a more effective strategy for improving the model's capacity than increasing the network's depth or width.

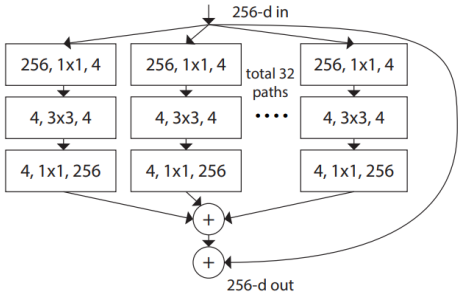


Figure 2.6: Overview of the ResNext residual block. The input is split into C branches, each comprising identical transformations, thus sharing the same topology. The branch transformations are aggregated and multiplied with the shortcut connection, similar to the ResNet block. Adapted from [18].

2.3 Transformers

2.3.1 Transformer

The Transformer [20] was developed in 2017 to address the challenge of sequence transduction, e.g. machine translation. This is accomplished without Recurrent Neural Networks (RNN) or convolutions, relying solely on self-attention. The model complies with the typical structure of other neural sequence transduction models, specifically an encoder-decoder architecture. Figure 2.8 provides an illustration of the architecture.

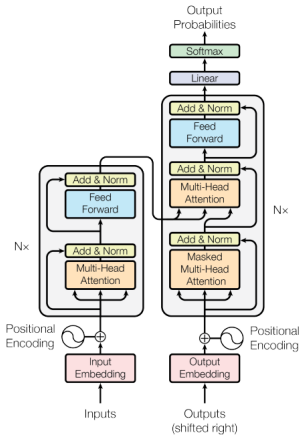


Figure 2.7: The Transformer architecture consists of an Encoder (left) and Decoder (right). Adapted from [20].

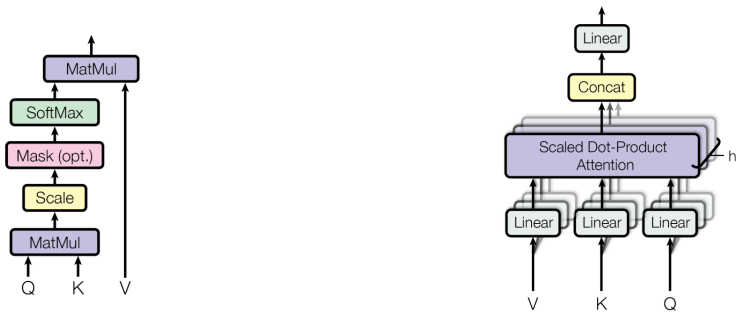
Encoder

Input Embedding and Positional Encoding As previously mentioned, the Transformer model aims to address the challenge of machine translation, which involves processing input data containing words. However, neural networks require a numerical representation of the input data. In order to address this, the encoder utilizes an embedding space in which every word is mapped to a vector representation. Further, the concept of positional encoding is employed, which involves using vectors that provide positional information for each word. This is due to the fact that a word's meaning may differ based on its position within a sentence.

Multi-Head Attention The Transformer employs a specific attention mechanism known as self-attention to effectively capture the relationship between words in the input. The technique used is a type of self-attention called Scaled Dot-Product Attention. This method generates a vectorized representation of the input through query Q, key K, and value V vectors, where Q and K have a dimension of d_k and V a dimension of d_v.

Initially, the dot product between Q and K is computed, which results in an attention score matrix that signifies the relationship between a word in the input sequence and all other words. The attention score is further scaled by dividing it by $\sqrt{d_k}$, which represents the square root of the common dimension of Q and K. In order to derive the attention scores/weights, the scaled matrix is passed on to a softmax function, which transforms the matrix values into probabilities. The attention weights are then multiplied by the V vector. Figure 2.8a shows the visual representation of the scaled dot-product technique, whereas Equation (2.3) presents the corresponding mathematical equation.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.3}$$



(a) Operations of Scale Dot-Product Attention.

(b) Design of the multi-head attention component.

Figure 2.8: Two key components in the Transformer for self-attention. Adapted from [20]

The above description outlines the procedure for one single attention head. However, the Transformer employs a multi-head attention mechanism to compute attention scores or weights. This is achieved by performing the self-attention process in h parallel attention layers, also known as heads. The inputs Q, K, and V are each fed into each of the h heads and processed according to the above-described method. The resulting output from each individual head is concatenated and subsequently multiplied by a parameter matrix to obtain the final attention score.

Feed-Forward Network The second component of the encoder is a position-wise fully connected feed-forward network that operates on a position-wise basis, i.e. a fully connected feed-forward network is created for each word representation. The network consists of two linear transformations, separated by the ReLU activation function. The purpose of this layer is to make the output of a given attention layer match the input of the subsequent attention layer.

Decoder

Input Embedding and Positional Encoding The first step of the decoder is to map the words in the input sequence to a vector representation, similar to the encoder, as previously described. For the decoder, the input sequence is the target sequence.

Masked Multi-Head Attention The initial block of the decoder is a multi-head attention block that includes a masking process, as illustrated in Figure 2.8a. This block is similar to the encoder's first block, however, with some modifications. Prior to applying the softmax function to the scaled matrix, specific values are masked by setting them to $-\infty$. In particular, for a given word located at position i, all words located at positions greater than i are masked out. Thus, the softmax function will assign zero probability to each masked word. By limiting the decoder's self-attention to attend only to positions up to and including a specific position i, the autoregressive property of the decoder is preserved. This is due to the lack of future words during inference, i.e. sentence translation. As a result, the mask is incorporated into the training process to facilitate learning of the input-target data relationship.

Multi-Attention Head The second multi-attention head of the decoder is also referred to as the encoder-decoder attention layer. By combining queries Q from the previous decoder layer with keys K and values V from the output of the last encoder layer, the Transformer can learn the correlation between words in the input and target sequences.

Feed-Forward Network, Linear Layer and Softmax The output generated by the encoder-decoder attention layer undergoes additional processing through the position-wise feed-forward network, following the same procedure as that of the

encoder. The decoder's output is fed into a linear layer which is essentially another feed-forward layer that increases the number of outputs to match the vocabulary size of the target language. The output from the decoder, i.e. the feed-forward network, is then passed through a linear layer which is simply another feed-forward layer that expands the number of outputs to the size of the vocabulary of the target language. The softmax function is applied to produce a probability distribution, wherein the word with the highest probability is considered next in the sequence.

2.3.2 Vision Transformer

The Transformer, as described in Section 2.3.1, is widely used for natural language processing tasks. It has also been utilized in certain computer vision tasks; however, often together with a CNN or replacing specific components of the CNN while preserving the overall structure. Inspired by the Transformer, the Vision Transformer [21] was introduced in 2020 to solve the image classification problem, achieving results comparable to State of The Art (SOTA) CNNs.

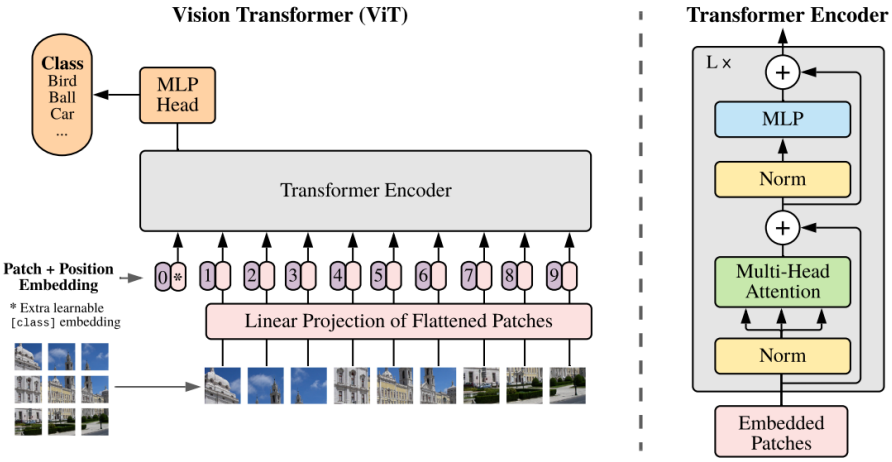


Figure 2.9: Overview of the Vision Transformer architecture. The Transformer Encoder block has the same architecture as the encoder block of the standard Transformer. Adapted from [21].

The main difference between conventional and Vision Transformers is in their input. As described in Section 2.3.1, the Transformer operates with one-dimensional sequences. However, in the case of the Vision Transformer, the input will be a two-dimensional image. In order to address this issue, the image is reshaped into a sequence of flattened 2D patches $x_p \in \mathbb{R}^{N\times(P^2\cdot C)}$. Here, P represents the size of each image patch, C represents the number of channels, and $N = HW/P^2$ is the number of patches, where H and W denote the height and width of the original input image.

Each flattened patch goes through a linear projection layer that generates

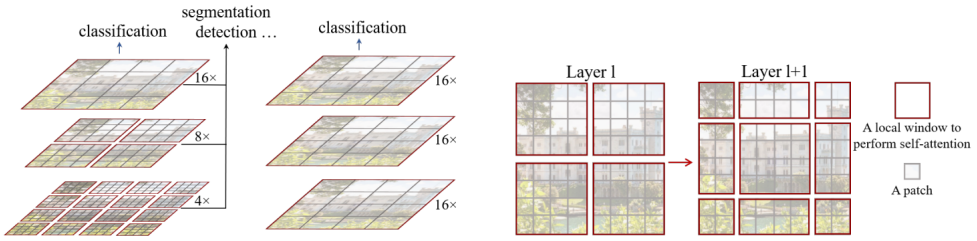
patch embeddings of dimension d. Furthermore, a trainable positional embedding is added into the patch embedding to incorporate positional information into the transformer. Similar to how words can have completely different meanings depending on their order in a sentence, the same is true for images. Prior to being fed to the transformer encoder, the sequence of patch embeddings and position encodings is enhanced with a trainable class embedding that represents the target class, illustrated by the asterisk in Figure 2.9.

The output of the transformer encoder is represented by a vector $C = (C_0, \ldots, C_N)$. For classification, however, only the class token C_0 is used, while the rest of the transformer encoder's output is disregarded. An MLP classification head combined with a softmax function is utilized to create a probability distribution that indicates the class the image represents.

2.3.3 Swin Transformer

The Swin transformer [22] was introduced in 2021 and is a variant of the Vision Transformer. The Swin Transformer employs a hierarchical processing approach and shifted windows approach to overcome the limitations of the vision transformer in managing high-resolution images. The Transformer and Vision Transformer utilize global self-attention, which requires calculating the connections between a token/patch and all other tokens/patches. This results in a quadratic increase in complexity to the number of tokens.

In the Swin Transformer, self-attention is computed within non-overlapping windows using a fixed size, as seen in Figure 2.10b. Limiting the self-attention to only be calculated within the windows would result in a lack of inter-window connections. Thus, a shifted window approach was implemented to facilitate learning of neighboring information between windows. Within these newly proposed windows, self-attention is again computed. The shifted windows approach can be seen in Figure 2.10b. Following this approach, the Swin transformer has a linear computational complexity, making it suitable as a general-purpose backbone for various vision tasks.



(a) Visualization of the hierarchical structure of the Swin Transformer, generating hierarchical feature maps that can be further leveraged by a FPN. The standard Vision Transformer generates a single resolution feature map, as illustrated to the right.

(b) Visualization of the shifted window approach. Following the computation of self-attention on Layer 1, the $M \times M$ windows are shifted by $\frac{M}{2}$ in both the x and y directions.

Figure 2.10: Illustrations of the primary concepts of the Swin Transformer, i.e. the hierarchical structure and shifted window scheme. Adapted from [22].

Inspired by the success of large-scale natural language processing models on language tasks, the authors of [23] sought to further enhance the performance of the Swin transformer by employing large-scale computer vision models. The proposed architecture was named Swin Transformer V2, which scales the Swin Transformer up to 3 billion parameters. The model was evaluated on four vision tasks, namely image classification, semantic segmentation, object detection, and video action recognition. The proposed architecture achieved new benchmark records in most of the experiments conducted.

2.4 Object Detection

Object detection is a field within computer vision that can classify and localize object in an image. The field has gained significant attention in recent years due to the emergence of deep learning, making it one of the most prominent branches of computer vision [24]. The task of generating 2D bounding boxes on images, also known as 2D object detection, has been successful with the implementation of models such as You Only Look Once (YOLO) [25] and Single Shot Detector (SSD) [26]. This technology has been utilized in various industrial products, including security monitoring, transportation surveillance, and robotics [24, 27].

2.4.1 Architecture

An object detection model typically consists of three main parts, i.e. a backbone for feature extraction, a neck that fuses features at different scales obtained by the backbone, and finally, a head that performs classification and bounding box regression. Some object detectors include a fourth stage that produces region proposals between the neck and head. In recent years, CNNs have been the most used

backbone. However, with the introduction of the Transformer, transformer-based approaches have been explored [22, 28, 29].

Feature fusion in the neck is usually achieved by using a FPN [30]. This involves fusing feature maps obtained from different levels, enhancing the model's ability to detect objects at different scales. The main components of the Feature Pyramid Network (FPN) are bottom-up and top-down pathways, and lateral connections, as illustrated in Figure 2.11. The bottom-up pathway refers to a hierarchy of feature maps generated by the backbone at various stages, each stage corresponding to a pyramid level. The top-down pathway constructs high-resolution feature maps by up-sampling low-resolution features that are semantically stronger. The feature maps are further enhanced with features from the bottom-up pathway via lateral connections which combine feature maps of corresponding spatial dimensions from both pathways.

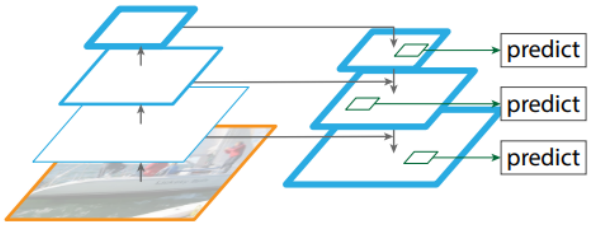


Figure 2.11: Illustration of a FPN with bottom-up and top-down pathways. Adapted from [30].

2.4.2 Bounding Box Regression

Object detection involves accurately localizing objects in an image using bounding boxes. The process of adjusting the predicted bounding box to improve the localization of the targeted object is known as bounding box regression, making it a crucial component in object detection tasks [31]. This includes minimizing the distance between the predicted and ground truth bounding box. There are several approaches for bounding box regression.

The traditional object detectors usually utilized the Mean Squared Error (MSE) to perform regression on bounding boxes on the format $(x_{center}, y_{center}, \text{width}, \text{height})$ or $(x_{min}, y_{min}, x_{max}, y_{max})$. Using the latter described bounding box format, the MSE formula can be defined as in Equation (2.4). Here, gt_i and p_i are the ground truth and predicted coordinates, respectively. A limitation of this approach is that it solely considers individual data points rather than the overall coverage area of the object.

$$L_{MSE} = \frac{1}{4} \sum_{i \in \{^{x_{min}, y_{min},}_{x_{max}, y_{max}}\}}^{4} (gt_i - p_i)^2 \tag{2.4}$$

Researchers proposed the Intersection over Union (IoU) loss to address this

problem. The proposed method involves computing the area of overlap between the predicted bounding box and the ground truth bounding box. This is expressed by the following formula, where B_p and B_{gt} are the predicted and ground truth bounding boxes, respectively.

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}} = \frac{B_p \cap B_{gt}}{B_p \cup B_{gt}} \tag{2.5}$$

A problem with this method is that the IoU is 0 if the ground truth and predicted bounding boxes do not overlap. Thus, the IoU metric has limitations in terms of bounding box regression. If a predicted bounding box gets closer to the actual bounding box without overlapping, the loss function would not reflect any improvement, preventing the network from learning. As a result, improved variants of the IoU metric have been proposed. In the subsequent sections, these techniques will be described in more detail.

GIoU

The Generalized Intersection over Union (GIoU) [32] addresses the non-overlapping cases that the IoU could not handle optimally by moving the predicted bounding box toward the ground truth bounding box in the cases without overlap. It can be described as follows, where B_p and B_{gt} are the prediction and ground truth bounding boxes, respectively, and C is the smallest box enclosing both B_p and B_{gt}:

$$\text{GIoU} = IoU - \frac{|C - (B_p \cup B_{gt})|}{|C|} \tag{2.6}$$

As can be seen from Equation (2.6), the GIoU converges to IoU as the overlap between B_p and B_{gt} increases. Despite being an improved variant of the standard IoU, this approach has some limitations, such as slow convergence.

DIoU

Distance Intersection over Union (DIoU) [33] was proposed to solve the slow convergence problem of GIoU. This was achieved by including a penalty term that minimizes the normalized distance between the center points of two bounding boxes. It is defined as follows:

$$\text{DIoU} = IoU - \frac{\rho^2(b_p, b_{gt})}{c^2} \tag{2.7}$$

where b_p and b_{gt} denote the central points of predicted bounding box B_p and ground truth bounding box B_{gt}, ρ is the Euclidean distance, and c is the length of the diagonal of the smallest box enclosing both B_p and B_{gt}.

Similar to the GIoU, DIoU moves the predicted bounding box closer to the ground truth in cases where there is no overlap. However, due to the penalty term, it converges much more quickly than GIoU.

CIoU

Complete Intersection over Union (CIoU) [33] is an extension of DIoU, proposed by the same authors. The metric includes three geometric parameters: IoU of the predicted and ground truth bounding boxes, the distance between their center points, i.e. DIoU, and the aspect ratio of the predicted and ground truth bounding box. The CIoU penalty term includes the DIoU penalty term and a factor α which considers the aspect ratio. The definition of the CIoU can be seen below:

$$\text{CIoU} = IoU - \frac{\rho^2(b_p, b_{gt})}{c^2} - \alpha v \tag{2.8}$$

The parameter α is a positive trade-off parameter that prioritizes overlaps over non-overlaps cases, and v measures the consistency of the aspect ratio. They are defined as follows:

$$v = \frac{4}{\pi}(arctan\frac{w^{gt}}{h^{gt}} - arctan\frac{w}{h})^2 \quad \alpha = \frac{v}{(1-IoU)+v} \tag{2.9}$$

Using these three geometric properties, the CIoU achieves better performance and faster convergence.

CDIoU

Inspired by the DIoU and CIoU, the Control Distance Intersection over Union (CDIoU) was proposed. CDIoU evaluates the similarity between predicted and ground truth bounding boxes without directly computing their center point distances and aspect ratios. A higher CDIoU value indicates a greater degree of similarity between the objects. The definition of CDIoU is as follows:

$$\text{CDIoU} = IoU + \lambda(1 - diou) = IoU + \lambda(1 - \frac{AE + BF + CG + DH}{4WY}) \tag{2.10}$$

WY is the diagonal of the smallest enclosing box covering both the predicted and ground truth box, similar to DIoU, and λ is a weight. AE, BF, CG, and DH correspond to the difference between the vertices of a ground truth bounding box ($ABCD$) and predicted bounding box $EFGH$.

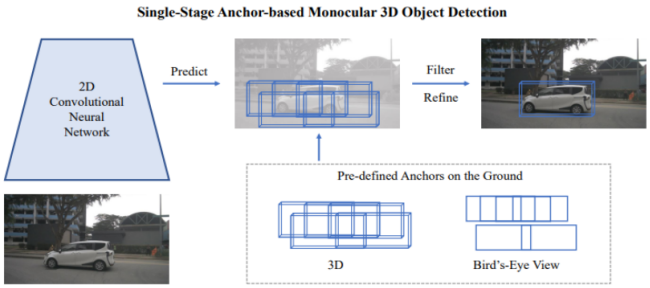
2.4.3 3D Object Detection

In recent years, object detection, specifically 3D object detection, has become an essential part of perception systems for onboard scenes, i.e. autonomous driving [29]. The field of 3D object detection involves accurately identifying and locating objects within the real-world 3D coordinate system. In contrast, 2D object detection primarily focuses on generating 2D bounding boxes within images, without considering the actual distance information of objects from the ego-vehicle [34]. The roadside scene is another environment where 3D object detection has become

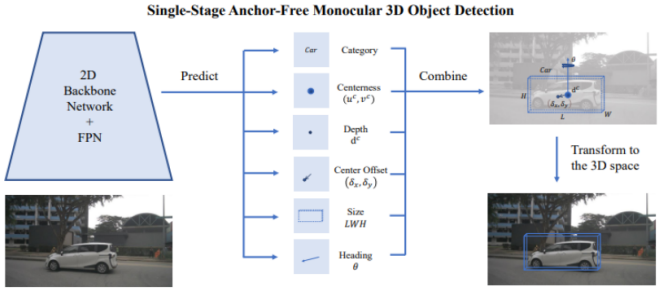
essential, for example, in Intelligent Transportation Systems (ITS) and CVIS for intelligent traffic management [29].

Light Detection and Ranging (LiDAR) and cameras are the most common sensors for solving the 3D localization task. Although cameras are cost-effective and easily accessible, resulting in their widespread implementation, they have certain limitations in the 3D localization task as they are unable to accurately estimate depth. LiDAR sensors measure the reflected information of emitted laser beams, thereby providing more precise 3D data than cameras. However, LiDAR technology is associated with higher costs and greater environmental requirements compared to cameras. [27, 29, 34]

The methodology for detecting 3D objects can be divided into three groups: point cloud based, vision-based, and multi-modal based. The vision-based method can further be divided into monocular camera based and stereo camera based. Monocular camera-based vision involves using a single image to identify objects in 3D space. This presents a significant challenge in predicting an object's 3D location due to the lack of depth information. Stereo camera-based vision can offer more accurate depth information than monocular images by utilizing a pair of images, namely left and right images. Thus, the stereo-based methods generally show greater detection performance compared to monocular-based methods. However, there is still a significant performance gap between stereo-based and point cloud-based methods. [34]



(a) Anchor based.



