Norges miljø- og biovitenskapelige universitet

Master's Thesis 2024 30 ECTS
Faculty of Science and Technology

Deep Learning based Models for Traffic Participant and Object Classification

Sathuriyan Sivathas
MSc in Industrial Economics and Technology Management

Abstract

Research on deep learning models is constantly advancing, and has emerged as an important field to study. Deep learning models have a vital role in the development of self-driving cars, making it essential to thoroughly understand their performance. This thesis focuses on the performance of selected deep learning models on various datasets related to traffic objects and participants. The deep learning models evaluated in this thesis include VGG-16, ResNet-50, WideResNet-50-2, EfficientNetB0 and Vision Transformer, all explored within the context of transfer learning. These models have been further trained on three datasets based on the images from the Audi Autonomous Driving Dataset (A2D2), specifically prepared for classifying traffic participants and objects. The first dataset comprises of normal and augmented images, and is referred to as *NAI* dataset. The second dataset, referred to as the *NANI* dataset, comprises of normal, augmented and noisy mixed-class images. The third dataset consists of normal, augmented and synthetic images generated from a Deep Convolutional Generative Adversarial Network (DCGAN), and is referred to as *NASI* dataset. All deep learning models were trained on the *NAI*, and *NANI* datasets, while VGG-16 and Vision Transformer are the only models to be trained on the *NASI* dataset. Through these datasets, the impact of normal and augmented images, the impact of noisy mixed-class images and the impact of images generated by a DCGANs are evaluated. The VGG-16 model outperformed all others, and achieved consistent performance across all three datasets. The WideResNet-50-2 model also performed well on the two datasets it was evaluated on, but did not match the performance of the VGG-16 model. The Vision Transformer model also showed promising results across all datasets, particularly in terms of consistency and stability. These results indicate that deep learning models can perform effectively and consistently across diverse datasets involving traffic participant and objects, whether the images are normal, augmented, noisy, or synthetic.

Acknowledgements

This thesis marks the end of my Master´s degree in Industrial Economics and Technology Management at the Norwegian University of Life Sciences (NMBU). I am deeply grateful to my main supervisor, Habib Ullah, for his continuous support and guidance throughout the writing process. Similarly, I appreciate the insightful comments from my co-supervisor, Fadi Al Machot, which significantly enriched this thesis.

I would like to give special recognition to my family and friends, for their continuous support during my time at NMBU. In particular, my parents, Rajakumary Sivathas and Sivathas Sivabalasingam, have provided unparalleled support throughout my entire education. Finally, my experience at NMBU has been shaped by many incredible students. These five years have been unforgettable.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and motivation

Driving is a big part of our lives, and many hours are spent in a vehicle each year. According to a study made by SSB, Norwegians traveled around 56.3 billion passenger kilometers in 2022, and have had a gradual increase for many years [10]. This substantial amount of time spent in vehicles also comes with a lot of risk involved. In 2022, there were 118 fatalities and 578 people seriously injured on Norwegian roads [11]. These statistics have almost halved in the last 15 years, and many safety advancements and precautions have been made through the years to achieve this [12]. However, most of the traffic accidents in Norway are due to human errors, which can be hard to completely vanish. These human errors were largely due to reasons such as high speed, drugs, and a lack of focus [13]. Human errors will likely continue to affect traffic, and therefore the number of fatalities may not reach zero, which is a vision Norway has [14]. In light of this, a potential alternative to driving has emerged in the last few decades as self-driving cars, which aims to make driving safer.

Self-driving cars have emerged as an exciting field, and it has experienced rapid development in the last couple of years. The industry states that self-driving cars will lead to an improvement in vehicle performance and efficiency, together with an overall improvement in mobility and quality of life [15]. Today, several research communities around the world study the field of self-driving cars, and many milestones have been reached in the last few years [16]. Since the beginning of its development, there have been various approaches to self-driving cars. Deep learning models have had a central role in self-driving cars. The rapid development within deep learning models, together with recent advancements in computational systems have further accelerated the development within self-driving cars [17].

The deep learning models used for self-driving systems must be reliable and robust [18][19]. Due to the role of deep learning models in self-driving cars, it can be beneficial to understand the performance and behaviour of these models. Also, it is well known that deep learning models require large amounts of data. Consequently, a good understanding of the deep learning models under varying visual-conditions is also needed to test the robustness of these models. Finally, collecting data to train deep learning models on self-driving car systems can be expensive and time-consuming [20], and the behaviour of these models under different data augmentation techniques is important to understand.

1.2 Thesis objectives

This study focuses on evaluating five benchmark deep learning models: VGG-16, ResNet-50, WideResNet-50-2, EfficientNetB0, and Vision Transformers. The performance of these models will be evaluated using three different datasets. These datasets are designed to represent the various conditions a self-driving car might encounter. Furthermore, they will serve as a test of the robustness of the models under varying visual conditions.

The Audi Autonomous Driving dataset (A2D2) was used as the source of images in these three datasets [21]. This means that the images from the three datasets in this thesis have been manually collected from the A2D2 dataset, annotated, and cropped. The first dataset consists of normal and augmented images, and is referred to as *NAI* dataset. The second is called *NANI* dataset and consists of normal, augmented, and mixed-class noise images. Finally, the third dataset, referred to as *NASI* dataset, includes normal, augmented, and fake images generated from a DCGAN model.

In the longer term, this study aims to contribute to a better understanding of how these deep learning models perform under varying conditions. Furthermore, the inclusion of the state-of-the-art Vision Transformer model allows for the exploration of its potential application in self-driving car tasks.

1.3 Research questions

The thesis considers the classification of traffic objects and participants through the lens of distinct deep learning models. Although extensive research has been carried out related to self-driving cars, a research gap remains to investigate the effect of selected deep learning models on data with different variations. The efficiency of original, augmented, and noisy images, alongside images generated via generative adversarial networks (GANs), is investigated. Through this exploration, the aim is to ascertain the optimal approach for improving the accuracy and robustness of traffic object and participant classification systems. The experiments conducted in this thesis are tailored and designed to answer the following research questions:

1. How do selected deep learning models perform in the domain of self-driving cars?

2. How does the performance of these deep learning models vary when using datasets consisting of normal and augmented images, as well as datasets containing normal, augmented, and noisy mixed-class data?

3. How does the performance of these deep learning models change when using images generated by DCGANs?

To enlighten the research questions above, an exploratory study was the chosen approach for this thesis. This approach allows for a flexible analysis of the various deep learning models across the different datasets.

Chapter 2

Literature Review

This chapter starts with a theoretical framework section, providing an overview of essential topics and key concepts relevant to this thesis. Following this, a *Related works* section will review relevant literature.

2.1 Theoretical framework

2.1.1 Machine learning

This subsection takes a closer look at machine learning and its key concepts, with a particular focus on supervised learning.

What is machine learning?

Machine learning is a subfield of artificial intelligence, and emerged in the second half of the 20th century. Using learning algorithms, machines could now interpret and understand data to make predictions on it, based on past knowledge [22]. This is done through a training phase, where the machine learning model is trained on a large set of data, called ”training data”. Using the knowledge and experiences the model gets from the training process, the model will try to make predictions on data that is not part of the dataset [23]. There are many ways of categorizing the machine learning field. The most traditional way to distinguish it, is with the following categories [24]: supervised learning, unsupervised learning and reinforcement learning.

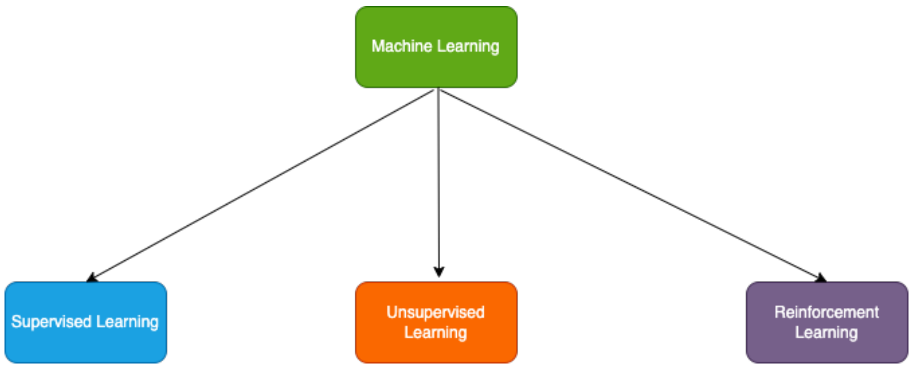


Figure 2.1: The three types of machine learning. The deep learning models in this thesis use supervised learning, whereas the DCGAN is implemented using an unsupervised learning approach. This figure was inspired by a figure in this paper [1].

This thesis will mostly use supervised learning, with unsupervised learning being implemented solely through the use of DCGAN. Both supervised and unsupervised learning will be covered here.

Supervised learning

Supervised learning is the most widely used form of machine learning. In supervised learning, a model is trained on labeled data to generate predictions on unlabeled data [25]. The machine learning model used for supervised learning tasks will be given a set of input variables and corresponding output variables [26].

Linear algebra and statistics are central to supervised learning and the broader field of machine learning. Machine learning models can be seen as a set of mathematical functions that try to map some input features into some output labels. These sets of functions can have several different architectures, such as neural networks, regression models, and convolutional neural networks. Only neural networks and convolutional neural networks are relevant to this thesis, and will be explained later [22]. The training process consist of several training loops called epochs.

Generalization can be seen as one of the primary objectives of any machine learning task. This refers to how good the machine learning model performs on unseen data [27]. Additionally, many machine learning algorithms suffer from something called overfitting and underfitting. Overfitting happens when the training accuracy of the model continues to increase, but degrades on the validation set. This typically occurs when the model learns the training data too well, including its noise and outliers, which indicates that the model is not generalizing effectively to previously unseen data [28]. On the other hand, underfitting happens when the model is not able to explain the variance in the data [29]. This can happen due to the model not training for a long enough time, the wrong choice of machine learning model, or that there is simply not enough data for the model to learn on [27].

A challenge that exists in any machine learning approach, including supervised learning, is bias. As it is known, machine learning models learn from the training data. A machine learning model can get biased if the data or the decisions taken on the data are biased. A dataset is generally described to be biased if the sampling distribution is different from the population distribution [30]. The role of bias in the datasets used in this thesis will be further discussed later in the discussion Chapter 5.

Unsupervised learning

Unsupervised learning is a type of learning where the machine receives inputs in the form of unlabeled data, and the model is left to explore and figure out meaningful information without guidance from previously known outcomes [22].

2.1.2 Artificial neural networks

Artificial neural networks, also known as neural networks, is a subcategory of supervised machine learning. ANNs have gained in popularity in the last few years, mainly due to recent advancements in computational power, and are the go-to for many tasks in the industry today [2].

Artificial neural networks consists of neurons, where each neuron can receive and process information. An artificial neuron is the basic building block of every neural network and is a simple mathematical function [31]. In an artificial neural network, the neurons are arranged into groups, called layers. A layer consists of several neurons, and the neurons in each layer are connected to the neurons in the next layer. The neurons in each layer communicate and process information with weights. These weights are coefficients that scale the input signal to a given neuron, and the weights make up the learnable parameters of a neural network. At first, these weights will be randomly initialized. The way these weights are adjusted are through an algorithm called backpropagation, which will be explained in more detail later in this section. Also, biases are added to the input of a neuron. These biases are essentially scalar values and allows the model to be more flexible and try new interpretations of the data. Like the weights, the biases are learnable parameters and are adjusted throughout the training process [2].

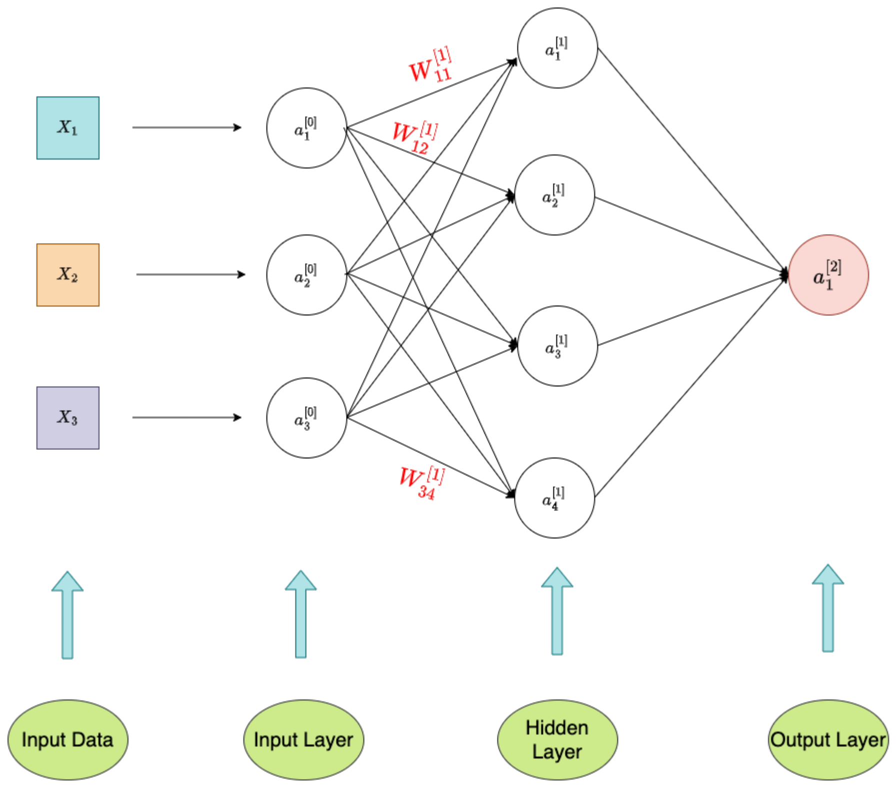


Figure 2.2: Simple artificial neural network, with an input layer, a hidden layer and an output layer. This figure is inspired by a figure in this book [2].

An artificial neural network consists of three layers: An input layer, one or more hidden layers, and an output layer. The input layer is how the input data gets into the network. Input data is fed into the input layer in the form of vectors, and the number of neurons in the input layer usually equals to the number of features in the input data. The input layer is followed by one or more hidden layers. The connections between the input layer and the hidden layers have weights [2].

Figure 2.2 shows a simple multilayered neural network, with an input layer, two hidden layers, and an output layer. The input data x_1,x_2,x_3 are fed into the input layer, which has the same amount of neurons. The hidden layers are important for the model´s ability to learn complex relationships, as the weights in these layers are the reason neural networks can model non-linear relationships in the data. The number of hidden layers are not strictly defined, and vary from problem to problem, and is often something researchers experiment with [2].