(b) Anchor free.

Figure 2.12: Overview of the pipeline of the single-stage monocular detection methods. Adapted from [34].

Similar to known 2D detection networks [25, 26, 35], the monocular 3D detection task can be solved by directly obtaining the 3D bounding box parameters from images using a Convolutional Neural Network (CNN). These approaches can be divided into single-stage and two-stage methods, where the former can further be divided into anchor-based or anchor-free methods.

The two-stage methods employ a two-dimensional detector in the initial stage to generate 2D Region of Interests (ROIs). The 3D object parameters are predicted based on the 2D ROIs, and the 2D boxes are further lifted to the 3D space. The single-stage anchor-based approach involves placing a set of 3D anchor boxes within every image pixel. Then, a CNN is employed to generate object parameters from the anchor boxes. See Figure 2.12a. Anchor-free monocular detection methods have the ability to predict 3D object attributes directly from images without the need to locate ROIs beforehand. A two-dimensional CNN is employed for image processing, followed by multiple heads to separately predict the object attributes such as class, center, and size [34]. See Figure 2.12b.

2.5 Metrics

The most common metrics for evaluation of the object detection challenge are Average Precision (AP) and mean Average Precision (mAP). To calculate the AP and mAP, various metrics are utilized, such as Intersection over Union (IoU), True Positive (TP), False Positive (FP), and False Negative (FN), precision and recall.

2.5.1 Intersection over Union

As described in Section 2.4, the IoU metric measures the difference between the predicted bounding box and the ground truth and is a scalar value that ranges from 0 to 1. A prediction is considered more accurate as the IoU value approaches 1. As described in Section 2.4.2, IoU can be used as a bounding box regression technique when training the model. The IoU is, however, also central in other parts of the evaluation of the model. During the evaluation of the model's detection, whether during training or testing, it is possible to establish an IoU threshold to determine the categorization of the detection as TP, FN, or FP. For instance, a threshold of 0.5 can be used. If the IoU between the predicted bounding box and the ground truth falls below the predetermined threshold, the detection will be classified as a FP. Similarly, when the IoU is greater than the specified threshold, the detection is classified as a TP. FN occurs when the model fails to make a prediction despite the presence of a ground truth.

2.5.2 Precision

Precision is a metric that measures the number of correct detections and is defined as the ratio between the true positives and the total number of predictions. It thus

serves as an indication of the model's reliability of the positive predictions, i.e. how often the model predicts correctly.

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{TP}{\text{All Detections}} \tag{2.11}$$

2.5.3 Recall

The recall metric determines the model's ability to correctly identify all relevant instances, i.e. ground truths. In order words, it indicates if the model predicts something every time it should have predicted. The calculation involves dividing the total number of true positives by the total number of ground truths.

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{TP}{\text{All Ground Truths}} \tag{2.12}$$

2.5.4 F1-Score

The F1-score is another evaluation metric for object detection tasks and is defined as the harmonic mean of the precision and recall [36]. See Equation (2.13).

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.13}$$

2.5.5 Average Precision and Mean Average Precision

The precision and recall can further be plotted in a PR-curve. The AP can be found by finding the area under the PR-curve. See Equation (2.14). The PR-curve is obtained by plotting the precision and recall values against the model's confidence score threshold. Due to the characteristics of the two metrics, the curve will be downward-sloping. A lower confidence threshold results in more detections by the model, reducing the likelihood of missed detections and thus resulting in a higher recall. A greater confidence threshold leads to the model's predictions being more precise, resulting in greater precision. [37]

$$AP = \int_0^1 p(r)dr \tag{2.14}$$

The calculation is typically done on a per-class basis. However, if there are multiple classes, the average AP score can be computed [38]. This is referred to as the mAP.

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{2.15}$$

where N is the total number of classes and i represents the ith class.

2.6 Camera Calibration

Camera calibration aims to determine the relationship between 2D points captured by a camera and their corresponding 3D points using intrinsic and extrinsic parameters. The extrinsic matrix denotes the conversion from the global coordinate system to the camera coordinate system and relies on the camera's placement and alignment. The intrinsic matrix is a transformation matrix that maps camera coordinates to pixel coordinates. This mapping is determined by camera properties such as focal length, pixel dimensions, and resolution. [39]

The projection mapping from world coordinates to pixel coordinates can be defined as:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = KRT \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.16}$$

Here, K represents the intrinsic matrix, while RT denotes the extrinsic matrix. The vectors $[u, v, 1]^T$ and $[x, y, z, 1]$ represent the homogeneous coordinates, i.e. a third dimension is added, of the 2D image pixel coordinates and 3D world point coordinates, respectively. The camera matrix, also known as the projection matrix, is defined as $H = KRT$. The matrices in H can be further defined as follows:

$$K = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.17}$$

$$R = \begin{bmatrix} \cos\theta & -\sin\phi & 0 \\ -\sin\phi\sin\theta & -\sin\phi\cos\theta & -\cos\phi \\ \cos\phi\sin\theta & \cos\phi\cos\theta & -\sin\phi \end{bmatrix} \tag{2.18}$$

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -h \end{bmatrix} \tag{2.19}$$

The variables s, f, θ, ϕ, and h in the above equations represent the scale factor, the focal length in pixels, camera pan angle, camera tilt angle, and camera height above the ground, respectively. The values of c_x and c_y correspond to half of the width and height of the image, respectively.

The adjusted world-to-image transformation can be described as follows, where $H = [h_{ij}], i = 1, 2, 3; j = 1, 2, 3, 4$:

$$u = \frac{h_{11}x + h_{12}y + h_{13}z + h_{14}}{h_{31}x + h_{32}y + h_{33}z + h_{34}}, \quad v = \frac{h_{21}x + h_{22}y + h_{23}z + h_{24}}{h_{31}x + h_{32}y + h_{33}z + h_{34}}$$

The adjusted image-to-world transformation can be described as follows:

$$\begin{cases} x = \frac{b_1(h_{22}-h_{32}v)-b_2(h_{12}-h_{32}u)}{(h_{11}-h_{31}u)(h_{22}-h_{32}v)-(h_{12}-h_{32}u)(h_{21}-h_{31}v)} \\ y = \frac{-b_1(h_{21}-h_{31}v)+b_2(h_{21}-h_{31}u)}{(h_{11}-h_{31}u)(h_{22}-h_{32}v)-(h_{12}-h_{32}u)(h_{21}-h_{31}v)} \end{cases} \tag{2.20}$$

where

$$b_1 = u(h_{33}z + h_{34}) - (h_{13}z + h_{14}), \quad b_2 = v(h_{33}z + h_{34}) - (h_{23}z + h_{24})$$

2.7 Related Work

2.7.1 CenterLoc3D

CenterLoc3D [29] is a single-stage, anchor-free object detector focusing on 3D vehicle localization for surveillance cameras in traffic scenes. In contrast, many existing single-stage, anchor-free monocular 3D object detection networks [40–42] target the onboard scene, i.e. for autonomous driving. The methodology comprises two fundamental components, namely camera calibration and 3D localization.

The 50-layer CNN ResNet-50 is used as the backbone, pre-trained on the ImageNet [43] dataset. Utilizing the hierarchical structure of ResNet, the feature maps obtained from the backbone are further used to construct a FPN, followed by a weighted feature-fusion to create a fused feature map. The actual 3D localization happens in the head of the model, using a multi-task detection head responsible for extracting parameters such as vehicle type, centroid, vertices, and dimensions of the 3D bounding box. An overview of the CenterLoc3D architecture can be seen in Figure 2.13.

The model achieved a mAP@0.7 of 51.3% on the car class, outperforming its competitors. However, it is important to note that the benchmark dataset for CenterLoc3D is not the same as its competitors. The evaluation of CenterLoc3D's performance was conducted on their own [44] test dataset, whereas methods like SMOKE [40] and KM3D [45] utilized the KITTI3D [46] dataset for evaluation. In addition, the obtained precision of both localization and dimension was measured. The results show that CenterLoc3D has an average 3D localization precision of 98%, and an average 3D dimension precision of 85%.

The architecture of the CenterLoc3D network can be seen in Figure 2.13

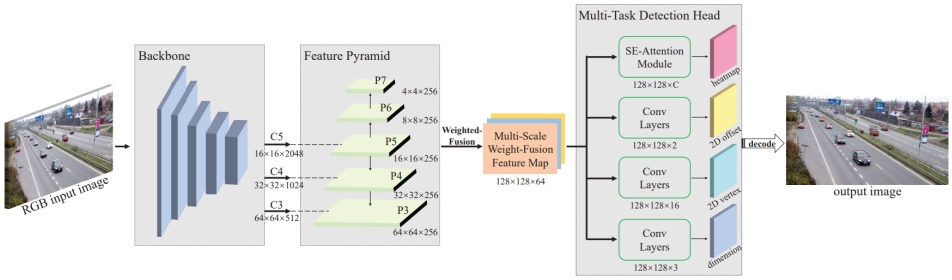


Figure 2.13: The architecture of CenterLoc3D, consisting of a backbone, multi-scale feature fusion, and a multi-task detection head. Adapted from [29].

2.7.2 Data Augmentation

Data augmentation is a widely used regularization method to prevent overfitting. It involves applying image transformations such as flipping, scaling, and rotation, as described in Section 2.1.1. This is attributed to its ease of implementation and effectiveness. This is in contrast to other strategies, which try to increase the model's generalization by improving the model architecture, leading to more complex architectures. Data augmentation, on the other hand, addresses overfitting by going to the root, i.e. the training dataset.

The augmentation techniques mentioned above fall under the category of geometric transformations, which modify the geometric structure of images by shifting image pixels from their original locations to new locations while keeping the pixel value unmodified. Transformations are applied to training data to enhance their ability to represent real-world changes in appearance caused by variations in perspective, scale, and viewpoint. [47]

Another technique is photometric transformation, or pixel-level data augmentation. The aim is to replicate photometric effects by modifying the visual properties of the image, such as brightness, contrast, saturation, and color adjustments. By employing these methods, the deep learning model can become more robust to environmental factors such as variations in weather and time of day, as well as artifacts that may be caused by imaging devices, such as noise and camera configurations. [47]

In the above described methods, data augmentation is applied using a single image. In recent years, there have also been proposed methods for multiple image processing, i.e. image mixing, such as Mosaic [48], Mixup [49], and CutMix [50]. Common methods for generating new image samples include pixel-wise mixing and patch-wise mixing. Pixel-wise mixing involves performing pixel-wise transformations on the entire image content, while patch-wise mixing involves extracting patches from different images and combining them.

Mixup

Mixup [49] belongs to the pixel-wise image mixing, i.e. combining two images into one. In the context of image classification, Mixup is a technique utilized to regularize neural networks by encouraging simple linear behavior. This is achieved by mixing up pixels through interpolations between pairs of images [49]. The implementation of the Mixup strategy demonstrated significant improvements in the model's robustness and generalization capability across diverse tasks, such as image classification, text and acoustic scene classification, and medical image segmentation [51].

It has a relatively straightforward approach:

$$\tilde{x} = \lambda x_i + (1-\lambda)x_j, \quad \tilde{y} = \lambda y_i + (1-\lambda)y_j \tag{2.21}$$

where i and j refer to the original and randomly drawn samples, respectively, and x and y represent the image and bounding box. The λ parameter, which regulates the mixing ratio of the two samples, is usually determined by a $\beta(\alpha, \alpha)$ distribution where $\alpha \in (0, \infty)$. λ determines the mixing ratio between the two samples and usually follows a $\beta(\alpha, \alpha)$ distribution for $\alpha \in (0, \infty)$. This means that a greater value of α leads to a more smoothed distribution causing the mixing ratio to be closer to 0.5. Thus, both images are effectively visible in the combined image. In the case of image classification, x represents the raw input vectors, such as images, while y denotes the one-hot label encodings.

The technique has also been applied and demonstrated improved results in 2D object detection tasks [48, 51, 52]. The simple approach is the same as the one presented in Equation (2.21); however, instead of representing one-hot encodings, y represents the bounding boxes in the two images x. An example is illustrated in Figure 2.14. The main difference in utilizing Mixup in the object detection task is, however, the selection of α. Previous studies have reported conflicting optimal values for α. One study [51] found that $\alpha = 0.2$ gave the best results, whereas another study [52] claimed that the α should be at least one as mixups with $\alpha = 0.2$ were compared with noise with such beta distribution. The latter study found that $\alpha = 1.5$ was marginally better than $\alpha = 1$. However, recent versions of the YOLO object detector have experimented with $\alpha = 8.0$[1] and $\alpha = 32.0$[2].

[1] `https://github.com/WongKinYiu/yolov7/blob/main/utils/datasets.py#L553`

[2] `https://github.com/ultralytics/ultralytics/blob/main/ultralytics/yolo/data/augment.py#L274`

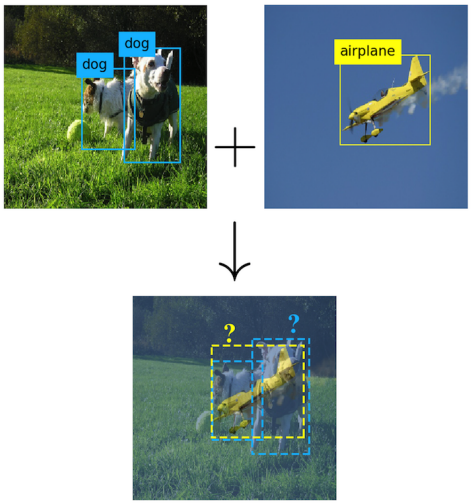


Figure 2.14: Illustration of the Mixup technique in the object detection task. The mixed image $\tilde{x}$ is obtained in the same manner as for the original Mixup, i.e. using a mixing ratio.$\tilde{y}$ is, however, a vector containing all the present bounding boxes from the two samples. Adapted from [51].

The 2D object detection data augmentation techniques have also been tested for 3D object detection [53], showing promising results. The proposed method is a modified version of the original Mixup [49]. Instead of merging two complete images, the original image is augmented with patches from another image, thus providing the same advantages of the standard approach but localized over multiple regions in the image.

The evaluation of this approach was conducted under two distinct conditions. In the initial approach, the patches were directly inserted into the target image. The second approach incorporated an IoU threshold to guarantee that there was no significant overlap between any two boxes in the two images, $IoU(B_i, B_j), \forall B_i \in y_i, B_j \in y_j$. If the calculated IoU between the bounding boxes exceeds a certain threshold, the incoming bounding box will be rejected. Consequently, the Mixup operation will not be performed. The experimental results indicate that the approach utilizing the IoU threshold yields better results in terms of mAP when compared to the approach without any filtering, particularly for higher mAP IoU threshold.

2.7.3 Vision Transformers in Object Detection

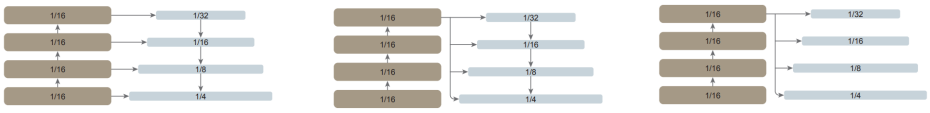
ViTDet

In recent years, different Vision Transformers approaches have also seen increased use in the computer vision field [21, 22], showing results comparable to or better than SOTA CNNs [18, 54].

Commonly used CNN backbones for object detection, e.g. ResNet [16], have

utilized the hierarchical nature of the convolutional network creating a feature map at different stages of the network, leading to them having different resolutions. Using a FPN is widely used in combination with hierarchical backbones as its motivation is to combine high-resolution features obtained in the early stages with stronger features from later stages.

The authors behind ViTDet [55] have explored the utilization of a FPN for plain vision transformers. The vision transformer's structural characteristics make it unsuitable for object detection tasks as it does not naturally provide feature maps at different resolutions, as discussed in Section 2.3.2. Therefore the authors wanted to explore different approaches for converting the single-resolution feature map to different resolutions using a FPN. They proposed and compared three different approaches for generating this FPN. These can be seen in Figure 2.15.



(a) FPN using four stages. **(b)** FPN using the last feature map. **(c)** A simple FPN.

Figure 2.15: Illustration of different approaches for generating a FPN in plain vision transformer approaches. Adapted from [55].

In the first variant, as illustrated in Figure 2.15a, the backbone is artificially divided into four stages to mimic a hierarchical backbone using top-down connections, i.e. creating high-resolution layers from semantically stronger low-resolution maps. The second variant, illustrated in Figure 2.15b, is similar to the first one; however, it uses only the last feature map to construct the FPN. The last variant is referred to as a simple feature pyramid, where only the last feature map from the backbone is utilized. This variant is illustrated in Figure 2.15c. Further, to create multi-scale feature maps of scale $\{\frac{1}{32}, \frac{1}{16}, \frac{1}{8}\frac{1}{4}\}$ a set of convolutions of strides $\{2, 1, \frac{1}{2}, \frac{1}{4}\}$, where a fractional stride indicates a deconvolution.

Experiments using these three approaches as backbones for object detector was performed on the COCO [56] dataset using Mask R-CNN [57] and Cascade Mask R-CNN [58]. The three experiments were done using the three variants proposed in the original Vision Transformer paper [21]: ViT-Base, ViT-Large, and ViT-Huge. For the object detection task, the results were reported using the AP metric.

The first experiment compared the different FPN variants using plain ViT-Base and ViT-Large as backbones and compared them with a baseline with no feature pyramid, i.e. the final layer of the backbone, i.e. the single-scale $\frac{1}{16}$ feature map. The experiments show that all three feature pyramid variants lead to better results than the baseline with both backbones. However, the simple pyramid showing the best performance, increasing the AP by 3.4% for both backbones, i.e. 51.2% and 54.6% for ViT-Base and ViT-Large, respectively.

When pre-training the ViT backbones as Masked Autoencoders (MAE) [59],

ViTDet competes with methods based on hierarchical backbone detectors such as Swin-B [22] for the base ViT for both detection heads Mask R-CNN and Cascade Mask R-CNN. The larger versions, ViT-L and ViT-H, outperform the detectors using a hierarchical backbone.

A study comparing different backbones for object detection on aerial images show similar results [60]. The results from one of the experiments show that the ViTDet outperforms ResNext-101 in most metrics on different datasets, e.g. an improvement of 6-10% on the AP metric. The most extensive experiment of the study compared three backbones, i.e. ViTDet, Swin Transformer, and ResNet, on the DOTA [61] dataset. The results show that ViT-B can compete with hierarchical architectures, both CNNs and Swin Transformers, achieving the best performance with a mAP of 80.89% when using data augmentation.

PVT

Similar to the authors of ViTDet [55], the authors behind the Pyramid Vision Transformer [28], hereby referred to as PVT, wanted to overcome the issue with the standard Vision Transformer, namely that it generated a single resolution feature map, making it unsuitable for dense predictions tasks such as object detection and semantic segmentation. To encounter this, they introduced a pyramid structure into the transformer network, thus being able to generate feature maps at different stages with different scales.

The overall architecture, visualized in Figure 2.16, is divided into four stages, each containing a patch embedding layer and a transformer layer. To generate feature maps at the four different stages, a progressive shrinking strategy is utilized. The input feature map at each stage is divided into $\frac{H_{i-1}W_{i-1}}{P_i^2}$ patches where H_i, W_i, and P_i denotes the height, width, and patch size at the i-th stage respectively. The patches are further flattened and go through a linear projection to obtain embedded patches.

Just like in the standard transformer, positional encoding is added to the input to give the patches' positional information. The combined input is then passed through a Transformer encoder. The transformer used in PVT is, however, slightly modified, i.e. by replacing the traditional multi-head attention layer with a spatial-reduction attention layer which, similarly to the traditional multi-head attention, receives a query Q, key K, and value V and outputs a refined feature. However, in the spatial-reduction layer, the spatial scale of K and V are reduced before the attention operation.

The output from the transformer is then reshaped into a feature map F of size $\frac{H_{i-1}}{P_i} \times \frac{W_{i-1}}{P_i} \times C_i$. Using this approach for all four stages results in feature maps of size $1/4, 1/8, 1/16$, and $1/32$ of the input size, making it suitable for a FPN. Comparing this proposed architecture with well-known CNNs like ResNet and ResNext, the results show that on the object detection task evaluated using the COCO [62] dataset, the PVT surpasses their counterparts in terms of AP.

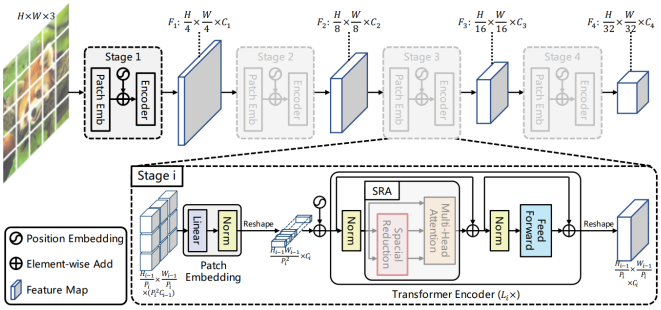


Figure 2.16: Overall architecture of the PVT. Each stage i is comprised of a patch embedding layer and a transformer encoder with L_i layers. Utilizing the progressive shrinking strategy, the PVT obtains multi-scale feature maps suitable for use with a FPN. Adapted from [28].