The neurons in the input layers of the network in Figure 2.2 are denoted as $a_1^{[0]}$, $a_2^{[0]}$ and $a_3^{[0]}$. This means that the superscript with a square bracket indicates which layer the neuron is located in, while the subscript represents which neuron in the layer it is. Similarly, the weights between neurons in two layers are written as $w_{ij}^{[k]}$. Here, the superscript k represent which layer the weight goes to. Meanwhile, the subscripts represents the position of the weight between the two neurons. For example, $w_{21}^{[1]}$, is the weight between neuron $a_1^{[0]}$ in the input layer and neuron $a_{12}^{[1]}$ in the hidden layer. This means that i stands for the number of the neuron in the second layer, and j stands for the number of the neuron in the hidden layer [22]. These notations will be used to explain the next concept.

Artificial neural networks make predictions through something called forward propagation. As seen in the last paragraph, the input data is fed into the networks and gets forward propagated through the network until the output layer, where a final prediction is made. Several calculations are being made throughout the neural network. The first calculation being made is the net input. This net input is calculated by taking each input value and calculating it by the corresponding weight. The result of this net input will then be passed to an activation function, before being passed to the neuron in the next layer [2]. Looking at Figure 2.2, the net input for the first neuron in the hidden layer will be given as:

$$z_1^{[1]} = w_{11}^{[1]}a_1^{[0]} + w_{21}^{[1]}a_2^{[0]} + w_{31}^{[1]}a_3^{[0]} + b_1^{[1]} \tag{2.1}$$

The resulting net input will then be passed through an activation function. An activation function will transform the net input, and the output will be the input for the next node. Activation functions will usually transform the data into a number between 0 and 1. Neurons in the same layer use the same activation function [2]. The process of calculating a net input, and passing it into an activation function will be repeated for all the neurons in each layer. Various activation functions are used in neural networks. The choice of activation function differs from problem to problem, but also within the layers of the same neural network [2].

Activation functions can be categorized into two subcategories: linear and non-linear. Linear activation functions are often used in the input layer of a neural network. The linear activation function will simply pass in the activation unchanged. As seen, the hidden layers of the neural network try to extract higher-order features from the data. Neural networks do this by introducing non-linearity using non-linear activation functions [2]. Rectified Linear Units (ReLU) is one of the most common non-linear activation functions used for the hidden layers. A neuron with the ReLU activation function will only pass a signal if the input is above a certain threshold. If the input value is below zero, the output will be zero. Today, there are several variants of the ReLU activation function, like the Leaky ReLU [2]. Figure 2.3 shows the regular ReLU function:

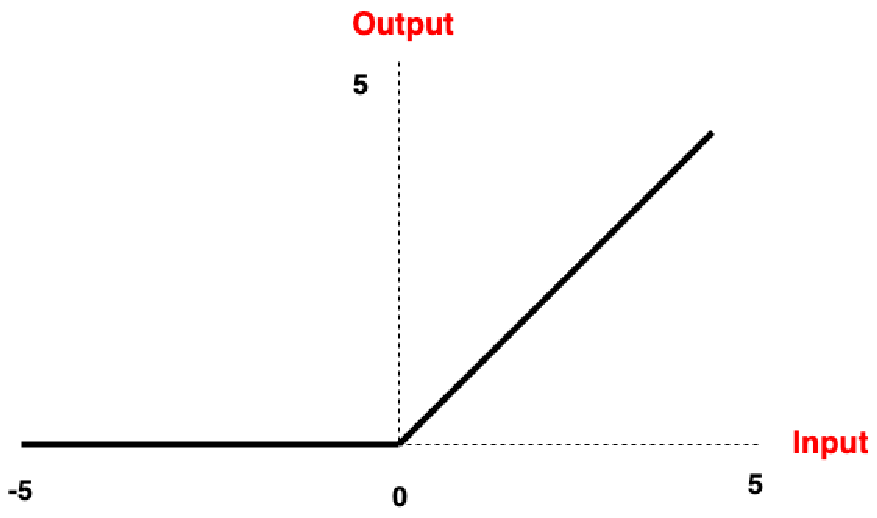


Figure 2.3: ReLU activation function. Input values below zero, will give outputs of zero. Input values above zero will remain unchanged. This figure is inspired by a figure in [2].

The calculations for net input are computed using matrix multiplications. A vector W which contains the weights, and a vector A, which contains the input values will be multiplied by each other [22]. As a result, the net input for the first neuron in the hidden layer can be written in this compact form using matrix multiplication:

$$z_1^{[1]} = W * A \tag{2.2}$$

$$W = \begin{bmatrix} w_{11}^{[1]} & w_{12}^{[1]} & w_{13}^{[1]} \\ w_{21}^{[1]} & w_{22}^{[1]} & w_{23}^{[1]} \\ w_{31}^{[1]} & w_{32}^{[1]} & w_{33}^{[1]} \\ w_{41}^{[1]} & w_{42}^{[1]} & w_{43}^{[1]} \end{bmatrix}$$

$$A = \begin{bmatrix} a_1^{[0]} \\ a_2^{[0]} \\ a_3^{[0]} \end{bmatrix}$$

After passing it into the activation function, $a_1^{[1]}$ will be the following:

$$a_1^{[1]} = f^{[2]}(z_1^{[2]}) \tag{2.3}$$

, where f is the activation function used.

Neural networks use backpropagation to adjust the weights of the model in the training process, and this is the way neural networks learn. The backpropagation process is where the neural network learns the mapping between input features and output labels. The loss function is central to this process. Loss is a measure of the difference between the predicted outputs and the actual correct outputs. A loss function will measure the performance of the model on the training data, and backpropagation is used to minimize this loss [27]. The backpropagation algorithm processes the neural network from the last layer to the first layer, and updates the weights between each neuron until the input layer is reached [2].

There are several types of loss functions available, and there is not one single loss function that is appropriate for all problems. For multi-class classification problems, a common loss function is categorical crossentropy. This computes the loss by comparing the probability distribution of the predicted output label and the true distribution of the labels [22]. The formula is shown below:

$$CategoricalCrossentropy = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{2.4}$$

Gradient descent is the most commonly used optimization algorithm in modern machine learning algorithms. An optimization algorithm will use the measured loss to find the combination of weights that yields the smallest possible loss. This is done by measuring the change in error caused by a change in the weight and trying to find the point where the loss function is lowest [2]. The gradient descent algorithm finds this by taking the derivative of the loss function to create the gradient. Using the gradient, it is able to find the direction to go for the lowest loss. The process of computing the error and adjusting the weights is repeated until a minimum value is found. A challenge with the gradient descent algorithm is when the loss function is non-linear. In these cases, the gradient of the loss function is complex, and it can have several local minimum values, which tricks the algorithm into thinking that it is the lowest point [2].

Finally, artificial neural networks often suffer from a problem called vanishing gradient, which usually occurs on very deep networks and happens during backpropagation. Backpropagation can be looked at mathematically, as an optimization of each function, in a chain of functions like the following [27]:

$$y = f_4(f_3(f_2(f_1(x)))) \tag{2.5}$$

In this context, the goal is to adjust each function based on the error observed at the output of f_4. The adjustment of parameters involves propagating the error backward through all the functions. However, each function in the chain introduces some amount of noise. The chain of functions shown above is for a smaller network, but as the network size increases, the amount of noise can become large and make the backpropagation algorithm stop working. This is called the vanishing gradient problem and will result in the model not training at all [27]. The vanishing gradient problem, as well as its proposed solution, will be addressed in the subsection dedicated to deep learning models, using the ResNet architecture.

2.1.3 Convolutional neural networks

The goal of a Convolutional Neural Network (CNN) is to extract more complex features from the data through convolutions. CNN is a type of artificial neural network specialized to handle image data [2]. The deep learning models used in this thesis are various types of CNN models, with the exception of the Vision Transformer model.

The higher-level components of a CNN consist of three groups: input layer, feature extraction layers, and classification layer(s). Like traditional ANNs, each layer serves a specific purpose. The input layer is where the image data is loaded, with image data sent into the input layer of a CNN. The final layers, known as the classification layers, are composed of one or more fully connected layers, similar to those explained in the ANN section [2]. At the end of these layers, a prediction is made, often in the form of class probabilities. The feature extraction layers are what sets CNNs apart from traditional neural networks. These layers have the responsibility to extract features from the images, starting with finding low-level features at the beginning and combining them to construct higher-level features in the later layers [2].

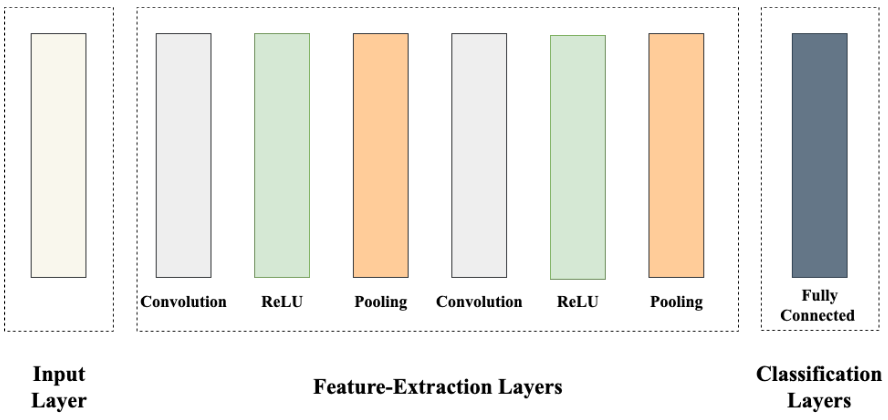


Figure 2.4: Higher level architecture of convolutional neural networks. The input layer, feature-extraction layers and classification layers are shown above. This figure was inspired by a figure here [2].

A convolutional operation is a fundamental mathematical operation used in the feature extraction layers. It combines two sets of information: the input data and a filter. Essentially, the filter, smaller in size than the input image, is systematically applied across the image, akin to a sliding window. The output, called a feature map, is derived by computing the dot product of the filter and the input region [2] . Additionally, filters are applied to every channel of the input. Filters can be viewed as feature detectors, with their values being the learnable parameters of the convolutional layer. The receptive field of a filter refers to how much of the width and height of the image it covers and is considered a hyperparameter in CNNs [2].

Convolutional neural network works well on image data largely due to two aspects: Sparse connectivity and parameter-sharing. In ANNs, the neurons in one layer are typically connected to all the neurons in the next layer, forming a fully connected neural network. However, this can be computationally expensive. In convolutional layers, each neuron is only connected to a smaller region of the preceding layers, thanks to filters whose receptive fields cover only a small patch of the image at a time. This is known as sparse connectivity [22]. Parameter sharing is a technique employed in the convolutional layer, where a filter with the same weights is used for different patches of the image. These two techniques are different from traditional fully connected neural networks and contribute to making CNNs computationally less expensive [2][22].

A convolutional layer is usually followed by an activation function. Similar to ANNs, the purpose of applying activation functions is to introduce non-linearity into the model. The ReLU activation function is the most used in convolutional layers and is applied on the feature map. ReLU will transform values that are zero or lower, to zero while values above remain unchanged [2].

A pooling layer is applied after the ReLU activation function, with max pooling being the most widely used type. A filter will go through the feature map after the ReLU activation. This filter can be of any size, but is commonly of size 2 x 2. Figure 2.5 illustrates a 2 x 2 filter with a stride of 2 on a 4 x 4 image. The stride determines how many pixels the filter moves across the input data each time. This filter slides over four patches on the feature map, capturing the highest value from each patch to generate a new feature map. These four maximum values then become the inputs for the new feature map. Another method, average pooling, computes the average value of the pixels within each patch, rather than extracting the maximum. Pooling layers aim to reduce the spatial dimensions of the input images, which can assist in addressing overfitting within the network. Finally, the filter in a pooling layer is non-learnable [2].

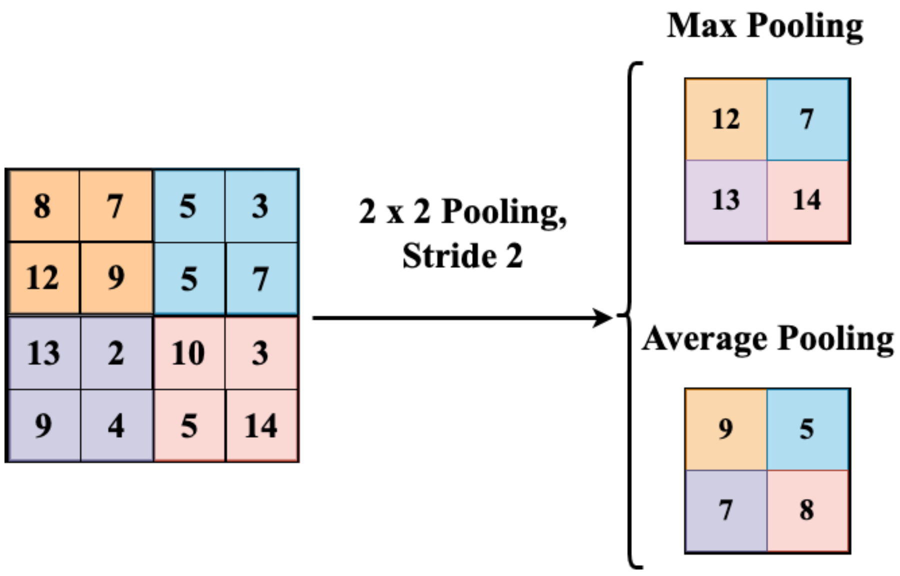


Figure 2.5: Max pooling and average pooling layer on 4 x 4 feature map. The filter is 2 x 2 with stride 2. This figure was inspired by a figure here [3].

2.1.4 Deep learning models

Deep learning is a special subfield of machine learning, that focuses on learning through successive layers of data representations. A deep learning model can have ten or even hundreds of layers [27]. These deep learning models differ from traditional multi-layer neural networks by having more neurons, more complex ways of connecting the neurons in each layer, and automatic feature extraction. Deep learning networks can model more complex problems compared to the previous ”shallow” models, with fewer layers and neurons [2]. This subsection of the thesis will elaborate on the deep learning models that are used in this thesis. The respective research papers of all the following deep learning models are recommended to read if more details about the models is needed.

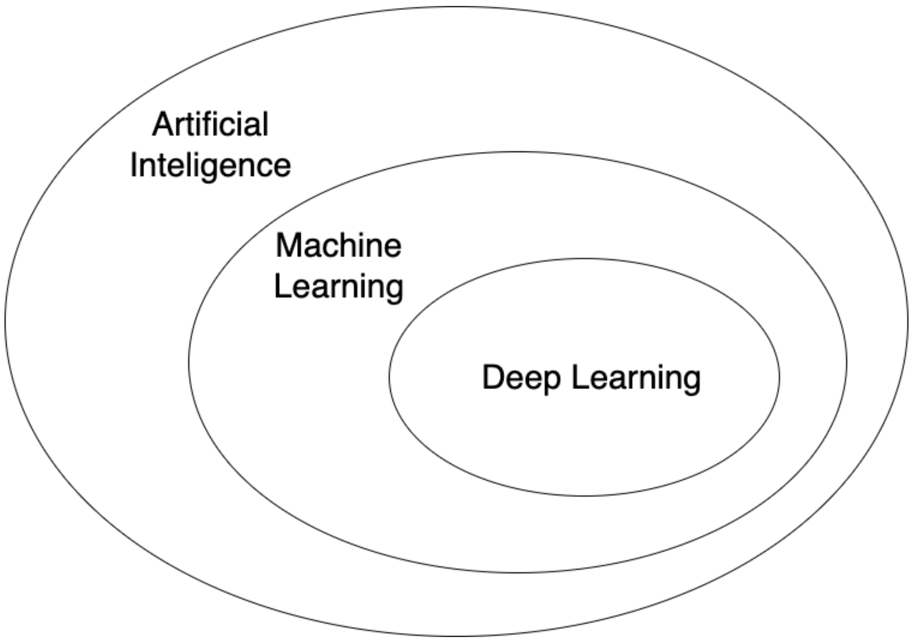


Figure 2.6: The relationship between artificial intelligence, machine learning, and deep learning. This figure was inspired by a figure here [4].