Swin Transformers

The Swin Transformer has successfully introduced the Transformer to the object detection field. The Swin Transformer, as described in Section 2.3.3, introduced a hierarchical feature representation that effectively optimizes the model for dense prediction tasks, demonstrating noteworthy performance on various dense prediction tasks by implementing the hierarchical structure.

The Swin Transformer demonstrated SOTA performance on the COCO [62] dataset. In comparison to similar sized CNNs like ResNet-50 and ResNext-101, the Swin Transformer demonstrated an improvement of $4.2-5.0\%$ and $3.6-4.5\%$ in AP, respectively, when utilized as the backbone of an object detector with a Cascade Mask R-CNN head. Further, the enhanced Swin Transformer, namely Swin Transformer v2, set new benchmark records, as described in Section 2.3.3. Compared to the first version of Swin, the Swin Transformer v2 saw an increase in AP of 2.1% on the COCO dataset.

Chapter 3

Method

This chapter provides an explanation and justification for the chosen architectures and configurations used in this thesis. The process of dataset preprocessing is also explained, including the collection of images and the generation of annotations. Some parts of this chapter have been adapted from the author's prior work [2].

3.1 Monocular 3D Vehicle Localization

The main focus of this thesis has been to investigate and experiment with different backbones for monocular 3D vehicle localization rather than investigating different approaches for solving the 3D localization problem. However, research done in the author's specialization project [2] concluded that CenterLoc3D [29] would be a good fit for solving this 3D localization problem. The main contributors for this conclusion was it solving the 3D localization task on a similar domain to this thesis, namely on the roadside scene using monocular cameras. Other 3D monocular object detection approaches have been used for solving the autonomous driving problem. An additional advantage was that the model came with a provided dataset comprising images that could be utilized for the project.

This thesis investigates two SOTA approaches for feature extraction, namely using CNN and transformer-inspired architectures as the backbone of the CenterLoc3D [29] network. When choosing the different methods belonging to the two approaches, speed and accuracy were essential factors. The end product should preferably work close to real-time and have good accuracy. Thus, methods that could have a good trade-off were considered.

3.1.1 Choice of Backbones

CNN

With the introduction of the residual block, ResNet allowed creating deeper networks, i.e. increasing the number of layers without increasing the complexity. The

suggested architecture achieved major success by obtaining the top rank in the ImageNet [43] image classification and COCO dataset object detection challenges.

The CenterLoc3D [29] paper employed ResNet and achieved superior results compared to other monocular 3D object detectors. It was thus seen as preferable to use this for this thesis. Originally, the ResNet-50 architecture was utilized, which is a 50-layer deep ResNet network.

However, this thesis utilized the 101-layer deep ResNet version, namely ResNet-101. While the ResNet-50 is faster due to its reduced size compared to ResNet-101, the latter has an increased accuracy [16]. As previously stated, it is preferable for the model to work close to real-time while also having good accuracy. ResNet-101 was thus chosen as the backbone.

Additionally, the improved version of the ResNet architecture was used in this thesis, namely ResNext. By implementing a new dimension, i.e. cardinality, the experiments conducted in the ResNext paper [18], demonstrated that increasing the cardinality effictively enhanced the model's capacity. Despite having a similar complexity to its ResNet counterpart, ResNext demonstrated improved performance. Specifically, the 50-layer network and 101-layer network showed an increase of 2.1% and 0.8% of AP@0.5, respectively. Based on this success, it was decided to use ResNext as a backbone as well. Specifically, the ResNext-101 with a cardinality of 32 was utilized.

Vision Transformer

The standard vision transformer has demonstrated promising results in image classification, with performance on par with SOTA CNNs.

Despite being used for dense prediction tasks such as object detection, the vision transformer is not explicitly designed for this domain as it only produces a single-resolution feature map. As described in Section 2.4.1, object detectors usually comprise a neck to combine feature maps generated by the backbone at different scales. Hierarchical Vision Transformer models, including the Swin Transformer and the Pyramid Vision Transformer, have been proposed to address this issue. As their names imply, they generate hierarchical feature maps that can be used by a FPN, which makes them more applicable to the object detection field. The Swin Transformer v2 demonstrated slightly better performance and was therefore selected as a transformer-based backbone. The small Swin Transformer, namely Swin-S, was chosen to have a fair comparison to the CNN approaches.

Despite being designed to only generate feature maps of a single resolution, the plain Vision Transformers have been explored for their potential in the object detection field. As described in Section 2.7.3, the ViTDet study implemented and compared three methods for creating a feature pyramid from a single-resolution feature map, where two of the approaches included concepts from the traditional FPN. The first mimics a traditional hierarchical backbone by dividing it into multiple stages and utilizing lateral and top-down connections. The second approach utilizes only the final feature map instead of the divided stages, while the third

approach is a basic feature pyramid without a FPN. This approach involves applying parallel convolutions or deconvolutions on the final feature map to generate multi-scale feature maps.

The experimental results indicate that the simple pyramid approach is sufficient for achieving good results on the COCO [62] dataset when compared to hierarchical vision transformers, such as the Swin Transformer. According to the findings, the ViTDet architectures utilizing the simple feature pyramid showed a $0.2\% - 3.2\%$ increase in AP compared to their Swin counterparts when using the Mask R-CNN head. Additionally, a $0 - 2.8\%$ increase in AP when using the Cascade Mask R-CNN head. Due to the intriguing findings in the ViTDet paper, the approach was chosen for this thesis. Similar to when deciding which Swin Transformer network size to choose, the decision was based on having a fair comparison to the CNN-based approaches. Therefore, the ViTDet-B/16 base version was selected. As implied by its name, it utilized a patch size of 16.

3.2 Datasets

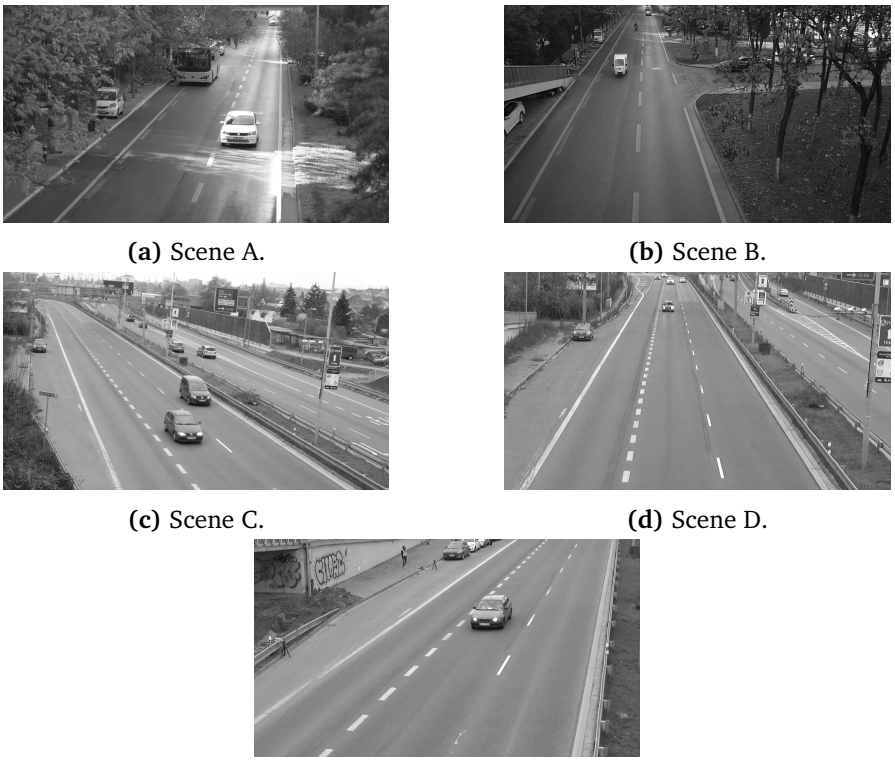
This thesis used two datasets: one provided by the author of CenterLoc3D [29], namely Surveillance Vehicle Localization Dataset (SVLD-3D), and a custom dataset including images provided by Q-Free. The dataset provided was considered more favorable compared to other datasets for 3D object detection. The dataset provided was preferred over KITTI3D [46] due to its similarity to the Q-Free dataset containing images from the roadside scene. The purpose of the provided dataset was to create pre-trained models which would serve as the source task when using the Q-Free dataset for fine-tuning, following the approach described in Section 2.1.1.

3.2.1 SVLD-3D dataset

The proposed dataset SVLD-3D included a combination of images from Sochor et al. [44] and images collected by the author of CenterLoc3D [29]. It was decided to convert the images from RGB to grayscale in order to have more similar images, as the models trained on this dataset would serve as a basis for the custom dataset. The dataset contained a total of 16830 images, of which 14593 were used for training and 2237 were used for testing, including the classes car, truck, and bus. The dataset was created using image sequences from videos taken from surveillance viewpoints of the traffic. Thus, multiple instances of the same vehicle are present in the dataset. The training dataset was initially divided into three separate datasets based on the scene from which they were recorded, with various class distributions. However, it was preferable with a large volume dataset containing all occurrences of the different classes. Thus the separate datasets were merged. Table 3.1 shows the distribution of annotations in the merged training and testing dataset.

Dataset	Class	Annotations
Train	Car	37611
	Bus	1166
	Truck	700
Test	Car	7396
	Bus	745
	Truck	1

Table 3.1: Original dataset distribution.



(a) Scene A. **(b)** Scene B. **(c)** Scene C. **(d)** Scene D. **(e)** Scene E.

Figure 3.1: Example of converted grayscale images from the original dataset from each of the five scenes. Adapted from [2].

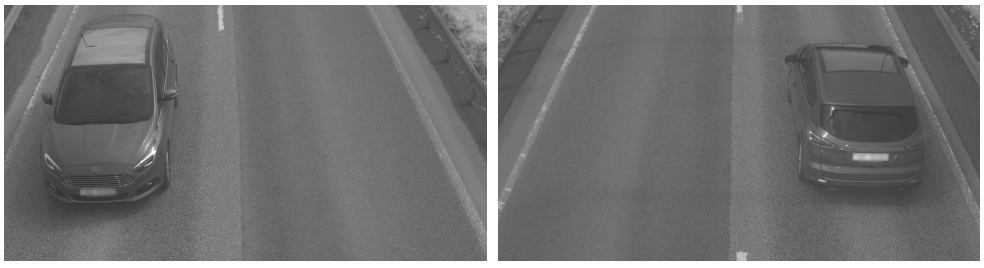
3.2.2 Q-Free dataset

The main dataset used in this thesis comprised 8,484 grayscale images, of which 7,836 were utilized as training images, and 648 were reserved for testing. These images were distributed over the car, truck, and bus classes. The images were gathered from one of Q-Free's sites in Trondheim which provided both front and rear images of the vehicles. The image capture was initiated based on the specific location of the vehicles underneath the gantry, resulting in a relatively homogen-

eous dataset in terms of the vertical positioning of the included vehicles. Due to this triggering mechanism, a specific vehicle for the majority of the cases only appeared once per view (front and rear) in the dataset. An exception for this case was, however, when other vehicles were close to the one triggering the image, resulting in duplicates of the vehicles over two images. The number of instances (annotations) per class for each view is presented in Table 3.2. As can be seen from the distribution, the dataset was somewhat imbalanced, i.e. the number of annotations were not equal for the three classes. The class car had nearly 1000 more samples in total compared to the truck and bus class. The imbalance was also present in the test dataset.

Dataset	Class	View	Annotations	
Train	Car	Front	1799	**3310**
		Rear	1511	
	Truck	Front	1368	**2370**
		Rear	1002	
	Bus	Front	1294	**2280**
		Rear	986	
Test	Car	Front	272	**494**
		Rear	222	
	Truck	Front	70	**86**
		Rear	16	
	Bus	Front	65	**81**
		Rear	16	

Table 3.2: Q-Free dataset distribution.



(a) Front. **(b)** Rear.

Figure 3.2: Example images from the Q-Free dataset.

Because the vehicles in the Q-Free dataset are driving on a multi-lane road, they may appear in both lanes for both perspectives. However, the positioning of the vehicles demonstrated a high degree of uniformity in terms of their placement for both views. From the front perspective, 3823 vehicles were located in the left lane and only 442 in the right lane. The rear perspective demonstrated the opposite trend, due to the rear image being a mirrored representation of the front

view. Here, 3057 vehicles appeared in the right lane and 638 in the left lane.

3.3 Dataset Preprocessing

3.3.1 Dataset Labeling

The provided SVLD-3D dataset included images and a corresponding annotation file containing the necessary information about the objects. The custom dataset, referred to as the Q-Free dataset, had to be manually labeled. This was done using LabelImg3D [63], which was also proposed by the author of CenterLoc3D [29]. To use this annotation tool, a file containing the camera calibration parameters was required in order to translate data from image coordinates to real-world coordinates and vice versa. This file included information such as focal length, tilt and pan angle of the camera, camera height, and vanishing point. An example of the camera calibration file can be seen in Code listing 3.1.

To comply with privacy standards, anonymization of the vehicles in the images was necessary for the Q-Free dataset. This was done prior to annotation. The anonymization involved removing identifiable details like license plates and visible individuals. The licence plate blurring was achieved through a Q-Free library provided for this purpose. After anonymizing the license plates, the dataset was manually reviewed to identify images that contained visible individuals. These images were then excluded.

```
<opencv_storage>
    <f>X</f>
    <fi>X</fi>
    <theta>X</theta>
    <h>X</h>
    <vanishPoints>
      X X X X
    </vanishPoints>
</opencv_storage>
```

Code listing 3.1: Example of the camera calibration file.

The calibration file was originally the only prerequiste for using the annotation tool. The tool was designed to automate the labeling process using a pre-trained YOLOv4 [48] model for generating a 2D bounding box around the detected objects in the image, functioning as a ROI for the 3D bounding box. However, the model failed to detect any object on the majority of the images of the Q-Free dataset.

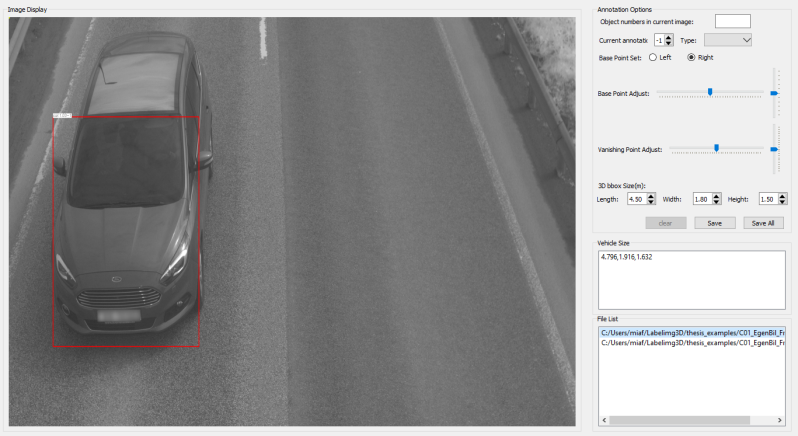
A Q-Free library was employed as a workaround to avoid manually labeling images with 2D bounding boxes prior to the 3D annotation task. Given a folder of images, the library could provide information such as 2D bounding box coordinates, class, and vehicle dimensions. To make use of this information during the annotation process, a script [1] was created to generate "dummy" annotation

[1] https://github.com/miafornes/tdt4900/blob/459f17a4a9d9af4f1da4596ebf9a49b5f5f2014f/dataset_preprocessing.ipynb

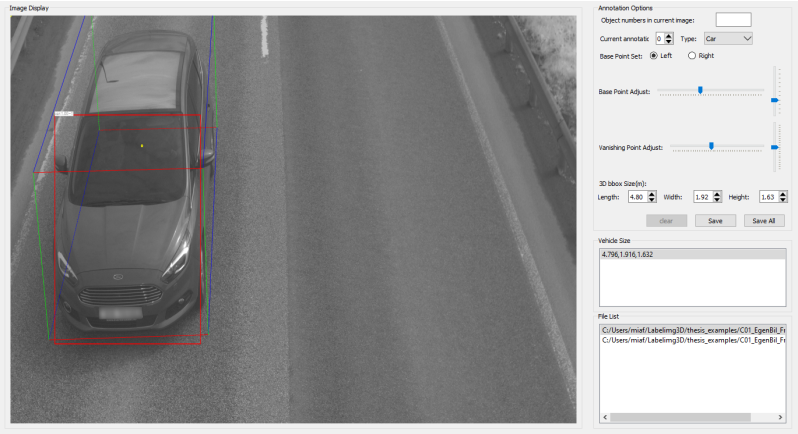
files for each image, containing this information. For the annotation annotation tool to comprehend the annotation file, all its fields had to be filled out. This was solved by making the script set the value of all other fields to 0. An example of the "dummy" annotation file can be seen in Code listing 3.2. As a result, the annotation process only involved making minor adjustments, such as placement of the 3D bounding box with the labeling tool. This led to a significant reduction in the workload associated with the annotation task. Figure 3.3 provides a visual representation of this process, while Figure 3.4 illustrates the dataset preprocessing workflow from beginning to end.

```
<annotation>
    <filename>image_path</filename>
    <calibfile>calibration_file_path</calibfile>
    <size>
        <width>image_width</width>
        <height>image_height</height>
        <depth>num_channels</depth>
    </size>
    <object>
        <type>vehicle_type</type>
        <bbox2d>top left width height</bbox2d>
        <vertex2d>
            (0, 0) (0, 0) (0, 0) (0, 0)
            (0, 0) (0, 0) (0, 0) (0, 0)
        </vertex2d>
        <veh_size>length width height</veh_size>
        <perspective>left</perspective>
        <base_point>0 0</base_point>
        <vertex3d>
            (0, 0, 0) (0, 0, 0) (0, 0, 0) (0, 0, 0)
            (0, 0, 0) (0, 0, 0) (0, 0, 0) (0, 0, 0)
        </vertex3d>
        <veh_loc_2d>0 0</veh_loc_2d>
    </object>
</annotation>
```

Code listing 3.2: Example of the "dummy" annotation file needed in order to use the provided annotation tool. All fields within the object tag, except for those that have been set to zero, are filled with data provided by the Q-Free library.



(a) Before 3D annotation.



(b) After 3D annotation.

Figure 3.3: A snippet of the LabelImg3D annotation tool. The 2D bounding box (red) serves as a ROI for the 3D bounding box. Based on the camera calibration file, the 3D bounding box is drawn and can further be adjusted.

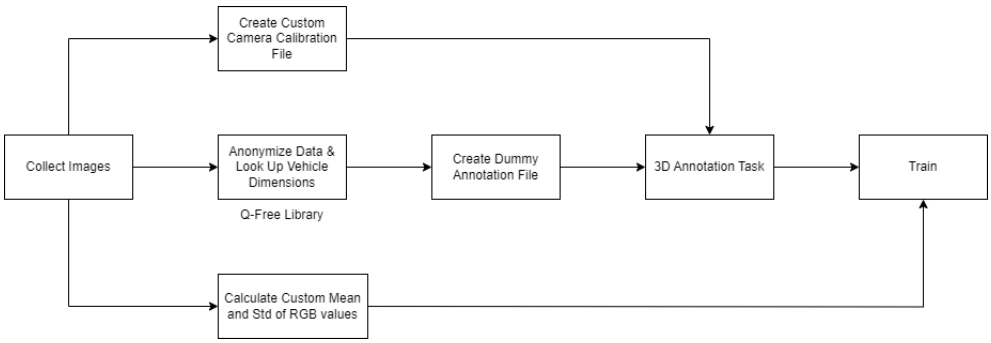


Figure 3.4: Illustration of the overall pipeline for the preprocessing steps for the Q-Free dataset. Adapted from [2].

3.3.2 Normalization and Standardization

During training, the images undergoes the concepts of normalization and standardization. However, this procedure employed parameters optimized for the SVLD-3D dataset. The Q-Free dataset contained grayscale RGB image, resulting in an identical pixel distribution for each color channel, as shown in Figure 3.5. The mean and standard deviation for the RGB channels for the Q-Free dataset are presented in Table 3.3. These values were additionally limited to the range $[0, 1]$, presented in Table 3.4.

	Red	Green	Blue
Mean	57.6058029	57.6058029	57.6058029
Std	15.84698955	15.84698955	15.84698955

Table 3.3: Mean and standard deviation over Q-Free dataset.

	Red	Green	Blue
Mean	0.22590511	0.22590511	0.22590511
Std	0.06214506	0.06214506	0.06214506

Table 3.4: Mean and standard deviation over Q-Free dataset in the range [0, 1].

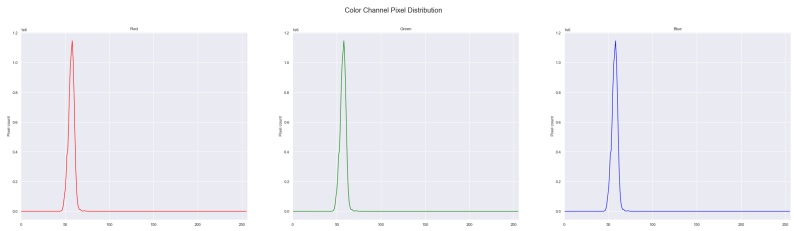


Figure 3.5: Average pixel value distribution in the Q-Free dataset for each channel using a sample of 1000 random images from the dataset. As seen from the distributions, the values are clustered around the same mean, hence the low standard deviation in Table 3.3 and Table 3.4.

3.4 Experimental Setup

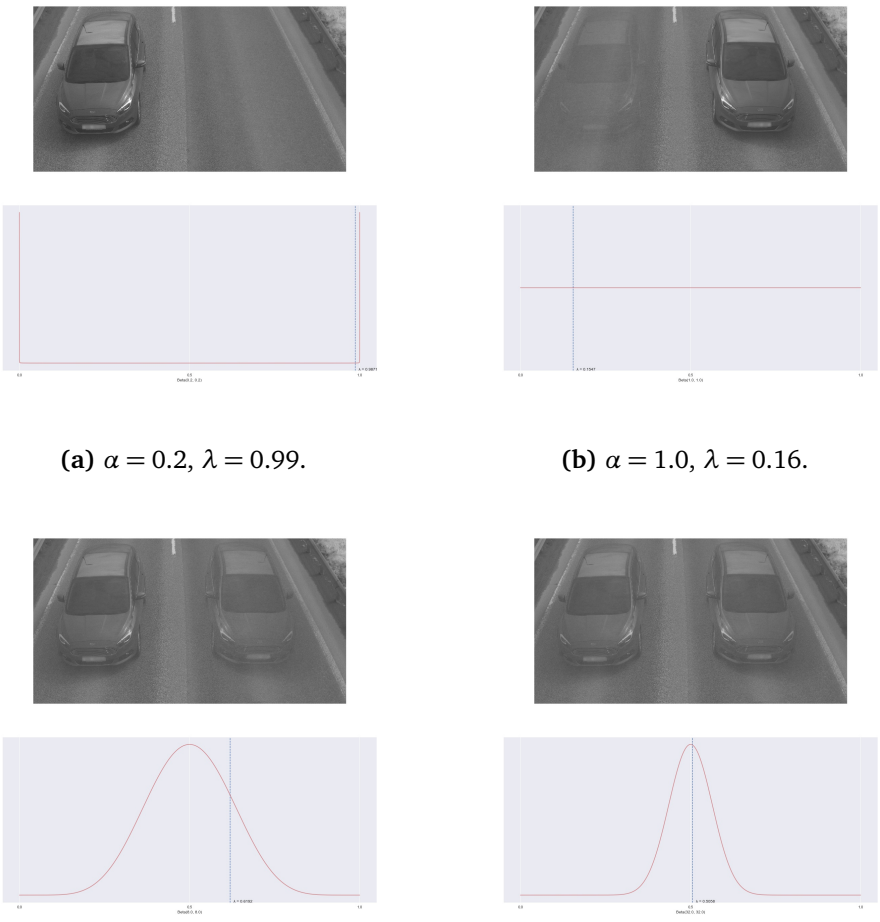
3.4.1 Data Augmentation

In addition to comparing the performance of traditional CNNs and vision transformers as backbones for the monocular 3D object detection task, another experiment in this thesis investigated whether the use of the data augmentation technique Mixup had any effect on the model's precision. Given its success in 2D object detection, it was worth investigating whether this approach would improve 3D object detection.

Two distinct methodologies were employed to implement the technique. The first experiment involved the application of the standard Mixup strategy, which consisted of merging an image with a randomly selected image and combining the bounding boxes from both samples. The second experiment involved utilizing an IoU threshold to determine whether or not an image and the randomly drawn sample should be merged, as described in Section 2.7.2. If the IoU between the bounding boxes in two images exceeded the given threshold, the Mixup was not performed. As the dataset comprised two distinct perspectives, namely front and rear, separate calibration files were necessary for each. To ensure accurate calibration during image mixup, the randomly selected sample was restricted to match the perspective of the original image.

The proposed Mixup method employed an IoU threshold of 0.4. This implies that the Mixup operation was only executed if the IoU value of the bounding boxes from each sample was below this threshold. The threshold value was adapted for this thesis, as it appeared to be a reasonable threshold for the problem at hand. As can be seen from the distribution of the vehicle's positions in the dataset in Section 3.2.2, most vehicles are positioned in the left lane when viewed from the front, and in the right lane in the rear view. Thus, when drawing a random sample from the same perspective, there was a high probability of a large overlap between the samples, resulting in a behavior similar to the original Mixup strategy. However, if the threshold was set too high, the pool of available samples from the same perspective would be rather limited due to the homogeneous distribution of vehicle positions. A high threshold value would also reduce the likelihood of performing the Mixup. It was therefore seen as more convenient to have a threshold that handled both cases, i.e. both ensured that the images were mixed, but also being more selective than the original Mixup.

The mixing of two images involved a mixing ratio λ to determine the strength of each sample in the merged image. The λ value was derived from a beta distribution $\beta(\alpha, \alpha)$ where α determines the shape of the beta distribution. As described in Section 2.7.2, the original mixup used $\alpha = 1.0$ in the image classification problem leading to a uniform beta distribution. This means that λ could be any value on the range $[0, 1]$. In cases where λ approaches values of 0 or 1, the resulting mixed image will contain one of the two samples - either the original image or the randomly selected one. A larger α was thus preferred to increase the likelihood of values closer to 0.5. As described in Section 2.7.2, recent versions of YOLO have utilized α of 8.0 or 32.0. These values result in a mixing ratio λ centered around 0.5, resulting in an equal combination of the two samples. For this thesis, an α value of 8.0 was chosen to achieve a balanced combination of the two images, including some outliers. Figure 3.6 depicts a visualization of the mixing process between two images using varying values of α.



(a) $\alpha = 0.2$, $\lambda = 0.99$.

(b) $\alpha = 1.0$, $\lambda = 0.16$.

(c) $\alpha = 8.0$, $\lambda = 0.62$.

(d) $\alpha = 32.0$, $\lambda = 0.51$.

Figure 3.6: Examples of different beta distributions $\beta(\alpha, \alpha)$ for varying alpha α values resulting in different Mixup ratios lambda λ, along with their corresponding Mixup images. As the value of λ approaches 0.5, the two samples have equal visibility.

3.4.2 Training

Pre-Training

The Q-Free dataset has been the primary dataset for this thesis. However, all experiments on the Q-Free dataset followed the concept of fine-tuning, i.e. transferring knowledge from a source task to a target task. A model trained on the provided SVLD-3D dataset represented the source task, using its best weights. As this thesis examined the utilization of various backbones for feature extraction, a distinct model was developed for each of these backbones, namely ResNet-101,

ResNext-101, ViTDet-B/16, and Swin-S. Utilizing a pre-trained model was considered advantageous for the vision transformer-based models like ViTDet-B/16 and Swin-S due to their need for a substantial amount of data to generalize on unseen data and avoid overfitting. Except for the backbone replacement, the training configurations outlined in the original paper were mostly preserved. However, the split ratio for dividing the training dataset into training and validation data was changed to 1/8 from the initial ratio of 1/9. This resulted in a distribution close to the typical 80-10-10 split for the training, validation, and test sets.

Training Configurations

Hyperparameters The Q-Free dataset experiments relied on training hyperparameters that were primarily based on those provided in the CenterLoc3D paper [29] and were, therefore, comparable to those used for pre-trained models. The model had an input size of $512 \times 512 \times 3$ ($W \times H \times C$), which was found reasonable for this task as well, serving as a good trade-off between precision and speed of the network. Further, Adam [64] was utilized as the optimizer with a weight decay of $5e-4$, which seeks to adjust the attributes such as weights and learning rate to minimize the model's loss. A learning rate scheduler was employed with the Adam optimizer to improve the model's learning rate during training. The scheduler reduced the learning rate by a factor of 10 if the validation loss did not decrease in 3 consecutive epochs. To ensure a fair comparison between CNN-based and transformer-based approaches in this thesis, the same hyperparameters were used for both methods. However, there were certain differences that will be described in more detail.

Name	Input Size	Epochs	Batch Size	Learning Rate	Early Stopping	Weight Decay	Optimizer	BBox Reg.	Data Aug.	
									Mixup	MixupIoU
ResNet-101$_{Baseline}$										
ResNet-101$_{Mixup}$	512 × 512 × 3	200	8	1e-4	12	5e-4	Adam	Ciou	✓	
ResNet-101$_{MixupIoU}$										✓

Table 3.5: ResNet-101 training configurations.

Name	Input Size	Epochs	Batch Size	Learning Rate	Early Stopping	Weight Decay	Optimizer	BBox Reg.	Data Aug.	
									Mixup	MixupIoU
ResNext-101$_{Baseline}$										
ResNext-101$_{Mixup}$	512 × 512 × 3	200	8	1e-4	12	5e-4	Adam	Ciou	✓	
ResNext-101$_{MixupIoU}$										✓

Table 3.6: ResNext-101 training configurations.

Name	Input Size	Epochs	Batch Size	Learning Rate	Early Stopping	Weight Decay	Optimizer	BBox Reg.	Data Aug.	
									Mixup	MixupIoU
ViTDet-B/16$_{Baseline}$										
ViTDet-B/16$_{Mixup}$	512 × 512 × 3	300	8	1e-4	20	5e-4	Adam	Ciou	✓	
ViTDet-B/16$_{MixupIoU}$										✓

Table 3.7: ViTDet-B/16 training configurations.

Name	Input Size	Epochs	Batch Size	Learning Rate	Early Stopping	Weight Decay	Optimizer	BBox Reg.	Data Aug.	
									Mixup	MixupIoU
Swin-S$_{Baseline}$										
Swin-S$_{Mixup}$	512 × 512 × 3	300	8	1e-4	20	5e-4	Adam	Ciou	✓	
Swin-S$_{MixupIoU}$										✓

Table 3.8: Swin-S training configurations.

Tables 3.5 to 3.8 include the training configurations used for the different models. As previously stated, most of the original paper's hyperparameters, such as input size, batch size, weight decay, and optimizer, were adopted. However, the number of epochs and early stopping patience were altered for this thesis.

The CNN-based methods, ResNet-101 and ResNext-101, were trained for 200 epochs with an early stopping patience of 12. Preliminary experiments revealed that the models ended the training process too early with the initially proposed early stopping patience value of 7. It was observed that the models continued to converge in subsequent epochs as the value was increased. Increasing the patience to 12 gave the models a chance to train longer before eventually being stopped due to it starting to overfit. The early stopping patience was further increased for the transformer-based approaches. This was based on the same evidence as the CNN-based models. Transformer-based methods exhibited slower convergence than the the CNN-based methods, necessitating an increase in the early stopping patience.

This also applied to the number of epochs. Due to the slower convergence of the transformer-based approaches, the number of epochs was set to 300, compared to 200 for the CNN-based approaches. However, it is essential to note that the number of epochs does not necessarily indicate the actual number of epochs for which the models were trained. The nature of the CenterLoc3D [29] code affects training configurations, such as the number of epochs and learning rate, when using a pre-trained model as a starting point. The epoch parameter refers to the maximum number of epochs the model should train for; however, instead of starting at 0, the count continues from the epoch of the pre-trained model. Training a model for 200 epochs with a pre-trained model with the final checkpoint at epoch 90 would require 110 training epochs for the target task. For this thesis, all pre-trained models stopped training around epoch 100. This means that CNN-based models with an epoch configuration of 200 would effectively train for 100 epochs, whereas transformer-based methods with 300 epochs would train for about 200 epochs unless the early stopping strategy terminated the training process earlier to prevent overfitting.

The same approach was employed when determining the appropriate learning rate for the target task. The final learning rate used in the target varied due to the implementation of a learning rate scheduler that dynamically adjusted the learning rate during the training process. As previously mentioned, it was preferable to compare the methods on a consistent basis. The learning rate for all models was thus set to $1e-4$, consistent with the unfreeze learning rate utilized in the original paper.

Loss The model employs a multi-task head, resulting in a multi-task loss function that comprising various components. One of its components is the IoU loss, which serves as the bounding box regression. As outlined in Section 2.4.2, there are several bounding box regression strategies. In the early stages of this thesis, the bounding box regression methods were tested and compared. The findings demonstrated that the CIoU approach exhibited superior performance compared to other regression strategies in terms of both standard object detection metrics like AP and mAP, as well as size and localization precision. Although the experiments were conducted on a smaller dataset, the promising findings led to adopting the CIoU bounding box regression strategy for all four models. Appendix B includes the bounding box regression results.

Feature Pyramid

The CenterLoc3D [29] model combines feature maps obtained from the backbone and generates a fused feature map, as explained in Section 2.7.1. CNN models are often designed to generate feature maps at different stages, which can be effectively employed in a FPN. The standard vision transformer does, however, generate a single-resolution feature map, making it unsuitable for the pyramid architecture. However, the transformer-based architectures employed in this thesis have tried to mimic the hierarchical structure. In order to ensure a fair comparison between the various approaches, it was decided that the feature maps to be fused would be of similar dimensions. Each of the four approaches generated feature maps C_3, C_4, and C_5 with sizes that were $1/8$, $1/16$, and $1/32$ of the input size, respectively. For an input size of 512, the corresponding sizes were 64, 32, and 16, respectively.

However, the number of channels varied for the different feature maps. For the CNN-based approaches, the number of channels was 512, 1024, and 2048 for C_3, C_4, and C_5, respectively. The ViTDet architecture used a simple pyramid structure to produce feature maps of varying sizes, such as those mentioned above. However, the number of channels remained consistent across all three feature maps. C_3, C_4, and C_5 all consisted of 768 channels. In the case of the vision transformer, this refers to the embedding dimension, which corresponds to the dimension of the vector representation of a patch, as described in Section 2.3.2. For the small Swin Transformer version, namely Swin-S, the last three feature maps consisted of 192, 384, and 768 channels, respectively, referring to the embedding dimension similar to the ViTDet-B/16. Table 3.9 provides a summary of the information mentioned above.

Backbone	C_3 Size	C_4 Size	C_5 Size
ResNet-101	$64 \times 64 \times 512$	$32 \times 32 \times 1024$	$16 \times 16 \times 2048$
ResNext-101	$64 \times 64 \times 512$	$32 \times 32 \times 1024$	$16 \times 16 \times 2048$
ViTDet-B/16	$64 \times 64 \times 768$	$32 \times 32 \times 768$	$16 \times 16 \times 768$
Swin-S	$64 \times 64 \times 192$	$32 \times 32 \times 384$	$16 \times 16 \times 768$

Table 3.9: Overview of the different feature map sizes ($H \times W \times C$) for the different backbones. Due to the non-hierarchical nature of the standard vision transformer, ViTDet-B/16 creates feature maps with the same number of channels, but decreases the size of the feature maps.

Hardware

Additional computational resources beyond those available to the author were required to conduct the experiments outlined in this thesis. The majority of this thesis' development was conducted on two distinct platforms. Preprocessing, development, and evaluation of the proposed models, i.e. work that could benefit from a Graphical Processing Unit (GPU), were completed using a provided Virtual Machine (VM) with a NVIDIA RTX8000. The training jobs were executed on the IDUN [65] cluster at NTNU. This cluster is dedicated to providing computational resources for research-related tasks. All experiments were trained using two GPUs, specifically NVIDIA A100 or NVIDIA P100.

As mentioned earlier, the VM was used for various purposes, including development and testing of model changes before initiating a larger training job on the IDUN [65] cluster. A graphical user interface was beneficial for this task, which was achieved using two different approaches: VMWare Horizon View [66] and the remote Secure Shell (SSH) connection feature in Visual Studio Code (VSCode). For the IDUN [65] cluster, the remote connection was primarily established through the terminal since it mostly included submitting Simple Linux Utility for Resource Management (SLURM) scripts via the command line.

3.4.3 Evaluation

The models were evaluated using standard metrics for object detection, as outlined in Section 2.5. These metrics include mAP, AP, F1-score, Precision, and Recall. The IoU thresholds of 0.5 and 0.7 were employed for this thesis when measuring the mAP and AP, following the standards defined in the CenterLoc3D [29] paper. As this thesis aims to address the localization task, it was also preferable to evaluate the model's precision in estimating the dimension and localization of the proposed bounding box. To determine the precision and error of the localization and dimension of the predicted 3D bounding box, the proposed metrics in the CenterLoc3D [29] paper were used. These are defined in Equation (3.1) and Equation (3.2), respectively.

$$P_{dim} = (1 - \sum_{k \in \{l_v, w_v, h_w\}} \frac{| k^{pred} - k^{gt} |}{k^{gt}}) \times 100\% \quad E_{dim} = \sum_{k \in \{l_v, w_v, h_w\}} | k^{pred} - k^{gt} | \tag{3.1}$$

$$P_{loc} = (1 - \sum_{k \in \{x, y\}} \frac{| k_{cen}^{pred} - k_{cen}^{gt}}{D_{rk}/2}) \times 100\% \quad E_{loc} = \sum_{k \in \{x, y\}} | k_{cen}^{pred} - k_{cen}^{gt} | \tag{3.2}$$

D_{rk} is the maximum distance that the roadside camera can perceive along and perpendicular to the road direction. The 3D vehicle centroids are obtained using Equation (2.20).

Chapter 4

Experiments and Results

The following sections present the findings for the Q-Free dataset. The results that were obtained from the SVLD-3D dataset, which was utilized as a pre-training dataset, are presented in Appendix A. The results are presented based on the used data augmentation strategy and include a quantitative and qualitative comparison of the model's performance utilizing different backbones. The models will henceforth be referred to by the backbone they employ. For instance, the model using the ResNet-101 backbone will be referred to as the "ResNet-101 model".

4.1 Experiment 1: No Data Augmentation

Backbone	mAP (%)		AP (%)		F1	Precision (%)	Recall (%)	FPS
ResNet-101$_{Baseline}$	@0.5	**61.4**	Car	91.7	0.94	92.9	95.8	23
			Truck	50.0	0.57	42.4	87.2	
			Bus	42.6	0.59	58.3	60.5	
ResNext-101$_{Baseline}$	@0.5	49.7	Car	93.4	0.95	94.3	96.2	17
			Truck	46.8	0.61	56.3	67.4	
			Bus	8.9	0.16	27.3	11.1	
ViTDet-B/16$_{Baseline}$	@0.5	27.1	Car	72.3	0.83	95.5	73.7	18
			Truck	9.2	0.16	72.7	9.3	
			Bus	0	0	0	0	
Swin-S$_{Baseline}$	@0.5	22.3	Car	48.7	0.66	94.7	50.6	16
			Truck	5.3	0.12	60.0	7.0	
			Bus	12.8	0.26	41.7	18.5	
ResNet-101$_{Baseline}$	@0.7	**52.5**	Car	79.6	0.87	85.7	88.3	23
			Truck	38.2	0.49	36.2	74.4	
			Bus	39.7	0.57	56.0	58.0	
ResNext-101$_{Baseline}$	@0.7	38.8	Car	85.5	0.90	88.9	90.7	17
			Truck	22.1	0.42	38.8	46.5	
			Bus	8.9	0.16	27.3	11.1	
ViTDet-B/16$_{Baseline}$	@0.7	17.6	Car	42.8	0.64	73.8	56.9	18
			Truck	4.9	0.12	54.6	7.0	
			Bus	0	0	0	0	
Swin-S$_{Baseline}$	@0.7	15.2	Car	35.0	0.54	77.7	41.5	16
			Truck	3.2	0.08	40.0	4.7	
			Bus	7.6	0.21	33.3	14.8	

Table 4.1: AP, mAP, F1-score, precision, and recall for the models using no data augmentation technique.

Based on the data presented in Table 4.1, it is evident that the ResNet-101 model demonstrated the highest mAP for both IoU thresholds of 0.5 and 0.7, achieving scores of 61.4% and 52.5%, respectively. For the car class, however, the ResNext-101 model achieved the highest AP for both thresholds, namely 93.4% and 85.5%. Due to the relatively poor performance of the bus class for ResNext-101, it achieved a mAP 11.7% and 13.7% lower than ResNet-101 for the IoU thresholds 0.5 and 0.7, respectively. Both CNN-based backbones demonstrated high precision and recall for the car class, indicating that the models accurately identified most of the relevant instances and that their predictions were reliable, i.e. the models predicted correctly. This can be seen in connection with the confusion matrix in Table 4.2.

As can be seen for the transformer-based models, i.e. ViTDet-B/16 and Swin-S, both had problems with the truck and bus class, resulting in an overall lower mAP than the CNN-based models. The ViTDet-B/16 model in particular, had issues with the bus class, showing no results for both IoU thresholds. Despite showing no detections for one of the three classes, ViTDet-B/16 achieved a higher overall mAP than Swin-S for both IoU thresholds. In contrast to the CNN-based backbones, the

transformer-based backbones had an overall higher precision than recall, indicating that the models predicted correctly when first predicting something but did not identify all relevant cases. As shown in Table 4.2, the transformer-based models had a much lower number of TPs than their CNN counterparts, indicating that they had many FNs, resulting in low recall. However, they did have relatively few FPs, hence the higher precision. The ResNet-101 model demonstrated superior performance in this experiment, achieving the highest overall scores and highest FPS rate. The ResNet-101 model exhibited a FPS of 23, while the other methods had similar a FPS ranging from 16 to 18.

Backbone	IoU Threshold	Class	TP	FP
ResNet-101$_{Baseline}$	@0.5	Car	473	36
		Truck	75	102
		Bus	49	35
ResNext-101$_{Baseline}$	@0.5	Car	475	29
		Truck	58	45
		Bus	9	24
ViTDet-B/16$_{Baseline}$	@0.5	Car	364	17
		Truck	8	3
		Bus	0	1
Swin-S$_{Baseline}$	@0.5	Car	250	14
		Truck	6	4
		Bus	21	15
ResNet-101$_{Baseline}$	@0.7	Car	436	73
		Truck	64	113
		Bus	47	37
ResNext-101$_{Baseline}$	@0.7	Car	448	56
		Truck	40	63
		Bus	9	24
ViTDet-B/16$_{Baseline}$	@0.7	Car	281	100
		Truck	6	5
		Bus	0	1
Swin-S$_{Baseline}$	@0.7	Car	205	59
		Truck	4	6
		Bus	24	12

Table 4.2: Confusion matrix for the different models using no data augmentation technique.