VGG-16

Overview

The VGG networks were first introduced in 2014 by Simonyan et al. through the paper, *Very Deep Convolutional Neural Networks for Large Scale Image Recognition* [32]. These networks are known for their straightforward design, and are easy to implement. Today, they are often used for transfer learning because of their ability to adapt to new datasets [33]. The original paper introduced many VGG-Networks, but only VGG-16 will be covered here, as it is the one used for this study [32].

Model architecture

The VGG-16 model has a simple architecture, compared to many other networks. As the name suggests, it consists of 16 convolutional layers, with 3 x 3 filters. In the paper, the input to the network is a fixed RGB image. The image data is then sent to the feature extraction layers. In this feature extraction layer, a stack of filters with a 3 x 3 receptive field is used. The stride is equal to 1, and same padding is used to preserve the spatial resolution after each convolution. Each convolutional layer is followed by a ReLU activation function. Lastly, a max pooling layer is applied. In total, there are 5 max pooling layers with 2 x 2 filters with a stride of 2 [32].

The feature extraction layers are followed by three fully connected layers. The first two fully connected layers have 4096 neurons, while the third and last layer of the network has 1000 neurons. All these layers lead to a total of 138 million learnable parameters for the VGG-16 model [32]. Simply put, the VGG-16 model architecture can be said to consist of repeated ”convolutional layer, convolutional layer, max pooling” blocks [27].

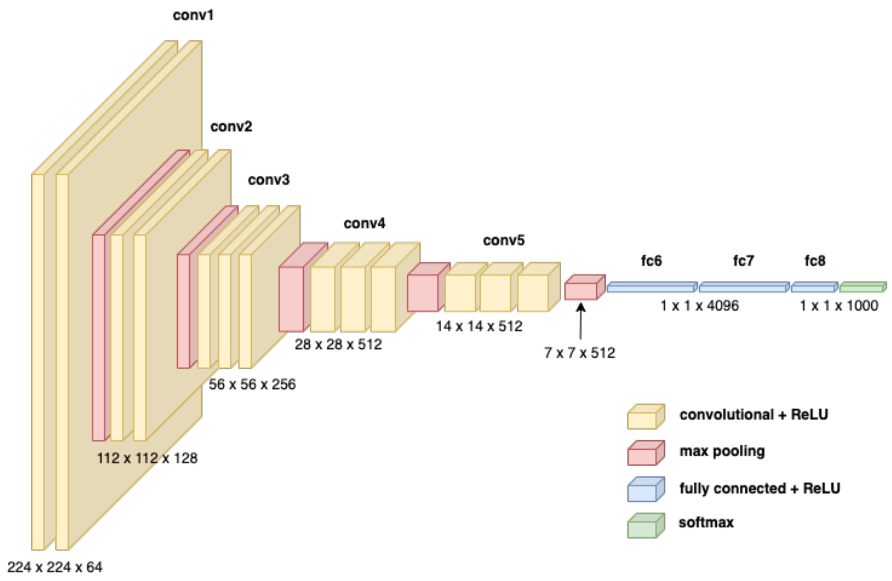


Figure 2.7: VGG-16 architecture, with 13 convolutional layers, 5 max pooling layers, 3 fully connected layers and a softmax layer in the end. This figure was inspired by [5].

Applications and performance in image classification

The VGG-16 model was initially trained and modeled for the ImageNet dataset (ILSVRC - 2012). According to the original paper, it achieved a top-1-validation-error of 27.0% and top-5-validation-error of 8.8% [32]. One disadvantage of the VGG-16 is the training time. If trained from scratch, the model will often struggle to get any initial learning process and is therefore often used pre-trained. This is largely due to the network´s depth [33].

ResNet-50

Overview

ResNet is important in the history of deep learning and introduced the concept of residual models and identity mappings. It was first introduced in the *Deep Residual Learning for Image Recognition* in 2015 by He et al [6]. The findings of this paper have made it possible to create very deep neural networks with a great accuracy [33].

Model architecture

A high number of layers can increase a model´s capacity to learn. However, as mentioned in Section 2.1.2, in practice the vanishing gradient problem limits many model architectures from getting too large. The ResNet architecture aims to solve this problem using residual connections. These connections will allow the model to skip one or more noisy layers. In this way, the gradient information will be preserved and the model can propagate backward through many layers. A residual connection is simply the addition of the input to the output of one or more layers [27]. This is shown in Figure 2.8.

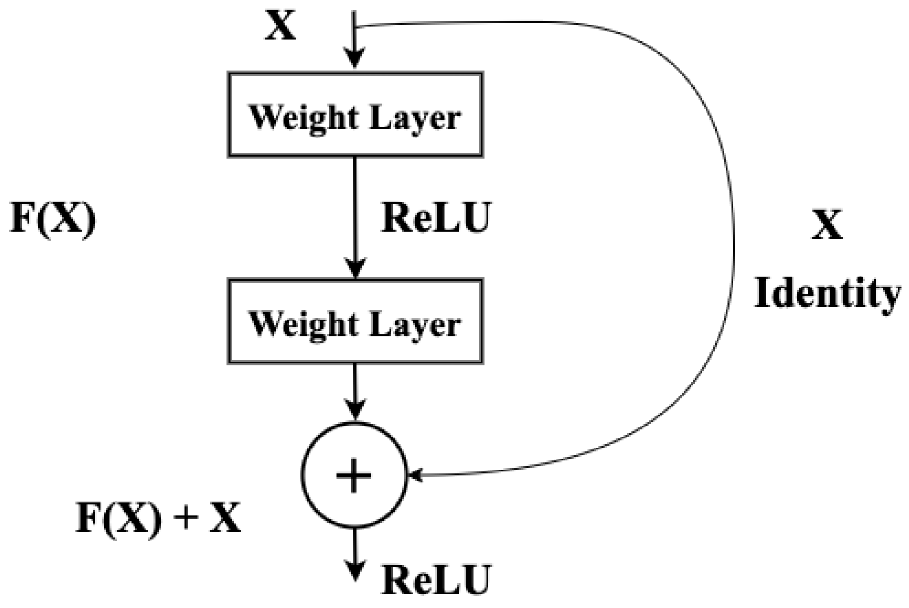


Figure 2.8: Residual connection. X is the input to the model, while $F(x)$ is the output after the weight layers. The original input and the output $F(x)$ are added together and passed through a ReLU activation function. This figure is inspired by one from the original ResNet paper [6].