While the CNN-based models demonstrated the highest mAP scores, the transformer-based models exhibited higher precision in predicting the sizes of the vehicles. Table 4.3 illustrates that the ViTDet-B/16 and Swin-S achieved a size precision of 85.4% and 84.4%, respectively, compared to 81.7% and 83.3% for ResNet-101 and ResNext-101. For instance, ViTDet-B/16 had a length estimation error of 27cm, 24cm, and 34.5cm less than the errors observed for ResNet-101 and ResNext-101, respectively. The CNN-based approach ResNet-101, however, outperformed the other models in terms of localization precision which refers to the accuracy in terms of placement of the centroid of the proposed 3D bounding box. This was particularly evident for the XC error, i.e. the horizontal error of the

centroid, with an error of 7cm compared to around 20cm for the other approaches.

Backbone	Size Prec. (%)	Loc. Prec. (%)	Size Error (m)	Loc. Error (m)	L Error (m)	W Error (m)	H Error (m)	XC Error (m)	YC Error (m)	ZC Error (m)
ResNet-101$_{Baseline}$	81.7	92.4	0.839	0.542	0.615	0.0596	0.165	0.0721	0.387	0.0823
ResNext-101$_{Baseline}$	83.3	90.3	0.699	0.601	0.510	0.0510	0.138	0.225	0.307	0.0691
ViTDet-B/16$_{Baseline}$	85.4	90.8	0.421	0.543	0.270	0.0509	0.101	0.238	0.254	0.0503
Swin-S$_{Baseline}$	84.4	89.8	0.553	0.634	0.361	0.0558	0.136	0.212	0.355	0.0680

Table 4.3: Size and localization precision and error for the baseline models using no data augmentation technique.

Figures 4.1 and 4.2 show prediction and ground truth visualizations for the four models, for the front and rear views, respectively. The 3D bounding box for ResNet-101 and ResNext-101 appears more accurate when compared visually. These also have higher mAP values than the transformer-based models. ViTDet-B/16 and Swin-S both exhibit missing detection, less precise 3D bounding box localization, and overall low mAP. Swin-S also incorrectly classifies the car as a bus, as illustrated in Figures 4.1h and 4.1l.

(a) ResNet-101$_{Baseline}$

(b) ResNext-101$_{Baseline}$

(c) ViTDet-B/16$_{Baseline}$

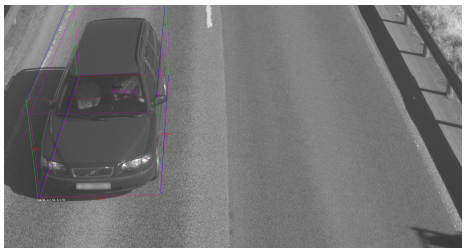
(d) Swin-S$_{Baseline}$

(e) ResNet-101$_{Baseline}$

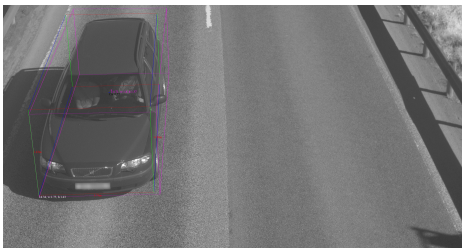
(f) ResNext-101$_{Baseline}$

(g) ViTDet-B/16$_{Baseline}$

(h) Swin-S$_{Baseline}$

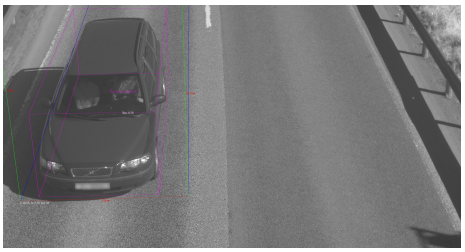


(i) ResNet-101$_{Baseline}$



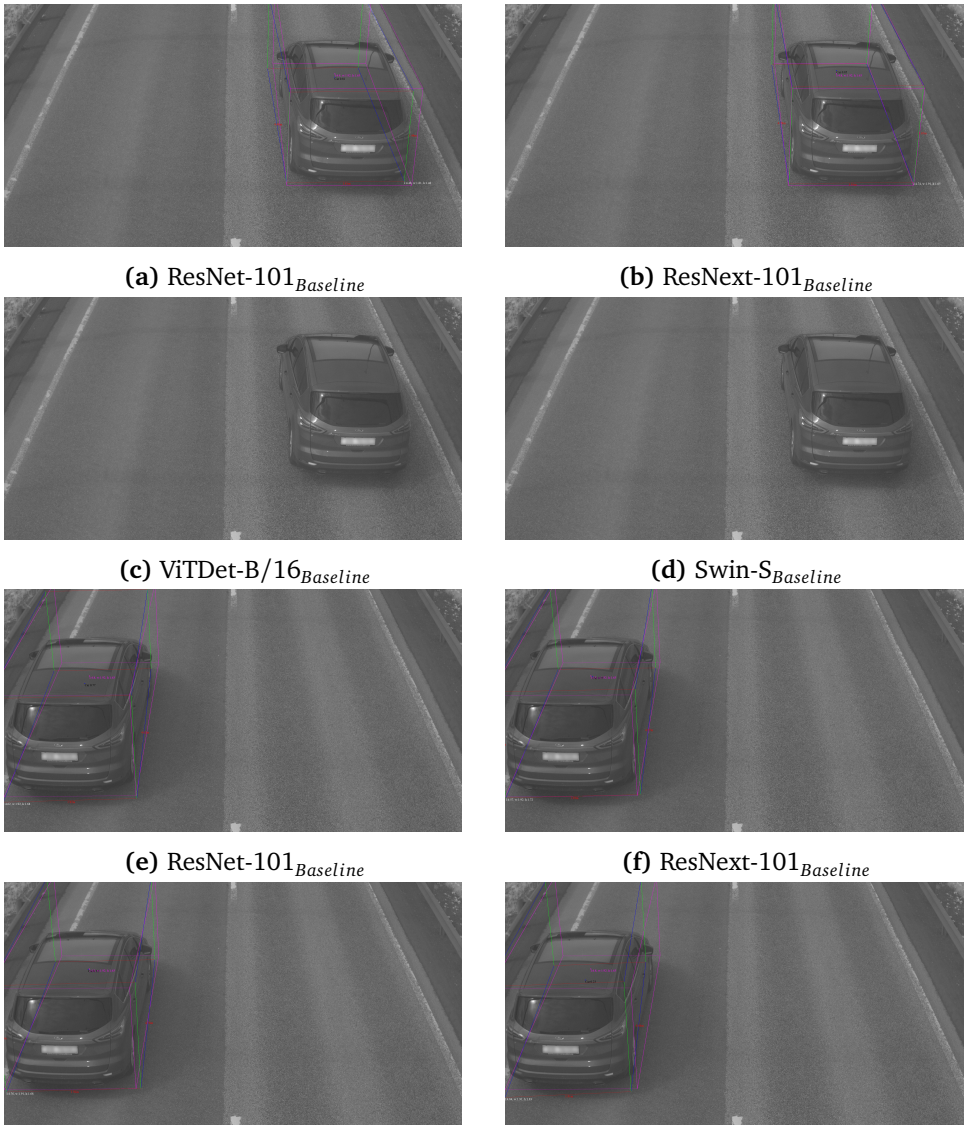
(j) ResNext-101$_{Baseline}$

(k) ViTDet-B/16$_{Baseline}$



(l) Swin-S$_{Baseline}$

Figure 4.1: Predictions and ground truth (purple bounding box) for the four models utilizing no data augmentation on the front view.



(a) ResNet-101$_{Baseline}$ **(b)** ResNext-101$_{Baseline}$

(c) ViTDet-B/16$_{Baseline}$ **(d)** Swin-S$_{Baseline}$

(e) ResNet-101$_{Baseline}$ **(f)** ResNext-101$_{Baseline}$

(g) ViTDet-B/16$_{Baseline}$ **(h)** Swin-S$_{Baseline}$

Figure 4.2: Predictions and ground truth (purple bounding box) for the four models utilizing no data augmentation on the rear view.

4.2 Experiment 2: Mixup

Backbone	mAP (%)		AP (%)		F1	Precision (%)	Recall (%)	FPS
ResNet-101$_{Mixup}$	@0.5	**63.4**	Car	87.8	0.93	90.2	95.0	24
			Truck	56.1	0.59	45.9	83.7	
			Bus	46.3	0.64	73.0	56.8	
ResNext-101$_{Mixup}$	@0.5	46.0	Car	91.3	0.92	89.4	95.3	16
			Truck	48.0	0.57	52.9	62.8	
			Bus	0.9	0.08	17.4	4.9	
ViTDet-B/16$_{Mixup}$	@0.5	11.8	Car	23.6	0.41	81.4	27.5	18
			Truck	5.2	0.16	31.0	10.5	
			Bus	6.5	0.19	24.5	16.1	
Swin-S$_{Mixup}$	@0.5	31.6	Car	48.7	0.65	91.9	57.0	17
			Truck	38.5	0.58	59.8	57.0	
			Bus	7.7	0.19	45.5	12.4	
ResNet-101$_{Mixup}$	@0.7	**50.3**	Car	80.5	0.87	85.0	89.5	24
			Truck	36.0	0.46	35.7	65.1	
			Bus	34.3	0.54	61.9	48.2	
ResNext-101$_{Mixup}$	@0.7	35.7	Car	76.3	0.83	80.1	85.4	16
			Truck	30.6	0.40	37.3	44.2	
			Bus	0.2	0.04	8.7	2.5	
ViTDet-B/16$_{Mixup}$	@0.7	6.6	Car	11.8	0.29	57.5	19.4	18
			Truck	3.4	0.10	20.7	7.0	
			Bus	4.5	0.16	20.8	13.6	
Swin-S$_{Mixup}$	@0.7	16.8	Car	31.8	0.50	69.6	38.5	17
			Truck	11.7	0.32	32.9	31.4	
			Bus	7.1	0.17	40.9	11.1	

Table 4.4: AP, mAP, F1-score, precision, and recall for the models using the Mixup data augmentation technique using the different backbones

Table 4.4 demonstrates that the CNN-based methods exhibited superior performance in AP for the different classes compared to the transformer-based methods, resulting in higher mAP values for both 0.5 and 0.7 thresholds. However, there were also considerable differences between the CNN-based approaches. Although ResNext-101 obtained the highest score for the car class, ResNet-101 demonstrated much higher scores for the remaining classes, namely truck, and bus. This was particularly evident for the bus class, where ResNext-101 nearly failed to detect any objects. As can be noticed from Table 4.5, ResNext-101 succeeded in detecting only 4 and 2 instances of the bus class out of the total 86 instances in the test dataset, for the thresholds of 0.5 and 0.7, respectively. Compared to the baseline models in Table 4.1, it can be seen that the ResNet-101 achieved a higher mAP, while ResNext-101 demonstrated a lower mAP, using the Mixup data augmentation technique. The increase for the ResNet-101 model was due to increased AP for the truck and bus class, whereas the decrease for ResNext-101 was mainly due to a significant drop in AP for the bus class. For the 0.5 threshold, ResNext-101$_{Mixup}$ showed a decrease of 8% for this class from 8.9% to only 0.9%.

As previously stated, the transformer-based approaches exhibited worse per-

formance overall than the CNN-based ones for all classes. This was particularly evident for the ViTDet-B/16 model, which only obtained a mAP of 11.8% and 6.6% for the two IoU thresholds. This represented a 15.3% and 11.0% decrease for the same threshold compared to the baseline ResNet-101 model, ResNet-101$_{Baseline}$, from the first experiment, as seen in Table 4.1. This was mainly due to the poor results for the car class, where ViTDet-B/16 only demonstrated an AP of 23.6% for the 0.5 IoU threshold and an AP of 11.8% for the 0.7 IoU threshold. Compared to the ViTDet-B/16 model, the Swin-S results did seem promising. A common observation among the transformer-based methodologies was their poor performance in terms of recall, which suggests that the models had problems identifying the relevant instances. This is evident from the confusion matrix presented in Table 4.5. Using the car class as an example, Swin-S with an IoU threshold of 0.5 had the highest number of TPs among the transformer-based approaches, with a count of 251. However, this number only represents around 50% of the total number of cars in the test dataset.

Table 4.4 demonstrates, similar to the results presented in Section 4.1, that ResNet-101 also outperformed the other models in terms of speed, achieving a FPS of 24, whereas the remaining models ranged between 16 and 18.

Backbone	IoU Threshold	Class	TP	FP
ResNet-101$_{Mixup}$	@0.5	Car	469	51
		Truck	72	85
		Bus	46	17
ResNext-101$_{Mixup}$	@0.5	Car	471	56
		Truck	54	48
		Bus	4	19
ViTDet-B/16$_{Mixup}$	@0.5	Car	136	31
		Truck	9	20
		Bus	13	40
Swin-S$_{Mixup}$	@0.5	Car	251	22
		Truck	49	33
		Bus	10	12
ResNet-101$_{Mixup}$	@0.7	Car	442	78
		Truck	56	101
		Bus	39	24
ResNext-101$_{Mixup}$	@0.7	Car	422	105
		Truck	38	64
		Bus	2	21
ViTDet-B/16$_{Mixup}$	@0.7	Car	96	71
		Truck	6	23
		Bus	11	42
Swin-S$_{Mixup}$	@0.7	Car	190	83
		Truck	27	55
		Bus	9	13

Table 4.5: Confusion matrix for the different models using the Mixup data augmentation technique.

The poor results for the ViTDet-B/16 were also evident in its performance

in terms of size and localization precision, achieving the lowest scores of 82.7% and 86.2%, respectively. The Swin-S model achieved higher precision for both metrics than ViTDet-B/16 and demonstrated greater size precision than ResNext-101. However, the size and localization precision of the Swin-S model were 0.5% and 0.8% higher for the Swin-$S_{Baseline}$ from the first experiment, respectively. ResNext-101 was, however, outperformed in terms of size precision by ResNet-101, which showed an increase of 1.3%. This was also much higher than for ResNet-$101_{Baseline}$, which had a size precision of 81.7%, as presented in Table 4.3. The localization performance of ResNet-101 was, on the other hand, slightly worse than that of ResNext-101.

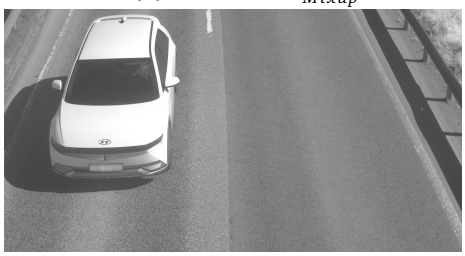
Backbone	Size Prec. (%)	Loc. Prec. (%)	Size Error (m)	Loc. Error (m)	L Error (m)	W Error (m)	H Error (m)	XC Error (m)	YC Error (m)	ZC Error (m)
ResNet-101_{Mixup}	84.3	89.2	0.716	0.668	0.533	0.0521	0.130	0.230	0.372	0.0651
ResNext-101_{Mixup}	83.0	89.9	0.684	0.619	0.495	0.0556	0.134	0.237	0.315	0.0669
ViTDet-B/16_{Mixup}	82.7	86.2	0.730	0.873	0.529	0.0478	0.152	0.219	0.578	0.0762
Swin-S_{Mixup}	83.9	89.0	0.658	0.681	0.464	0.0495	0.145	0.233	0.375	0.0723

Table 4.6: Size and localization precision and error for the models using the Mixup data augmentation technique.

Figures 4.3 and 4.4 show prediction and ground truth visualizations for the four models, for the front and rear views, respectively. The ViTDet-B/16 findings are also evident in these figures, with three of five samples missing detections. Swin-S demonstrated more predictions compared to ViTDet-B/16 but did overall achieve low APs and poor localization of the 3D bounding box. It also incorrectly classified the car from Figure 4.4d as a bus. ResNet-101 appears to have the best trade-off between AP and size and localization precision among the CNN-based approaches, despite ResNext-101 demonstrating the highest localization precision in Table 4.6.

(a) ResNet-101$_{Mixup}$

(b) ResNext-101$_{Mixup}$



(c) ViTDet-B/16$_{Mixup}$

(d) Swin-S$_{Mixup}$

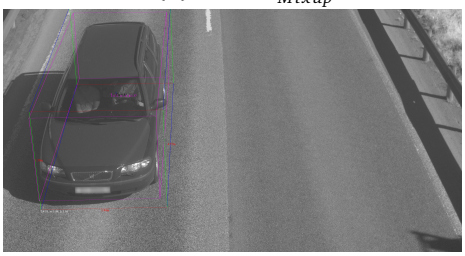
(e) ResNet-101$_{Mixup}$

(f) ResNext-101$_{Mixup}$

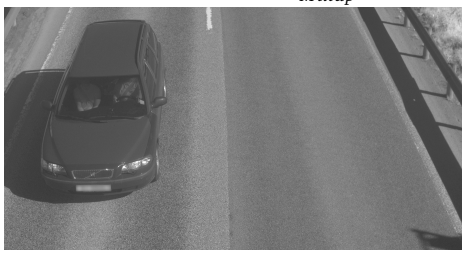
(g) ViTDet-B/16$_{Mixup}$

(h) Swin-S$_{Mixup}$

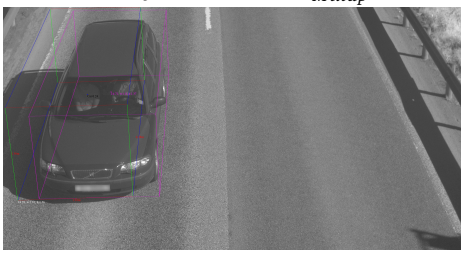
(i) ResNet-101$_{Mixup}$



(j) ResNext-101$_{Mixup}$

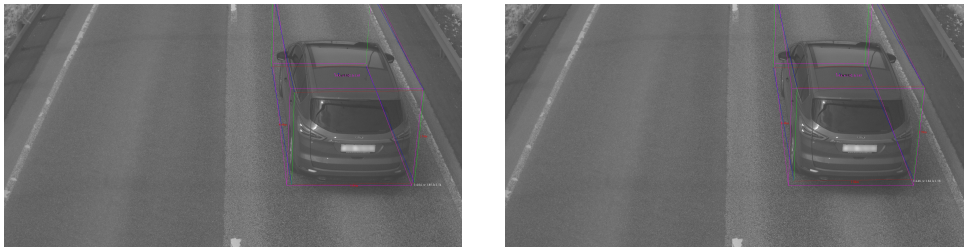


(k) ViTDet-B/16$_{Mixup}$

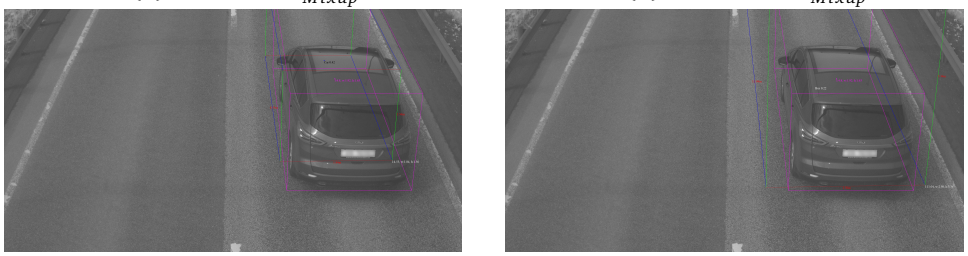


(l) Swin-S$_{Mixup}$

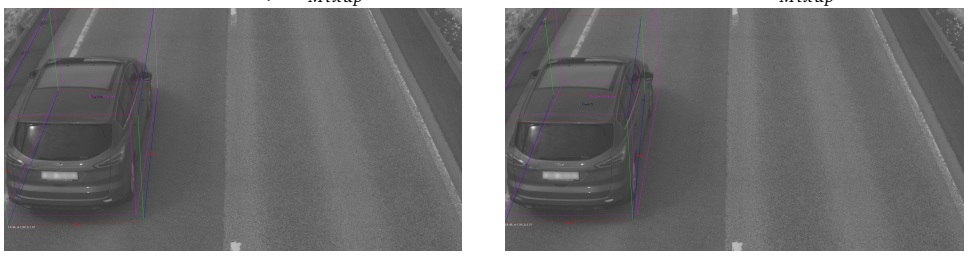
Figure 4.3: Predictions and ground truth (purple bounding box) for the four models utilizing the standard Mixup approach on the front view.



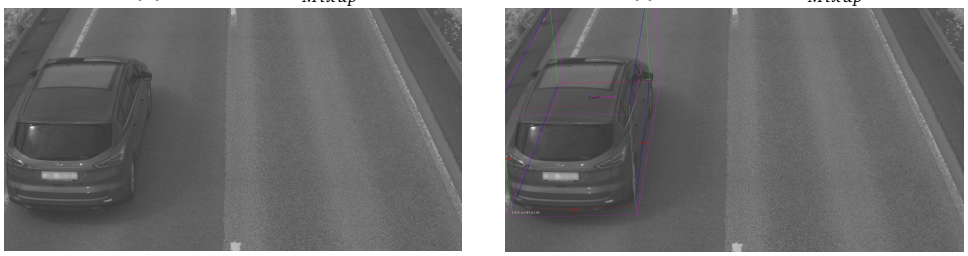
(a) ResNet-101$_{Mixup}$ **(b)** ResNext-101$_{Mixup}$



(c) ViTDet-B/16$_{Mixup}$ **(d)** Swin-S$_{Mixup}$



(e) ResNet-101$_{Mixup}$ **(f)** ResNext-101$_{Mixup}$



(g) ViTDet-B/16$_{Mixup}$ **(h)** Swin-S$_{Mixup}$

Figure 4.4: Predictions and ground truth (purple bounding box) for the four models utilizing the standard Mixup approach on the rear view.