Residual blocks are the building blocks of residual neural networks and are what sets them apart from other networks. First, the input data is passed through a 3 x 3 convolutional layer with a stride of 1 and a padding of 1. This results in the output having the same dimensionality as the input. This convolutional layer is then followed by a batch normalization layer, which normalizes the feature map. The data is then passed through a ReLU activation function. Then, the data is sent through another 3 x 3 convolutional layer, with the same configurations as the first one [6]. Finally, the data is passed through a batch normalization layer. The output after all these layers, will be added together with the input from before the layers using the residual connection, which was explained in the last paragraph. After this addition, the data will be passed through another ReLU activation function. All these layers define the basic residual block introduced in the ResNet-paper [6]. The ResNet block is shown in Figure 2.9.

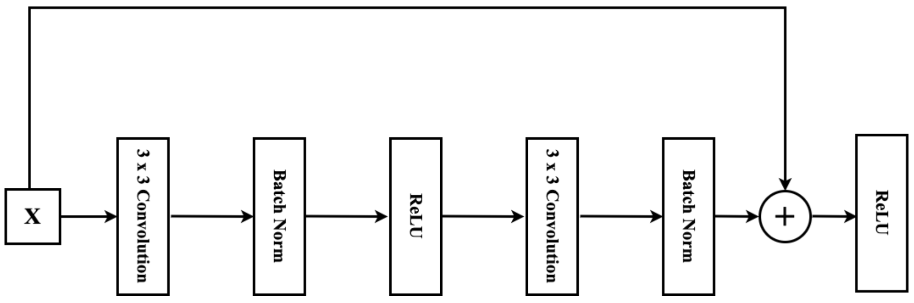


Figure 2.9: Basic residual block with a residual connection. This figure was created using the descriptions of the original paper [6].

The residual block described in the last paragraph keeps the dimensionality the same throughout the whole block. This means that there are no challenges in adding the input and the output together through the residual connections. However, it is important to reduce the dimensionality of the input data in image classification tasks, due to training time [6]. The authors of the paper proposed a 1 x 1 convolution over the input data, with a stride of 2. This configuration will result in an output feature map that halves the width and height, but doubles the number of channels. This option will allow the element-wise addition of the input and the output after the layers while reducing the size of the input data. These 1 x 1 convolutions are placed throughout the whole network, and the resulting ResNet block including the 1 x 1 convolution is called a ”bottleneck” building block [33]. The final block used for ResNet-50 is shown in the Figure 2.10:

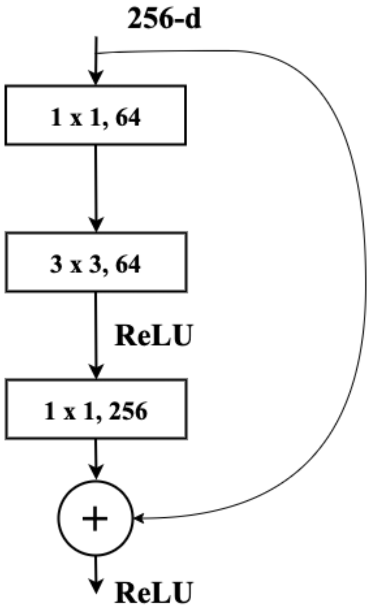


Figure 2.10: "Bottleneck" block used for ResNet-50. This illustration was inspired by a figure from the original paper [6].

Applications and performance in image classification

ResNet50 achieved a top-1-validation-error of 22.85% and a top-5-validation-error of 6.71% on the ImageNet dataset, which is better than the VGG-16 model described earlier. Various residual networks were also experimented with the CIFAR-10 dataset, and compared with plain convolutional neural networks with a comparable network depth as the residual networks. These plain networks are inspired by the VGG networks. It was shown that the plain networks suffered from the depth of the model, while the residual networks managed to overcome this and increased in accuracy as the depth increased [6]. It is also to be noted that ResNet-50 has a total of 25.6 million parameters, which is much smaller than VGG-16 [34].

WideResNet-50

Overview

The WideResNet architecture can be considered an extension of the original ResNet architecture. It was first introduced in the paper, *Wide Residual Networks*, by Zagoruyko et al. in 2016 [35]. The authors aimed to solve one of the main problems with the ResNets, which involved the residual blocks and the skip connections in these networks. The addition of residual blocks and skip connections allows the gradients to flow through two paths. The first one through the network with the residual blocks, and the second is through the residual connections. This alternative pathway allows the gradient to flow past the layers in the residual blocks. As a consequence, the gradient can go from the output of a block to the input of a block with very few adjustments in the backpropagation. The authors of this paper found that widening that network could lead to improvements in the model accuracy [35]. Many WideResNet architectures were proposed in the original paper, but the focus will be on the WideResNet-50-2, as this is the type of WideResNet model used for this thesis.

Model architecture

The wide residual networks (WideResNet) have more filters (channels) than the equivalent Residual Networks. WideResNets also consists of sequences of residual blocks, similar to those of the ResNet's described earlier [35]. The residual block of WideResNet-50-2 can be shown below in Figure 2.11, together with the regular ResNet block. Both the regular and WideResNet bottleneck first have a 1×1 convolutional layer with F number of filters. Also, the block ends with a similar 1×1 convolutional layer with F number of filters. The difference in these blocks lies in the middle layer. Both of the blocks have a 3×3 convolutional filter, but the ResNet block has $\frac{F}{4}$ number filters, while the WideResNet bottleneck has $\frac{F}{4} \times k$ number of filters where k is the widening factor. This widening factor should be larger than 1, and it increases the width of the network. The WideResNet-50-2 used in this thesis has a widening factor of 2, which means it has double the width of a ResNet-50 model [35].

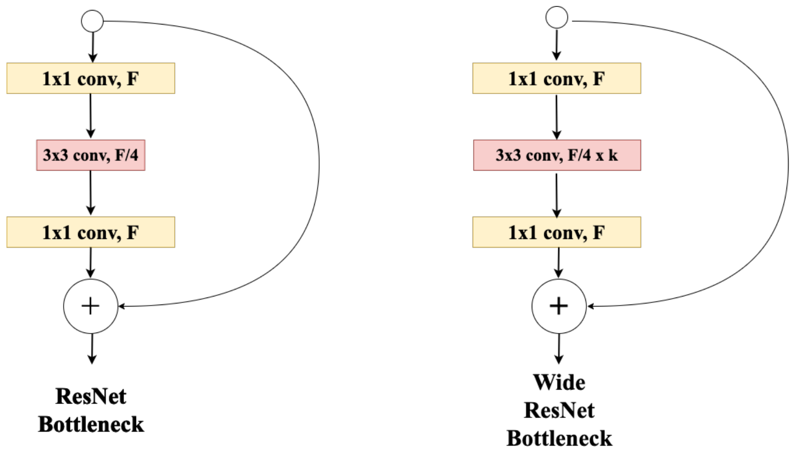


Figure 2.11: The ResNet bottleneck is shown on the left, while the suggested WideResNet bottleneck is shown on the right. This figure was inspired by [7].

Applications and performance in image classification

The authors of the WideResNet paper claimed that their networks achieved better accuracies than the equivalent ResNets, and the following performance confirms this. WideResNet-50-2 was used on ILSVRC-2012 validation dataset and achieved a top-1-validation-error of 21.9% and a top-5-validation-error of 6.03% . WideResNet-50-2 consists of 68.9 million parameters, which is much more than the ResNet-50 [35].

EfficientNetB0

Overview

The EfficientNet architectures were first introduced in the paper *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks* by Tan et al. in 2019 [36]. Adding more layers to the model; using higher-resolution input images and widening the networks with more filters have been the main ways to scale up a model. The authors of the EfficientNet paper aimed to find a way of combining these three methods to achieve the best possible accuracy. To solve this challenge, the authors of the paper proposed a compound scaling method to help find the best way to combine those three methods and scale up a model [36].