4.3 Experiment 3: Mixup with IoU

Backbone	mAP (%)		AP (%)		F1	Precision (%)	Recall (%)	FPS
ResNet-101$_{MixupIoU}$	@0.5	**55.0**	Car	90.9	0.95	94.4	95.6	25
			Truck	44.8	0.54	44.7	68.6	
			Bus	29.3	0.49	52.9	45.7	
ResNext-101$_{MixupIoU}$	@0.5	43.9	Car	94.3	0.96	95.0	96.8	18
			Truck	21.8	0.38	45.2	32.6	
			Bus	15.5	0.27	66.7	17.3	
ViTDet-B/16$_{MixupIoU}$	@0.5	42.9	Car	72.4	0.83	93.2	39.5	18
			Truck	38.1	0.54	87.2	39.5	
			Bus	18.2	0.31	88.2	18.5	
Swin-S$_{MixupIoU}$	@0.5	24.5	Car	54.0	0.70	95.8	55.3	16
			Truck	2.6	0.07	75.0	3.5	
			Bus	16.8	0.32	58.1	22.2	
ResNet-101$_{MixupIoU}$	@0.7	**46.6**	Car	81.7	0.89	88.4	89.5	25
			Truck	29.7	0.44	36.4	55.8	
			Bus	28.5	0.48	51.4	44.4	
ResNext-101$_{MixupIoU}$	@0.7	39.1	Car	82.5	0.88	87.5	89.1	18
			Truck	19.4	0.34	40.3	29.1	
			Bus	15.5	0.27	66.7	17.3	
ViTDet-B/16$_{MixupIoU}$	@0.7	33.6	Car	53.4	0.67	75.4	60.1	18
			Truck	29.3	0.46	74.4	33.7	
			Bus	18.2	0.31	88.2	18.5	
Swin-S$_{MixupIoU}$	@0.7	16.2	Car	38.3	0.55	75.1	43.3	16
			Truck	1.6	0.04	50.0	2.3	
			Bus	8.9	0.23	41.9	16.1	

Table 4.7: F1-score, precision, recall, and corresponding AP and mAP for the models using the Mixup with IoU data augmentation technique.

According to the findings in Table 4.7, the ResNet-101 model demonstrated the most promising results in terms of mAP for both thresholds when implementing the Mixup strategy with the IoU threshold. In particular, the model achieved a mAP of 55.0% and 46.6% for the two thresholds, respectively. These results were significantly better than for the ResNext-101 model, which had a mAP 11.1% and 7.5% lower than ResNet-101 for the 0.5 and 0.7 IoU threshold, respectively. ResNet-101 also showed the best results in terms of speed, with an observed FPS of 25, compared to 16-18 for its competitors. These results correspond to those presented in both Sections 4.1 and 4.2. However, the mAPs obtained for ResNet-101 and ResNext-101 stand out negatively compared to the prior experiments. For instance, the ResNet-101 mAP was 9.4% lower than for ResNet-101$_{Mixup}$ for the 0.5 IoU threshold. The ResNext-101 also obtained a lower mAP for the 0.5 IoU threshold compared to the two prior experiments; however, the mAP for the 0.7 IoU threshold was 0.3% and 3.4% higher than the ones presented in Sections 4.1 and 4.2, respectively.

In this experiment, the Swin-S performed relatively poorly compared to the other models, primarily due to the poor performance of the truck class. As presen-

ted in Table 4.8, the Swin-S model had only 3 and 2 TPs for the truck class for the two thresholds, respectively. This resulted in over 80 non-identified instances for this particular class.

ViTDet-B/16, on the other hand, achieved comparable results to those of the CNN-based ResNext101 in terms of overall mAP. At the 0.5 threshold, the observed difference between them was only 1.0%. However, the difference increased notably for the 0.7 threshold due to a significant drop in the car AP for ViTDet-B/16. For the truck and bus classes, it can be observed that the ViTDet-B/16 obtained higher AP compared to the ResNext-101 model. Additionally, for the truck class in particular, the obtained AP was on par with the ResNet-101 results for the 0.7 IoU threshold. The ViTDet-B/16 outcomes were the best among all experiments, outperforming both ViTDet-B/16$_{Baseline}$ and ViTDet-B/16$_{Mixup}$.

Backbone	IoU Threshold	Class	TP	FP
ResNet-101$_{MixupIoU}$	@0.5	Car	472	28
		Truck	59	73
		Bus	37	33
ResNext-101$_{MixupIoU}$	@0.5	Car	478	25
		Truck	28	34
		Bus	14	7
ViTDet-B/16$_{MixupIoU}$	@0.5	Car	367	27
		Truck	34	5
		Bus	15	2
Swin-S$_{MixupIoU}$	@0.5	Car	273	12
		Truck	3	1
		Bus	18	13
ResNet-101$_{MixupIoU}$	@0.7	Car	442	58
		Truck	48	84
		Bus	36	34
ResNext-101$_{MixupIoU}$	@0.7	Car	440	63
		Truck	25	37
		Bus	14	7
ViTDet-B/16$_{MixupIoU}$	@0.7	Car	297	97
		Truck	29	10
		Bus	15	2
Swin-S$_{MixupIoU}$	@0.7	Car	214	71
		Truck	2	2
		Bus	13	18

Table 4.8: Confusion matrix for the different models using the Mixup with IoU data augmentation technique.

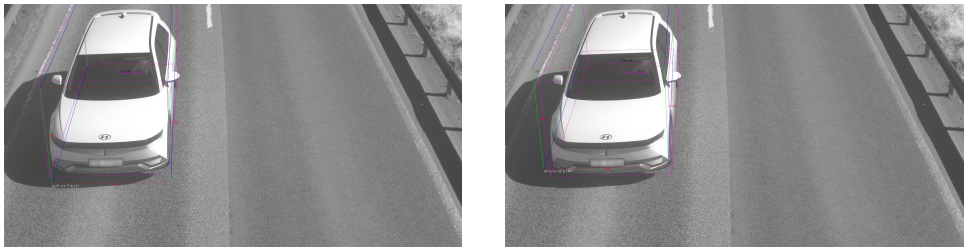
The positive results for the ViTDet-B/16 model were also evident in terms of size and localization precision, as presented in Table 4.9. This particular model demonstrated the highest size precision when compared to its competitors, surpassing ResNet-101, ResNext-101, and Swin-S by 1.8%, 0.8%, and 0.2%, respectively. Regarding localization precision, ResNext-101 exhibited the highest precision of 91.9%, beating its CNN- and transformer-based competitors. Overall, the localization precision results for all models in this experiment were higher than

those documented in Table 4.6 in Section 4.2.

Backbone	Size Prec. (%)	Loc. Prec. (%)	Size Error (m)	Loc. Error (m)	L Error (m)	W Error (m)	H Error (m)	XC Error (m)	YC Error (m)	ZC Error (m)
ResNet-101$_{MixupIoU}$	83.3	89.5	0.724	0.656	0.525	0.0548	0.145	0.224	0.360	0.0722
ResNext-101$_{MixupIoU}$	84.4	91.9	0.562	0.492	0.390	0.0520	0.120	0.204	0.228	0.0598
ViTDet-B/16$_{MixupIoU}$	85.1	90.1	0.497	0.588	0.335	0.0485	0.114	0.245	0.286	0.0570
Swin-S$_{MixupIoU}$	84.9	89.2	0.541	0.648	0.375	0.0507	0.115	0.247	0.343	0.0574

Table 4.9: Size and localization precision and error for the models using the Mixup with IoU data augmentation technique.

Figures 4.5 and 4.6 show prediction and ground truth visualizations for the four models, for the front and rear views, respectively. ResNet-101 produced the most accurate predictions overall for the front view, whereas ResNext-101 was more promising for the rear view, displaying a higher AP and more precise bounding box localization. Low APs and missing detections were apparent for the transformer-based ViTDet-B/16 and Swin-S models. However, ViTDet-B/16 produced one more prediction than Swin-S., which can be seen in Figures 4.5g and 4.5h.

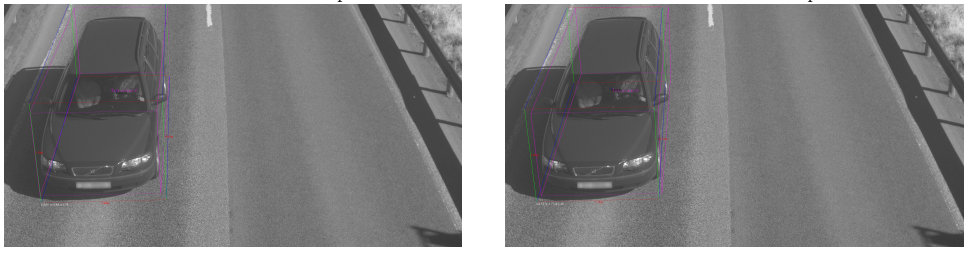


(a) ResNet-101$_{MixupIoU}$ **(b)** ResNext-101$_{MixupIoU}$

(c) ViTDet-B/16$_{MixupIoU}$ **(d)** Swin-S$_{MixupIoU}$

(e) ResNet-101$_{MixupIoU}$ **(f)** ResNext-101$_{MixupIoU}$

(g) ViTDet-B/16$_{MixupIoU}$ **(h)** Swin-S$_{MixupIoU}$



(i) ResNet-101$_{MixupIoU}$ **(j)** ResNext-101$_{MixupIoU}$

(k) ViTDet-B/16$_{MixupIoU}$ **(l)** Swin-S$_{MixupIoU}$

Figure 4.5: Predictions and ground truth (purple bounding box) for the four models utilizing the Mixup with IoU approach on the front view.

(a) ResNet-101$_{MixupIoU}$ **(b)** ResNext-101$_{MixupIoU}$

(c) ViTDet-B/16$_{MixupIoU}$ **(d)** Swin-S$_{MixupIoU}$

(e) ResNet-101$_{MixupIoU}$ **(f)** ResNext-101$_{MixupIoU}$

(g) ViTDet-B/16$_{MixupIoU}$ **(h)** Swin-S$_{MixupIoU}$

Figure 4.6: Predictions and ground truth (purple bounding box) for the four models utilizing the Mixup with IoU approach on the rear view.

4.4 Summary

This section attempts to summarize the findings from Tables 4.1, 4.4 and 4.7 into one table, Table 4.10, for easier comparisons between the results from the three experiments. Only the mAP for IoU thresholds of 0.5 and 0.7 is presented for better readability. Additionally, the figures included in Sections 4.1 to 4.3 are combined into four distinct figures, Figures 4.7a to 4.7d, based on the utilized backbone.

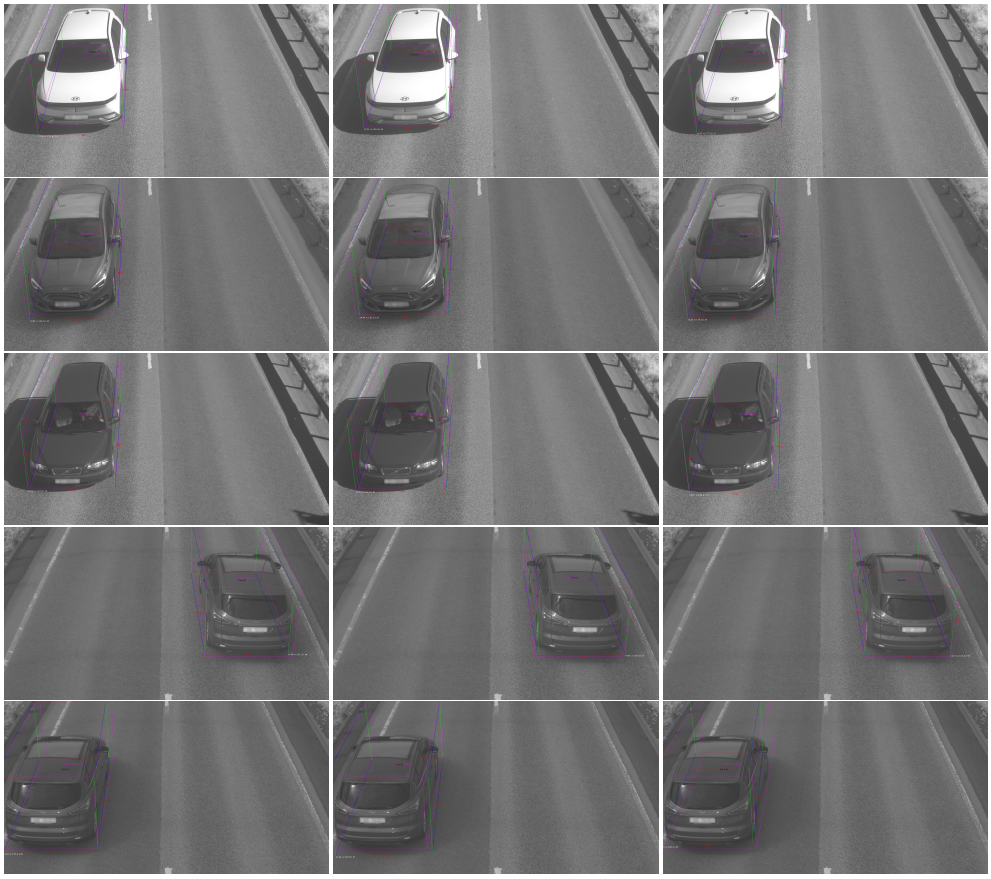
The experiments' findings are evident from Table 4.10 and Figure 4.7. The

transformer-based approaches, namely ViTDet-B/16 and Swin-S, achieved the best results using Mixup with IoU and Mixup, respectively. The visualizations in Figures 4.7c and 4.7d demonstrate that in most cases, the models got more detections using either Mixup or Mixup with IoU than their baseline models. However, these detections are imprecise in terms of both localization, size, and confidence.

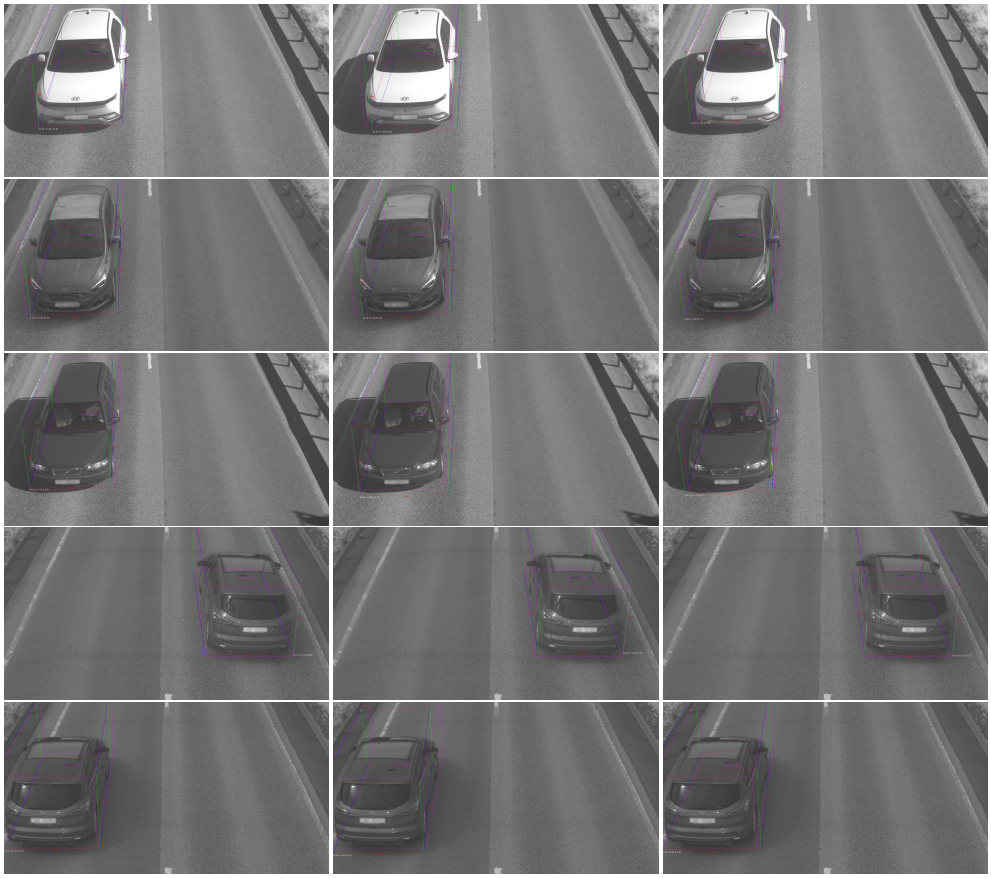
For the CNN-based approaches, namely ResNet-101 and ResNext-101, some of the best results were obtained using the baseline model, using no data augmentation technique. This is also visually evident. In most cases shown in Figures 4.7a and 4.7b, the baseline models exhibit the most accurate 3D bounding box, while the models utilizing Mixup techniques tend to be less precise.

Backbone	mAP	Baseline	Mixup	Mixup with IoU
ResNet-101	@0.5	61.4	63.4	55.0
	@0.7	52.5	50.3	46.6
ResNext-101	@0.5	49.7	46.0	43.9
	@0.7	38.8	35.7	39.1
ViTDet-B/16	@0.5	27.1	11.8	42.9
	@0.7	17.6	6.6	33.6
Swin-S	@0.5	22.3	31.6	24.5
	@0.7	15.2	16.8	16.2

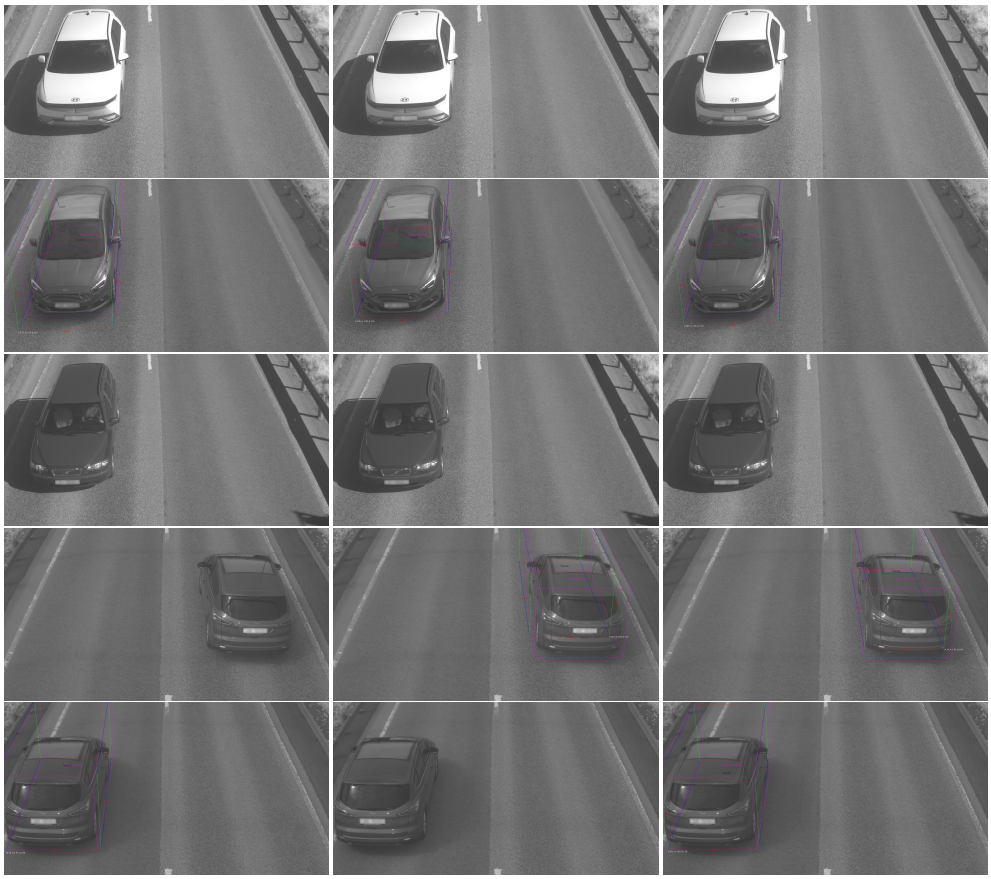
Table 4.10: Summary of the obtained mAP scores in the different experiments, i.e. baseline, Mixup, and Mixup with IoU for the different backbones. The experiments involving Mixup and Mixup with IoU yielded the highest mAP for the transformer-based model for both thresholds. ResNet-101 achieved the highest mAP@0.5 with the Mixup strategy and the highest mAP@0.7 with the baseline model. The ResNext-101 showed the highest mAP@0.5 with the baseline model and the highest mAP@0.7 on the Mixup with IoU experiment.