Model architecture

The compound scaling method will uniformly scale the network´s width, depth, and image resolution with a constant ratio, based on the compound coefficient, ϕ. The equation for this method is shown below.

$$
\begin{aligned}
d &= \alpha^{\phi} \\
w &= \beta^{\phi} \\
r &= \gamma^{\phi}
\end{aligned} \tag{2.6}
$$

Here, the depth coefficient is α, the width coefficient is β and the resolution coefficient is γ. Instead of manually adjusting the alpha, beta, and gamma, a compound coefficient is used as a constraint to uniformly scale the three coefficients for scaling. Using this method, a better way of scaling up the model is found, leading to better performance and efficiency [36].

A baseline model was developed by the authors, using the compound scaling method. This baseline model is called EfficientNetB0 and is one of the models used for this thesis. The coefficients for the EfficientNetB0 for the width, depth, and image resolution are as follows: α = 1.2, β = 1.1, and γ = 1.15 [36].

Applications and performance in image classification

EfficientNetB0 was shown to achieve great results for the ImageNet dataset, with fewer model parameters compared to equivalent models. The authors of the EfficientNet paper compared the EfficientNetB0 model against several other deep learning models. According to the paper, the EfficientNetB0 achieved slightly higher accuracies than the ResNet-50 model on the ImageNet dataset. The EfficientNetB0 got a top-1-accuracy of 76.3% , while the ResNet-50 got a top-1-accuracy of 76% . However, EfficientNetB0 did so with 5.3 million parameters while ResNet-50 used 26 million parameters [36].

Vision Transformer

Overview

Convolutional neural networks have been the de-facto standard for image classification tasks for many years. However, in 2021, Vision Transformers were introduced in the paper, *An Image is Worth 16 x 16 words: Transformers for Image Recognition at Scale* by researchers at Google Research Brain Team [8]. This was a new approach to image classification and solved the task without the use of convolutions. Vision Transformer (ViT) is highly inspired by transformers, which have been used for natural language processing (NLP) and use many of the same ideas. Also, it was shown that this new architecture outperformed CNNs, and did so with less computational resources [8].

Model architecture

The ViT model architecture consists of several components. It all begins by splitting the input image into non-overlapping patches, which are of the same size. Each patch is then flattened, and mapped to a higher-dimensional vector representation, which is done using a matrix operation. After that, positional embeddings are added to the newly created patch embeddings, to keep positional information about the patches. In this way, each patch has information about its position, and its position relative to the other patches [8][37].

Now, the sequence of embedded patches is fed into the transformer encoder. The components of the transformer encoder are shown in Figure 2.12. It consists of a multi-head self attention layer, multi-layer perceptron layer, and layer norm. The self-attention layer is important and will determine the attention score for each pixel. This attention score is calculated by considering the pixel, and its relationship with all the other pixels in the original input image. From there, the input will be sent into the multi-layer perceptron layer, which will add a non-linearity to the input using a gaussian error linear unit (GELU) activation function. A layer norm is applied before both of these layers, and the purpose of this is to improve the overall training time and performance of the model. Furthermore, a residual connection is also added around the multi-head attention and a MLP layer in the transformer encoder. Lastly, the ViT will make the final prediction using an MLP head, which is usually a fully connected layer [8][37].

A key difference between CNN models and ViT model lies in the way these two models see images. The ViT will look for both local and global spatial information in the earlier layers of the model, whereas the CNN models will only look for local information. In the later layers, both the ViT and the CNN models will look globally. As a result, the ViT have access to more global information in the earlier layers compared to comparable CNN models. In the paper *Do*

Vision Transformers See Like Convolutional Neural Networks, it was shown that this difference in how the two types of deep learning models represent and learn information actually leads to them acquiring different features [38]. Additionally, the differences in how these two models perceive information have been observed to affect the results and the data requirements of these deep learning architectures. Due to the local modelling behaviour of the CNN models, they are able to perform good on even smaller datasets. However, the ViT model with its use of global spatial information require more data. This is further described and shown in the next paragraph [39].

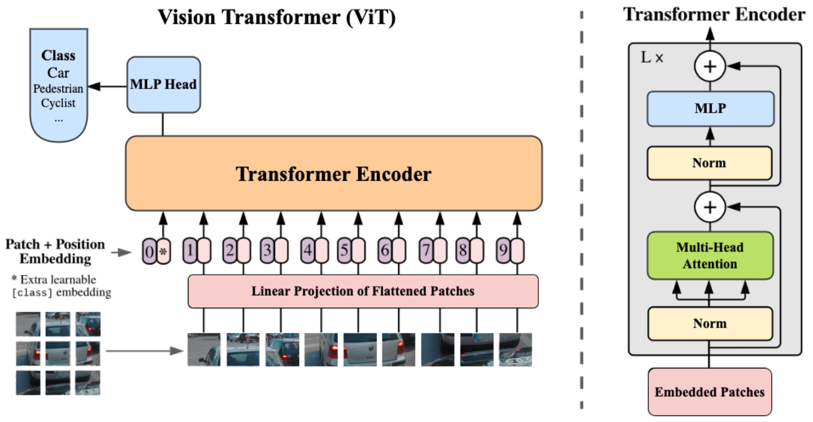


Figure 2.12: Architecture overview of Vision Transformer, as described in the original paper. This figure was inspired by a similar one from the original paper [8].

Applications and performance in image classification

In the original paper, the several ViT architectures were compared with a variant of the ResNet model. The ViT and the ResNet model were all trained on datasets of varying sizes to understand the data requirements and the performance of these models. The following datasets were used to train these models, from smallest to largest: ILSVRC - 2012 dataset, ImageNet 21K, JFT-300M dataset. After being trained on all these datasets, the models were then transferred to standard image classification benchmark datasets [8]. Through the experiments, it is shown that a ViT model pre-trained on the JFT-300M dataset outperforms the ResNet models on all benchmark tasks, while the ViT model that is pre-trained on the smaller ImageNet-21K dataset achieves good results, but is not able to outperform the state-of-the-art models on any of the tasks. Finally, the ViT models achieved these results with far less computational resources during pre-training than other models [8].

2.1.5 Transfer learning

Transfer learning is a way for models to quickly get good results on a dataset. Instead of random initialization of the weights on the dataset at hand, the network is trained on another larger dataset [28]. After the model has been trained on this larger dataset, the model is further trained on a more task-specific image dataset. Transfer learning is typically considered when the available training dataset is small, or if it is similar to the larger dataset that is used for pre-training [2].

2.1.6 Augmentation techniques

Traditional augmentation

Data augmentation is a technique used to increase the size of a training dataset and is commonly used within deep learning. As mentioned earlier, deep learning models require huge amounts of data, however acquiring such datasets is not easy. Data augmentation encompasses several techniques to increase both the variety and the size of a dataset [40].

There are many types of augmentation techniques used for images, and they can be roughly separated into two categories: photometric and geometric transformations. Photometric transformations, also known as color space transformations, refer to techniques that manipulate the pixel values of an image. Some common photometric transformation techniques include contrast, sharpness, blurring, color changes, and brightness adjustments [40]. On the other hand, geometric image transformations refer to the image augmentation techniques that manipulate the spatial arrangement of the pixels in an image. When applied, the width, height, and orientation of the image will change. Some common geometric image transformation techniques include flipping, cropping, and translations. Geometric transformations are useful to tackle positional bias in the dataset. When used together, photometric and geometric image augmentation can increase the quality of the dataset, and lead to a better-performing model [40].