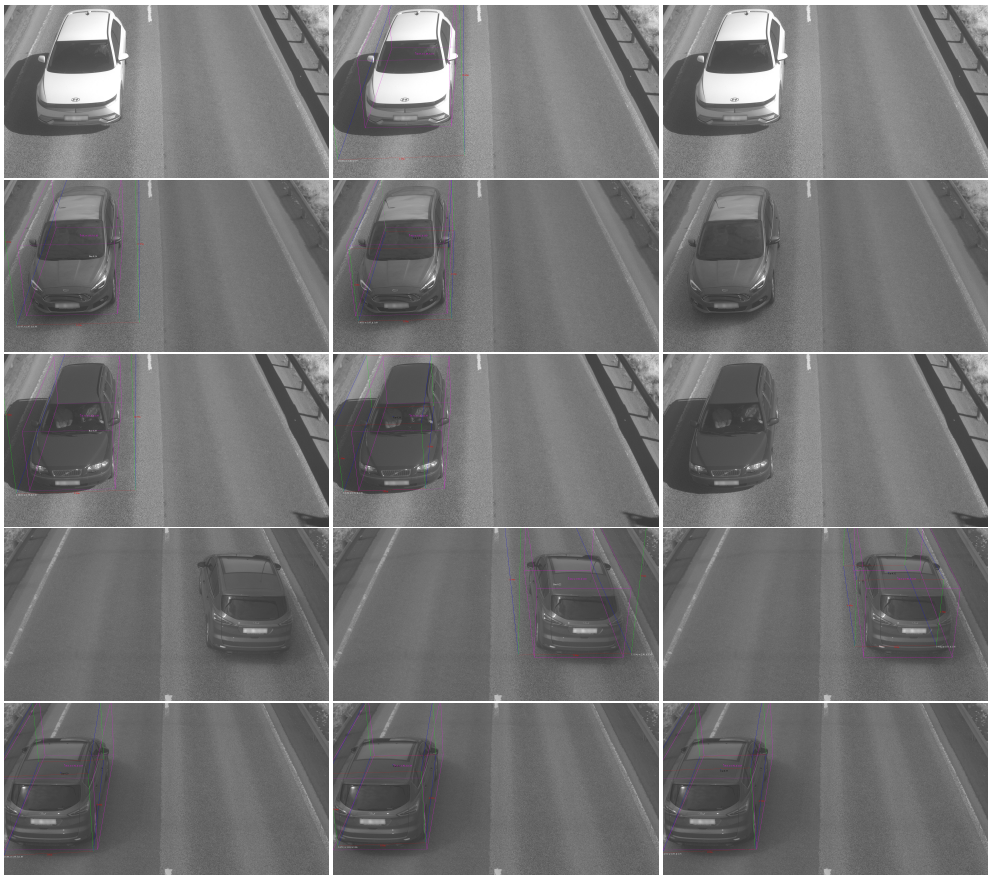
(a) ResNet-101.



(b) ResNext-101.



(c) ViTDet-B/16.



(d) Swin-S.

Figure 4.7: A summarized visualization of the four models used in the various experiments. First column: Baseline, second column: Mixup, and third column: Mixup with IoU.

Chapter 5

Discussion

This chapter focuses on analyzing and interpreting the results obtained in this study. The discussion will be revolved around the research questions posed in this thesis.

Regarding **RQ1** (*How accurate is volumetric identification of vehicles using monocular images?*), the findings presented in Sections 4.1 to 4.3 demonstrated that ResNet-101 models achieved the best results in terms of mAP across all three experiments and both IoU thresholds. The best ResNet-101 models were, however, achieved in different experiments, namely with ResNet-101_{Mixup} and ResNet-$101_{Baseline}$, where the former achieved an mAP of 63.4% and the latter an mAP of 52.5% for the 0.5 and 0.7 IoU thresholds, respectively. When comparing the CNN-based approaches, the results were somewhat counterintuitive. The mAP for the ResNet-101 models was around 11.1-17.4% and 7.5%-14.6% higher than the ResNext-101 models for the IoU thresholds of 0.5 and 0.7, respectively. ResNext models have demonstrated improved performance in the object detection tasks with the incorporation of the cardinality dimension, resulting in higher values of AP. It is, however, important to note that the majority of favorable ResNext outcomes have been observed solving the 2D object detection tasks, whereas this thesis has focused on 3D object detection.

The car class achieved the highest AP among all the models examined in this thesis. A potential factor contributing to this outcome may be attributed to the imbalanced distribution of samples for each class within the training dataset, with the truck and bus classes having approximately 1000 fewer samples than the car class. However, the primary contributor to why the model encountered a significant challenge with the truck and bus class may have been the nature of the training images. The perspective from which the images were captured did not fully capture the larger vehicles, such as trucks and buses. The vehicle's full height and length were not visible in most images containing these classes. It can thus have been a factor for the inaccurate results for these classes. For these classes, it would be preferable to use images where the entire vehicle is visible to see if the accuracy would increase.

Furthermore, the rear images of trucks may have caused a more significant challenge for all models. The number of annotations and the diversity in the appearance of trucks from the rear view may have been major contributors. While cars and most buses usually contain similar features from the rear view, trucks have significant variations depending on the load they are transporting. From the rear view, trucks encompass a range of vehicles, including those with standard cargo beds, agitator trucks, empty truck trailers, and those carrying loads such as wooden planks, logs, or other vehicles. A possible solution to enhance the model's robustness for such scenarios could be to add more samples from this view.

It was also evident in all experiments that the models had problems estimating the vehicle's length. The width and height errors were approximately 5-6cm and 10-15cm, respectively. However, in some cases, the length error was as high as 60cm. This could also be highly attributed to the perspective from which the images were taken. The camera pan angle was relatively close to 0, making the features along the vehicle length direction somewhat incomplete and consequently harder to estimate.

In addition to the already-discussed metrics, and as mentioned in Section 3.1, the speed of the models was also considered when choosing the different backbones for this thesis in addition to accuracy. All three experiments demonstrated that ResNet-101 overall had the best performance, with an FPS between 23-25. This was, however, not very surprising as it was one of the smallest models in this thesis. Based on the findings mentioned above, ResNet-101 seemed to be the most complete model as it served the best precision and speed.

With regards to **RQ2** (*How do transformer-based architectures compare to CNNs for feature extraction for monocular 3D object detection?*), the results of all experiments indicate that ViTDet-B/16 and Swin-S models consistently underperformed compared to the CNN-based approaches, exhibiting a relatively low mAP. This was evident when considering the number of TPs for the transformer-based methods across all experiments. The Swin-S models consistently had around 200 fewer detections than ResNet-101, its CNN counterpart. Like the Swin-S model, ViTDet-B/16 exhibited a notable difference in performance compared to its ResNext-101 counterpart, where ResNext-101 outperformed ViTDet-B/16 in most experiments. An exception was, however, found in the third experiment for the truck and bus classes, wherein ViTDet-B/16 outperformed its CNN counterpart. For the car class, however, the results were significantly lower for ViTDet-B/16.

When comparing the transformer-based approaches, it can be seen that the results favored the ViTDet-B/16 model. The ViTDet-B/16 models often demonstrated a greater number of TPs, and thus fewer FNs and FPs compared to the Swin-S models, resulting in higher mAPs. An exception was, however, ViTDet-B/16$_{Mixup}$ which had a relatively bad performance. Apart from this experiment, the results indicate that the simple pyramid is sufficient compared to the hierarchical approach Swin-S utilizes. The ViTDet-B/16 model was, however, larger than Swin-S, which could be the reason behind the findings in this thesis.

However, the transformer-based approaches seemed to achieve more prom-

ising results regarding size and localization precision. In terms of size precision, the ViTDet-B/16 achieved the highest results in the first and third experiments, ViTDet-B/16$_{Baseline}$ and ViTDet-B/16$_{MixupIoU}$, with a precision of 85.4% and 85.1%, respectively. Furthermore, the Swin-S model obtained the second-highest size precision score in all three experiments. Although these results seem promising, it should be taken into consideration that these models produced fewer predictions overall than the CNN-based approaches. As the size and precision measurements were solely performed on images with model detections, a considerable number of samples were excluded from the evaluation for the transformer-based approaches. Truck and bus detection were especially missing for these approaches. As discussed above, regarding **RQ1**, these classes presented an additional challenge as the entire vehicle was not captured within the image, making the estimation of the vehicle's dimension much harder. Thus, the promising results for the transformer-based approaches were most likely due to the fact that the measurements were mainly based on the results for the car class.

Regarding both **RQ1** and **RQ2**, a limitation of this thesis was the relatively limited amount of time dedicated to tuning the model's training hyperparameters. As discussed in Section 3.4.2, this was to ensure a fair comparison between the different models. However, the transformer-based approaches may have required other hyperparameters, given their distinct architecture compared to traditional CNNs. The hyperparameters utilized in this thesis were primarily derived from the original CenterLoc3D [29] paper, potentially resulting in a bias towards the CNN-based approaches such as ResNet-101 and ResNext-101. This may have contributed to the overall poorer performance of the transformer-based models compared to the CNN-based ones. As described earlier, transformer-based models have outperformed traditional CNNs in image classification and dense prediction tasks like object detection. Thus, the findings in this thesis were rather unexpected as they contradicted this trend. However, it should be noted that the domain of this thesis differs from that of the related work, as this thesis focuses on 3D object detection rather than 2D object detection.

To address **RQ3** (*Can training strategies from 2D object detectors be utilized to enhance the precision of monocular 3D object detectors?*), the primary focus of this thesis was implementing the Mixup data augmentation technique. Related work on the topic found that utilizing this technique increased the models' generalization ability and made them more robust. An enhanced version of the technique was also proposed, using a threshold to determine whether or not the mixing of two images should be performed. This approach demonstrated superior performance in the 3D object detection task compared to the original strategy. According to the findings presented in this thesis, the observed effect of the Mixup approaches was less prominent.

Looking at the standard Mixup technique, it can be observed that the ResNet-101$_{Mixup}$ obtained the highest mAP among all experiments using this strategy. The Swin-S$_{Mixup}$ also demonstrated some improved results in terms of mAP for both thresholds compared to its baseline counterpart, Swin-S$_{Baseline}$,

which did not employ any data augmentation. However, the results obtained using the standard Mixup strategy indicated that the models did not benefit from it, as they exhibited lower performance. This was especially evident for the ViTDet-B/16_{Mixup}, which demonstrated a significant reduction in mAP for both IoU thresholds when utilizing the standard Mixup technique. Although the AP for the bus class showed an increase from 0% to 6.5% and 3.4% for IoU 0.5 and 0.7, respectively, the AP for the car class demonstrated a substantial drop. The former could be attributed to the nature of the standard Mixup technique, where the number of samples of the bus class could have increased when selecting a random sample for mixing. The latter may be closely associated with the former. More bus samples during training may have increased the likelihood of noise generation for the car class. Due to the dataset's homogeneous distribution of vehicle placement, the randomly selected image would likely contain a vehicle in a similar position to the original image. Consequently, if the drawn image contained a bus, it would most likely cover the car completely, making the car less prominent than when not using the Mixup technique.

In terms of the enhanced Mixup strategy, utilizing an IoU threshold to determine whether or not to perform the Mixup, appeared to have a positive effect on the transformer-based approaches. Specifically, ViTDet-B/$16_{MixupIoU}$ showed significant improvement compared to its baseline counterpart. The CNN-based approaches exhibited the opposite results compared to their baseline counterpart, except for ResNext-$101_{MixupIoU}$, which demonstrated a slight improvement in mAP for IoU threshold 0.7 compared to ResNext-$101_{Baseline}$ at the same threshold. Compared to the standard Mixup strategy employed in the second experiment, it is evident that the Mixup strategy utilizing the IoU threshold generally resulted in lower mAP. One exception observed was the ViTDet-B/$16_{MixupIoU}$ model, which was primarily attributed to its poor precision using the standard Mixup technique. The results presented in this thesis related to the Mixup strategy differ from those of the prior study. The prior study suggested that including the IoU threshold in Mixup led to superior outcomes compared to the standard Mixup method. Although the IoU threshold of 0.4 was argued to be a good match for this thesis, it may have appeared too liberal. It was observed that despite being from the same position in the image, cars and larger vehicles, such as trucks and buses, often had an overlap below the set IoU threshold, resulting in the Mixup being performed. This could have generated more noise than the initial wished effect of serving a more diverse dataset.

Another observation was that the transformer-based methods benefited more from the Mixup data augmentation strategy than CNN-based strategies. Vision transformer networks have a history of being hard to optimize and suffering from overfitting, so it may appear that the Mixup strategies successfully addressed these challenges. However, it would be reasonable to assume that the CNN-based approaches would also benefit from these approaches.

Chapter 6

Conclusion and Further Work

This chapter presents the conclusions drawn from this thesis and identifies areas that require further investigation in future research.

6.1 Conclusion

The focus of this thesis has been to evaluate the performance of monocular 3D object detection, using both conventional object detection metrics and metrics specific for 3D object detection, such as size and localization precision and error. This thesis explored methods of optimizing the precision of the monocular 3D object detection model, which initially employed a CNN backbone.

The first strategy involved the integration of two vision transformer methodologies, namely ViTDet and Swin Transformer, as a backbone for the monocular 3D object detection task. They had shown remarkable results in image classification and dense prediction tasks such as object detection. Due to limited research on the topic, it was intriguing to investigate the potential benefits of this approach in this field and how they would compare to CNN backbones.

Another strategy was to investigate if the 3D object detection tasks could benefit from techniques that have shown great success in the 2D object detection task. The main focus of this thesis was on the Mixup data augmentation technique, which has been proven to be effective in improving model robustness and generalization Two different Mixup approaches were utilized. The first approach resembled the original Mixup, whereas the second approach incorporated an IoU threshold, which had shown enhanced performance compared to the standard Mixup technique in previous studies.

The results presented in this thesis indicated that transformer-based approaches did not provide any improvements for the 3D monocular object detection task. In all experiments conducted, it was observed that the transformer-based approaches were outperformed by either the CNN-based ResNet-101 or ResNext-101 models. Regarding the Mixup techniques, the results were also unexpected. In general, the CNN-based approaches demonstrated worse results when employing

the Mixup techniques than when no data augmentation was used. In contrast to the previous study, which found the Mixup with the IoU threshold to produce better results, most models in this thesis performed better with the standard Mixup approach.

6.2 Further Work

Further investigation into various aspects of the dataset would be the primary factor in improving the precision of the models. The Q-Free dataset used in this thesis had an imbalanced class distribution, particularly for the truck and bus classes, which was not beneficial. In addition to having fewer samples, the trucks and buses also posed an additional challenge. As previously discussed, the entire vehicle was not visible for these classes in the provided images, making the 3D object detection task challenging. Hence, it would be recommended to use images where the entire vehicle is visible in future work to better evaluate the model's performance in terms of size and localization.

The camera pan angle may also have contributed to imprecise vehicle size and localization measurements. Estimating the length was particularly hard due to the pan angle being relatively close to 0°. Hence, a solution for further work could be to use cameras with more suitable pan angles to see if the precision could be improved.

The data is also highly relevant regarding the precision of the transformer-based approaches. The results indicated that the ViTDet-B/16 and Swin-S models performed better when including the Mixup data augmentation strategy. This suggests that a more extensive and diverse dataset could potentially enhance their performance even further. Thus, it would be necessary to have a larger data volume and conduct additional experiments with the Mixup strategy for further work. Another critical factor when exploring the use of a transformer-based backbone would be to conduct further research on the optimal tuning of hyperparameters for these models. As discussed in Chapter 5, this was not investigated in this thesis, which may have affected the obtained results.

Bibliography

[1] A. Khan, A. Sohail, U. Zahoora and A. S. Qureshi, 'A Survey of the Recent Architectures of Deep Convolutional Neural Networks,' en, *Artificial Intelligence Review*, vol. 53, no. 8, pp. 5455–5516, Dec. 2020, arXiv:1901.06032 [cs], ISSN: 0269-2821, 1573-7462. DOI: 10.1007/s10462-020-09825-6. [Online]. Available: http://arxiv.org/abs/1901.06032 (visited on 07/06/2023).

[2] M. Fornes, 'Volumetric Identification of Vehicles on the Roadside Scene,' Tech. Rep., Dec. 2022. (visited on 05/05/2023).

[3] A. Çakır, *How Things Work: Artificial Neural Networks*, en, Dec. 2020. [Online]. Available: https://towardsdatascience.com/how-things-work-artificial-neural-networks-6b1291c43d75 (visited on 17/11/2022).

[4] A. F. Agarap, *Deep Learning using Rectified Linear Units (ReLU)*, arXiv:1803.08375 [cs, stat], Feb. 2019. [Online]. Available: http://arxiv.org/abs/1803.08375 (visited on 02/05/2023).

[5] M. A. Nielsen, 'Neural Networks and Deep Learning,' en, 2015, Publisher: Determination Press. [Online]. Available: http://neuralnetworksanddeeplearning.com (visited on 12/05/2023).

[6] abhigoku10, *Topic DL01: Activation functions and its Types in Artifical Neural network*, en, Apr. 2018. [Online]. Available: https://abhigoku10.medium.com/activation-functions-and-its-types-in-artifical-neural-network-14511f3080a8 (visited on 15/11/2022).

[7] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. [Online]. Available: https://www.deeplearningbook.org/contents/regularization.html (visited on 17/11/2022).

[8] S. Yang, *Deep learning basics — weight decay | by Sophia Yang | Analytics Vidhya | Medium*. [Online]. Available: https://medium.com/analytics-vidhya/deep-learning-basics-weight-decay-3c68eb4344e9 (visited on 05/12/2022).

[9] A. Kumar, *Weight Decay in Machine Learning: Concepts*, en-US, Jun. 2022. [Online]. Available: https://vitalflux.com/weight-decay-in-machine-learning-concepts/ (visited on 05/12/2022).

[10] C. Dilmegani, *Top Data Augmentation Techniques: Ultimate Guide*, en-US. [Online]. Available: https://research.aimultiple.com/data-augmentation-techniques/ (visited on 27/11/2022).

[11] L. Torrey and J. Shavlik, *Transfer Learning*. [Online]. Available: https://ftp.cs.wisc.edu/machine-learning/shavlik-group/torrey.handbook09.pdf (visited on 27/11/2022).

[12] *14.2. Fine-Tuning — Dive into Deep Learning 1.0.0-beta0 documentation*. [Online]. Available: https://d2l.ai/chapter_computer-vision/fine-tuning.html (visited on 28/05/2023).

[13] *Convolutional Neural Network*, May 2019. [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/convolutional-neural-network (visited on 17/11/2022).

[14] A. H. Reynolds, *Anh H. Reynolds*, en-us. [Online]. Available: https://anhreynolds.com/ (visited on 15/11/2022).

[15] *MaxpoolSample2.png - Computer Science Wiki*. [Online]. Available: https://computersciencewiki.org/index.php/File:MaxpoolSample2.png (visited on 15/11/2022).

[16] K. He, X. Zhang, S. Ren and J. Sun, *Deep Residual Learning for Image Recognition*, arXiv:1512.03385 [cs], Dec. 2015. [Online]. Available: http://arxiv.org/abs/1512.03385 (visited on 21/03/2023).

[17] S. Zagoruyko and N. Komodakis, *Wide Residual Networks*, arXiv:1605.07146 [cs], Jun. 2017. [Online]. Available: http://arxiv.org/abs/1605.07146 (visited on 27/05/2023).

[18] S. Xie, R. Girshick, P. Dollár, Z. Tu and K. He, *Aggregated Residual Transformations for Deep Neural Networks*, arXiv:1611.05431 [cs], Apr. 2017. [Online]. Available: http://arxiv.org/abs/1611.05431 (visited on 10/05/2023).

[19] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, *Going Deeper with Convolutions*, arXiv:1409.4842 [cs], Sep. 2014. [Online]. Available: http://arxiv.org/abs/1409.4842 (visited on 27/05/2023).

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, *Attention Is All You Need*, arXiv:1706.03762 [cs], Dec. 2017. [Online]. Available: http://arxiv.org/abs/1706.03762 (visited on 12/05/2023).

[21] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit and N. Houlsby, *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*, arXiv:2010.11929 [cs], Jun. 2021. [Online]. Available: http://arxiv.org/abs/2010.11929 (visited on 26/04/2023).

[22] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin and B. Guo, *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*, arXiv:2103.14030 [cs], Aug. 2021. [Online]. Available: http://arxiv.org/abs/2103.14030 (visited on 10/05/2023).

[23] Z. Liu, H. Hu, Y. Lin, Z. Yao, Z. Xie, Y. Wei, J. Ning, Y. Cao, Z. Zhang, L. Dong, F. Wei and B. Guo, *Swin Transformer V2: Scaling Up Capacity and Resolution*, arXiv:2111.09883 [cs], Apr. 2022. [Online]. Available: http://arxiv.org/abs/2111.09883 (visited on 23/05/2023).

[24] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng and R. Qu, 'A Survey of Deep Learning-Based Object Detection,' *IEEE Access*, vol. 7, pp. 128 837–128 868, 2019, Conference Name: IEEE Access, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2939201. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8825470 (visited on 31/10/2022).

[25] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, *You Only Look Once: Unified, Real-Time Object Detection*, arXiv:1506.02640 [cs], May 2016. DOI: 10.48550/arXiv.1506.02640. [Online]. Available: http://arxiv.org/abs/1506.02640 (visited on 31/10/2022).

[26] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and A. C. Berg, 'SSD: Single Shot MultiBox Detector,' in vol. 9905, arXiv:1512.02325 [cs], 2016, pp. 21–37. DOI: 10.1007/978-3-319-46448-0_2. [Online]. Available: http://arxiv.org/abs/1512.02325 (visited on 31/10/2022).

[27] X. Ma, W. Ouyang, A. Simonelli and E. Ricci, *3D Object Detection from Images for Autonomous Driving: A Survey*, arXiv:2202.02980 [cs], Feb. 2022. DOI: 10.48550/arXiv.2202.02980. [Online]. Available: http://arxiv.org/abs/2202.02980 (visited on 29/08/2022).

[28] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo and L. Shao, *Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions*, arXiv:2102.12122 [cs] version: 2, Aug. 2021. [Online]. Available: http://arxiv.org/abs/2102.12122 (visited on 24/05/2023).

[29] T. Xinyao, W. Wei, S. Huansheng and Z. Chunhui, *CenterLoc3D: Monocular 3D Vehicle Localization Network for Roadside Surveillance Cameras*, arXiv:2203.14550 [cs], Jan. 2023. DOI: 10.48550/arXiv.2203.14550. [Online]. Available: http://arxiv.org/abs/2203.14550 (visited on 27/01/2023).

[30] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan and S. Belongie, *Feature Pyramid Networks for Object Detection*, arXiv:1612.03144 [cs] version: 2, Apr. 2017. [Online]. Available: http://arxiv.org/abs/1612.03144 (visited on 24/05/2023).

[31] N. Ravi, S. Naqvi and M. El-Sharkawy, 'BIoU: An Improved Bounding Box Regression for Object Detection,' Sep. 2022. DOI: https://doi.org/10.3390/jlpea12040051. (visited on 19/05/2023).

[32] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid and S. Savarese, *Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression*, arXiv:1902.09630 [cs], Apr. 2019. [Online]. Available: http://arxiv.org/abs/1902.09630 (visited on 20/05/2023).

[33] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye and D. Ren, *Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression*, arXiv:1911.08287 [cs], Nov. 2019. [Online]. Available: http://arxiv.org/abs/1911.08287 (visited on 13/05/2023).

[34] J. Mao, S. Shi, X. Wang and H. Li, *3D Object Detection for Autonomous Driving: A Review and New Outlooks*, arXiv:2206.09474 [cs], Jun. 2022. [Online]. Available: http://arxiv.org/abs/2206.09474 (visited on 17/10/2022).

[35] S. Ren, K. He, R. Girshick and J. Sun, *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*, arXiv:1506.01497 [cs], Jan. 2016. [Online]. Available: http://arxiv.org/abs/1506.01497 (visited on 31/10/2022).

[36] T. Wood, *F-Score*, May 2019. [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/f-score (visited on 05/12/2022).

[37] *Mean Average Precision (mAP) Explained: Everything You Need to Know*, en. [Online]. Available: https://www.v7labs.com/blog/mean-average-precision,%20https://www.v7labs.com/blog/mean-average-precision (visited on 16/05/2023).

[38] R. Padilla, S. Netto and E. da Silva, *A Survey on Performance Metrics for Object-Detection Algorithms*. Jul. 2020. [Online]. Available: https://www.researchgate.net/profile/Rafael-Padilla/publication/343194514_A_Survey_on_Performance_Metrics_for_Object-Detection_Algorithms/links/5f1b5a5e45851515ef478268/A-Survey-on-Performance-Metrics-for-Object-Detection-Algorithms.pdf (visited on 27/11/2022).

[39] A. Anwar, *What are Intrinsic and Extrinsic Camera Parameters in Computer Vision?* en, Mar. 2022. [Online]. Available: https://towardsdatascience.com/what-are-intrinsic-and-extrinsic-camera-parameters-in-computer-vision-7071b72fb8ec (visited on 23/05/2023).

[40] Z. Liu, Z. Wu and R. Tóth, *SMOKE: Single-Stage Monocular 3D Object Detection via Keypoint Estimation*, arXiv:2002.10111 [cs], Feb. 2020. [Online]. Available: http://arxiv.org/abs/2002.10111 (visited on 31/10/2022).

[41] P. Li, H. Zhao, P. Liu and F. Cao, *RTM3D: Real-time Monocular 3D Detection from Object Keypoints for Autonomous Driving*, arXiv:2001.03343 [cs, eess], Jan. 2020. [Online]. Available: http://arxiv.org/abs/2001.03343 (visited on 31/10/2022).

[42] A. Mousavian, D. Anguelov, J. Flynn and J. Kosecka, *3D Bounding Box Estimation Using Deep Learning and Geometry*, arXiv:1612.00496 [cs], Apr. 2017. [Online]. Available: http://arxiv.org/abs/1612.00496 (visited on 31/10/2022).

[43] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg and L. Fei-Fei, *ImageNet Large Scale Visual Recognition Challenge*, arXiv:1409.0575 [cs], Jan. 2015. DOI: 10.48550/arXiv.1409.0575. [Online]. Available: http://arxiv.org/abs/1409.0575 (visited on 13/12/2022).

[44] J. Sochor, R. Juránek, J. Špaňhel, L. Maršík, A. Široký, A. Herout and P. Zemčík, 'Comprehensive Data Set for Automatic Single Camera Visual Speed Measurement,' *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 5, pp. 1633–1643, May 2019, Conference Name: IEEE Transactions on Intelligent Transportation Systems, ISSN: 1558-0016. DOI: 10.1109/TITS.2018.2825609.

[45] P. Li, *Monocular 3D Detection with Geometric Constraints Embedding and Semi-supervised Training*, arXiv:2009.00764 [cs], Sep. 2020. [Online]. Available: http://arxiv.org/abs/2009.00764 (visited on 11/12/2022).

[46] *The KITTI Vision Benchmark Suite*. [Online]. Available: https://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d (visited on 11/12/2022).

[47] A. Mumuni and F. Mumuni, 'Data augmentation: A comprehensive survey of modern approaches,' en, *Array*, vol. 16, p. 100 258, Dec. 2022, ISSN: 25900056. DOI: 10.1016/j.array.2022.100258. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2590005622000911 (visited on 21/05/2023).

[48] A. Bochkovskiy, C.-Y. Wang and H.-Y. M. Liao, *YOLOv4: Optimal Speed and Accuracy of Object Detection*, arXiv:2004.10934 [cs, eess], Apr. 2020. [Online]. Available: http://arxiv.org/abs/2004.10934 (visited on 28/02/2023).

[49] H. Zhang, M. Cisse, Y. N. Dauphin and D. Lopez-Paz, *Mixup: Beyond Empirical Risk Minimization*, arXiv:1710.09412 [cs, stat], Apr. 2018. [Online]. Available: http://arxiv.org/abs/1710.09412 (visited on 20/02/2023).

[50] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe and Y. Yoo, *CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features*, arXiv:1905.04899 [cs], Aug. 2019. [Online]. Available: http://arxiv.org/abs/1905.04899 (visited on 21/05/2023).

[51] S. Bouabid and V. Delaitre, *Mixup Regularization for Region Proposal based Object Detectors*, arXiv:2003.02065 [cs], Mar. 2020. [Online]. Available: http://arxiv.org/abs/2003.02065 (visited on 21/05/2023).

[52] Z. Zhang, T. He, H. Zhang, Z. Zhang, J. Xie and M. Li, *Bag of Freebies for Training Object Detection Neural Networks*, arXiv:1902.04103 [cs], Apr. 2019. [Online]. Available: http://arxiv.org/abs/1902.04103 (visited on 28/02/2023).

[53] S. T, S. M, K. Santhakumar, B. R. Kiran, T. Gauthier and S. Yogamani, *Exploring 2D Data Augmentation for 3D Monocular Object Detection*, arXiv:2104.10786 [cs], Apr. 2021. [Online]. Available: http://arxiv.org/abs/2104.10786 (visited on 28/02/2023).

[54] A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly and N. Houlsby, *Big Transfer (BiT): General Visual Representation Learning*, arXiv:1912.11370 [cs] version: 3, May 2020. [Online]. Available: http://arxiv.org/abs/1912.11370 (visited on 22/05/2023).

[55] Y. Li, H. Mao, R. Girshick and K. He, *Exploring Plain Vision Transformer Backbones for Object Detection*, arXiv:2203.16527 [cs], Jun. 2022. [Online]. Available: http://arxiv.org/abs/2203.16527 (visited on 28/04/2023).

[56] *[1405.0312] Microsoft COCO: Common Objects in Context*. [Online]. Available: https://arxiv.org/abs/1405.0312 (visited on 22/05/2023).

[57] K. He, G. Gkioxari, P. Dollár and R. Girshick, *Mask R-CNN*, arXiv:1703.06870 [cs], Jan. 2018. [Online]. Available: http://arxiv.org/abs/1703.06870 (visited on 22/05/2023).

[58] Z. Cai and N. Vasconcelos, *Cascade R-CNN: High Quality Object Detection and Instance Segmentation*, arXiv:1906.09756 [cs], Jun. 2019. [Online]. Available: http://arxiv.org/abs/1906.09756 (visited on 22/05/2023).

[59] K. He, X. Chen, S. Xie, Y. Li, P. Dollár and R. Girshick, *Masked Autoencoders Are Scalable Vision Learners*, arXiv:2111.06377 [cs], Dec. 2021. [Online]. Available: http://arxiv.org/abs/2111.06377 (visited on 22/05/2023).

[60] L. Wang and A. Tien, 'Aerial Image Object Detection With Vision Transformer Detector (ViTDet),' en,

[61] G.-S. Xia, X. Bai, J. Ding, Z. Zhu, S. Belongie, J. Luo, M. Datcu, M. Pelillo and L. Zhang, *DOTA: A Large-scale Dataset for Object Detection in Aerial Images*, arXiv:1711.10398 [cs], May 2019. [Online]. Available: http://arxiv.org/abs/1711.10398 (visited on 22/05/2023).

[62] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick and P. Dollár, *Microsoft COCO: Common Objects in Context*, arXiv:1405.0312 [cs], Feb. 2015. [Online]. Available: http://arxiv.org/abs/1405.0312 (visited on 22/05/2023).

[63] stjuliet, *Labelimg3D*, original-date: 2021-03-22T03:18:14Z, Nov. 2022. [Online]. Available: https://github.com/stjuliet/Labelimg3D (visited on 27/11/2022).

[64] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, arXiv:1412.6980 [cs], Jan. 2017. [Online]. Available: http://arxiv.org/abs/1412.6980 (visited on 02/11/2022).

[65] M. Själander, M. Jahre, G. Tufte and N. Reissmann, *EPIC: An Energy-Efficient, High-Performance GPGPU Computing Research Infrastructure*, arXiv:1912.05848 [cs], Feb. 2022. DOI: 10.48550/arXiv.1912.05848. [Online]. Available: http://arxiv.org/abs/1912.05848 (visited on 05/12/2022).

[66] *VMware Horizon | VDI Software Solutions*, en. [Online]. Available: https://www.vmware.com/products/horizon.html (visited on 12/06/2023).

Appendix A

Original Dataset Results

A.1 ResNet-101

Backbone	mAP (%)		AP (%)		F1	Precision (%)	Recall (%)
ResNet-101	@0.5	26.6	Car	71.3	0.76	79.1	73.8
			Truck	0.0	0.0	0.0	0.0
			Bus	8.6	0.18	78.7	9.9
	@0.7	18.4	Car	47.9	0.57	59.0	55.0
			Truck	0.0	0.0	0.0	0.0
			Bus	7.4	0.16	70.2	8.9

Table A.1: AP, mAP, F1-score, precision, and recall for ResNet-101.

Backbone	IoU Threshold	Class	TP	FP
ResNet-101	@0.5	Car	5455	1446
		Truck	0	33
		Bus	74	20
	@0.7	Car	4070	2831
		Truck	0	33
		Bus	66	28

Table A.2: Confusion matrix for ResNet-101.

Backbone	Size Prec. (%)	Loc. Prec. (%)	Size Error /m	Loc. Error /m	L Error /m	W Error /m	H Error /m	XC Error /m	YC Error /m	ZC Error /m
ResNet-101	89.6	98.6	0.316	0.374	0.226	0.0444	0.0453	0.0449	0.307	0.0226

Table A.3: Size and localization precision and error for ResNet-101.

A.2 ResNext-101

Backbone	mAP (%)		AP (%)		F1	Precision (%)	Recall (%)
ResNext-101	@0.5	31.1	Car	71.1	0.78	81.1	75.0
			Truck	0.0	0.0	0.0	0.0
			Bus	22.2	0.38	80.3	24.6
	@0.7	20.4	Car	51.6	0.60	62.8	58.1
			Truck	0.0	0.0	0.0	0.0
			Bus	9.6	0.22	46.5	14.2

Table A.4: AP, mAP, F1-score, precision, and recall for ResNext-101.

Backbone	IoU Threshold	Class	TP	FP
ResNext-101	@0.5	Car	5546	1297
		Truck	0	60
		Bus	183	45
	@0.7	Car	4296	2547
		Truck	0	60
		Bus	106	122

Table A.5: Confusion matrix for ResNext-101.

Backbone	Size Prec. (%)	Loc. Prec. (%)	Size Error /m	Loc. Error /m	L Error /m	W Error /m	H Error /m	XC Error /m	YC Error /m	ZC Error /m
ResNext-101	88.9	98.5	0.355	0.418	0.259	0.0450	0.513	0.0493	0.343	0.0257

Table A.6: Size and localization precision and error for ResNext-101.

A.3 ViTDet-B/16

Backbone	mAP (%)		AP (%)		F1	Precision (%)	Recall (%)
ViTDet-B/16	@0.5	21.0	Car	57.7	0.66	70.0	62.2
			Truck	0.0	0.0	0.0	0.0
			Bus	5.3	0.20	30.1	15.4
	@0.7	11.9	Car	35.7	0.46	48.7	43.3
			Truck	0.0	0.0	0.0	0.0
			Bus	0.13	0.03	4.7	2.4

Table A.7: AP, mAP, F1-score, precision, and recall for ViTDet-B/16.

Backbone	IoU Threshold	Class	TP	FP
ViTDet-B/16	@0.5	Car	4600	1977
		Truck	0	0
		Bus	115	267
	@0.7	Car	3202	3375
		Truck	0	0
		Bus	18	364

Table A.8: Confusion matrix for ViTDet-B/16.

Backbone	Size Prec. (%)	Loc. Prec. (%)	Size Error /m	Loc. Error /m	L Error /m	W Error /m	H Error /m	XC Error /m	YC Error /m	ZC Error /m
ViTDet-B/16	89.8	97.8	0.310	0.476	0.216	0.0444	0.0498	0.0815	0.310	0.0249

Table A.9: Size and localization precision and error for ViTDet-B/16.

A.4 Swin-S

Backbone	mAP (%)		AP (%)		F1	Precision (%)	Recall (%)
Swin-S	@0.5	25.5	Car	63.6	0.69	69.3	69.0
			Truck	0.0	0.0	0.0	0.0
			Bus	12.9	0.26	56.2	17.1
	@0.7	16.3	Car	41.8	0.49	49.3	49.0
			Truck	0.0	0.0	0.0	0.0
			Bus	7.1	0.18	38.1	49.0

Table A.10: AP, mAP, F1-score, precision, and recall for Swin-S.

Backbone	IoU Threshold	Class	TP	FP
Swin-S	@0.5	Car	5101	2258
		Truck	0	10
		Bus	127	99
	@0.7	Car	3626	3733
		Truck	0	0
		Bus	86	140

Table A.11: Confusion matrix for Swin-S.

Backbone	Size Prec. (%)	Loc. Prec. (%)	Size Error /m	Loc. Error /m	L Error /m	W Error /m	H Error /m	XC Error /m	YC Error /m	ZC Error /m
Swin-S	89.6	98.0	0.318	0.472	0.224	0.0432	0.0510	0.0724	0.374	0.0255

Table A.12: Size and localization precision and error for Swin-S.

Appendix B

Q-Free Dataset BBox Regression Results

Dataset Distribution (Car/Bus/Truck):

- Train: 3260/23/108
- Test: 490/2/9

Iou

Backbone	mAP@0.5	mAP@0.7	Size Precision	Loc. Precision	Size Error (m)	Loc. Error (m)	L Error (m)	W Error (m)	H Error (m)	XC Error (m)	YC Error (m)	ZC Error (m)
ResNet101	28.3% (Car = 84.8%)	26.5% (Car = 79.5%)	84.7%	89.7%	0.423	0.531	0.265	0.0540	0.104	0.151	0.328	0.0521

Ciou

Backbone	mAP@0.5	mAP@0.7	Size Precision	Loc. Precision	Size Error (m)	Loc. Error (m)	L Error (m)	W Error (m)	H Error (m)	XC Error (m)	YC Error (m)	ZC Error (m)
ResNet101	30.0% (Car = 90.0%)	28.1% (Car = 84.2%)	86.4%	94.3%	0.375	0.313	0.236	0.0440	0.0485	0.0880	0.178	0.0474

Cdiou

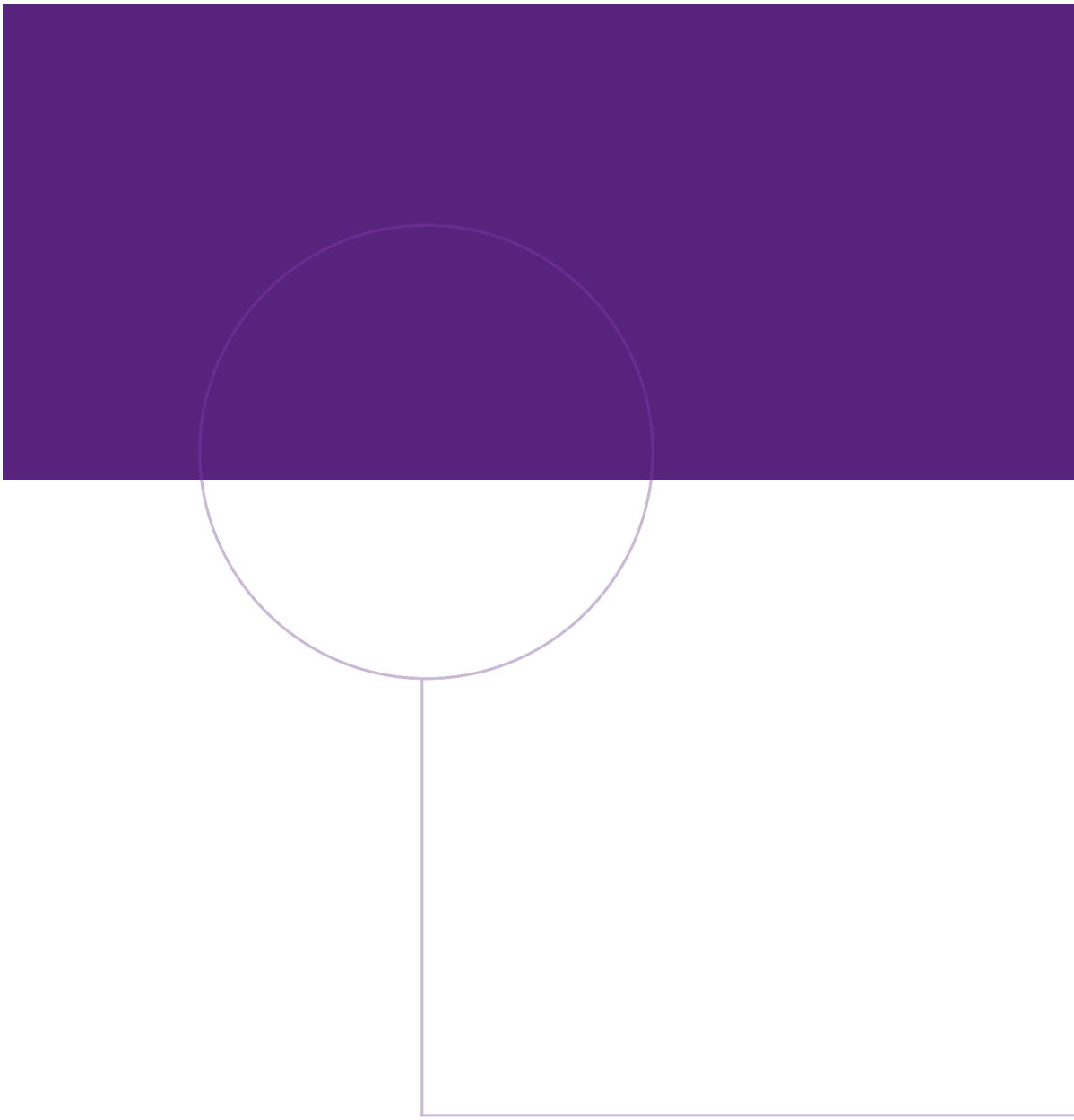
Backbone	mAP@0.5	mAP@0.7	Size Precision	Loc. Precision	Size Error (m)	Loc. Error (m)	L Error (m)	W Error (m)	H Error (m)	XC Error (m)	YC Error (m)	ZC Error (m)
ResNet101	27.97% (Car = 83.9%)	25.8% (Car = 77.5%)	84.5%	90.8%	0.413	0.478	0.248	0.0574	0.108	0.143	0.281	0.0539

Diou

Backbone	mAP@0.5	mAP@0.7	Size Precision	Loc. Precision	Size Error (m)	Loc. Error (m)	L Error (m)	W Error (m)	H Error (m)	XC Error (m)	YC Error (m)	ZC Error (m)
ResNet101	27.1% (Car = 81.2)%	25.4% (Car = 76.3%)	84.3%	88.5%	0.431	0.583	0.269	0.0584	0.103	0.168	0.364	0.0517

Giou

Backbone	mAP@0.5	mAP@0.7	Size Precision	Loc. Precision	Size Error (m)	Loc. Error (m)	L Error (m)	W Error (m)	H Error (m)	XC Error (m)	YC Error (m)	ZC Error (m)
ResNet101	29.3% (Car = 87.8%)	27.6% (Car = 82.9%)	84.1%	90.1%	0.448	0.514	0.285	0.0532	0.110	0.140	0.320	0.0548



NTNU

Norwegian University of
Science and Technology