The use of photometric and geometric augmentations in this study will be described in Section 3.3.2. However, it is important to note that an excessive amount of image augmentations may decrease performance [41]. This issue often arises when the augmentations cause the images to become overly correlated, which limits the model´s ability to learn a diverse set of features, and generalize effectively. This problem of augmentation is much more evident when augmentation are applied to smaller datasets [27]. This point will be discussed further in Section 5.1.3 and 5.1.4.

DCGAN

Generative Adversarial Networks (GANs) are a class of machine learning networks, that uses two separate neural networks to generate new synthetic data. These two networks, which are called discriminator and generator, are trained simultaneously. The generator is trained to generate new data, while the discriminator is trained to separate the fake data from the real data. The approach for GANs and the relationship between the generator and discriminator can be looked at as a game. The task of the generator is to generate images that are indistinguishable from the rest of the real data, while the discriminator has the task of trying to distinguish those generated images. Both the discriminator and the generator train and get better using each others feedbacks. Since the initial release of GANs, many variants of the model have been released, but the general philosophy of GANs across all variants remains the same. Also, GANs can be described as a unsupervised learning approach [42].

The desired outcome of the training loop in GANs is for the generator and discriminator to reach an equilibrium. A GANs network has reached an equilibrium if the generator can generate images that are indistinguishable from the real images, and the discriminator makes random guesses on whether an image is real or fake. This means that both the generator and the discriminator have nothing to gain from adjusting their weights and biases [42].

Deep convolutional generative adversarial networks (DCGAN) is one of the most well-known variants of GANs, and was introduced in 2016. The generator and discriminator from the original GANs network consist of neural networks. However, in DCGANs, the neural networks are replaced with a CNN. The generator in DCGANs will generate images using a convnet architecture. It will take a vector as input, and upsample it to an image. It does so using transposed convolutions. These transposed convolutions will increase the width and the height while reducing the depth. The transposed convolutions are followed by a batch norm layer and a LeakyReLU activation. This structure continues until the final layer, where a tanh activation function is applied. The use of CNNs in the discriminator is much more familiar, as it´s task is to predict an image. Therefore, the input to the discriminator is an image, and it outputs a prediction vector, similar to the CNNs explained earlier in this section [42].

There are several ways to evaluate the quality of the synthetic images created by GANs. The first and most obvious way is to simply look at the images and evaluate their quality. Furthermore, the progress of the training process in GANs can be monitored by observing the model losses of both the generator and the discriminator. However, a more statistical way of measuring the quality of the generated images is through the inception score [42]. The details of this will be elaborated in Section 2.1.7.

The training process of GANs can be complex, and there are several challenges one might encounter during this process. Some of the main problems that might occur include mode collapse, slow convergence and overgeneralization. Mode collapse happens to GANs when it starts generating very similar images. The detection and monitoring of this behaviour can be performed by visually inspecting the images. Slow convergence of the generator and discriminator loss is another big problem that affects GANs, and is largely due to computational constraints during the training process. Lastly, overgeneralization happens when GANs generate new synthetic images that has elements that should not be present in the image [42].

GANs generally require a large amount of data to perform effectively and avoid overfitting, which is a key problem when working with small datasets. More specifically, the use of smaller datasets can cause the discriminator in GANs to overfit on the training data, leading the training process to diverge. This essentially means that the discriminator becomes too good at distinguishing between real and synthetic images. A common way to tackle this problem, is by using data augmentations on the dataset the GANs trains on. However, a possible consequence of this could be that the GANs learn from the augmented images in its train data. This result in the GAN network to generate images that have features similar to the augmented images, and can be referred to as augmentation ”leakage”. This was described in the paper *Training Generative Adversarial Networks with Limited Data* by Karras et al. [43], and its role in the training process of the DCGAN model in this thesis will be discussed further in the Discussion chapter.

2.1.7 Model evaluation

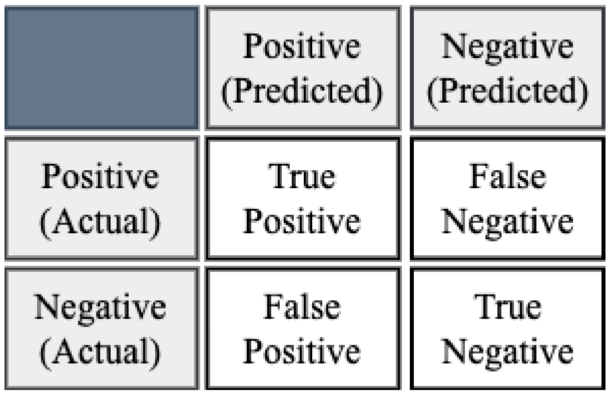
Model evaluation is an important task in machine learning. As described, a dataset is divided into subsets, which are evaluated on their own. The resulting performance indicate how well a machine learning model generalizes. There is a wide variety of model evaluation techniques available that aim to improve the generalization capabilities of machine learning models, and overall a better model [27]. The metrics used for this thesis, will be described below.

The predicted samples are generally divided into four groups, as described here [2]. These groups are as follows:

1. **True Positive(TP):** This is a positive prediction made by the model, and the label was also positive.
2. **False Positive(FP):** This is a positive prediction made by the model, but the label was negative.
3. **True Negative(TN):** This is a negative prediction made by the model, and the label was also negative.
4. **False Negative(FN):** This is a negative prediction made by the model, but the label was actually positive.

Confusion matrix

A confusion matrix is a square matrix, that visualizes the counts of True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) predictions made by the machine learning model [22]. It is widely used to understand how well the model performs at making correct predictions [2]. Figure 2.13 is a visualization of the confusion matrix.



	Positive (Predicted)	Negative (Predicted)
Positive (Actual)	True Positive	False Negative
Negative (Actual)	False Positive	True Negative

Figure 2.13: Confusion matrix with TP, FP, TN and FN.

Accuracy

Accuracy is a metric used in machine learning to measure the performance of a model. It tells how many of the model´s predictions were correct, and does so by finding a ratio between the number of correct predictions, and the total number of predictions. Using this ratio, one can answer the question of how often the model correctly predicts a sample [2]. The formula for the accuracy metric is shown below.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} = \frac{TP+TN}{TP+TN+FP+FN} \tag{2.7}$$

Inception score

Inception score (IS) is a statistical method used to evaluate the synthetic images generated with GANs. This method has been shown to closely match the human perception of what a good quality image is [42]. The IS score is calculated using a pre-trained inception model to compute the logits of the generated images. With this approach, the IS score provides insights into the diversity and quality of the generated images [44]. Consequently, the higher the IS score, the better the diversity and quality of the images. Understanding what constitutes a good IS score is useful for evaluating synthetic images based on this metric. In the paper *How good is my GAN?*, the details of the IS-score and the definition of a good IS score was discussed [44]. In one of the experiments, it was shown that the real images of the well-known CIFAR10 dataset had an IS-score of 11.33. Furthermore, a DCGAN trained to generate images from this dataset resulted in an IS score of 6.69. A SNGAN was also trained on the same dataset and achieved an IS score of 8.43, and was described to generate good quality synthetic images [44]. These scores will be used as a benchmark for comparing the IS scores of the synthetic images generated by the DCGANs in this thesis, later in the discussion section.

2.2 Review of related literature

The development of self driving cars started in the 1980s by researchers and car companies. Since then, many achievements and milestones have been reached within this field. The architecture of self-driving cars can be roughly divided into two main parts: Perception system and decision-making system. The perception system consists of many subsystems, which together will help the self driving car to estimate the state of the car and give a representation of the environment [16]. The focus of this thesis will be to investigate the effect of selected deep learning models when considering data with different variations, in traffic object and participant classification systems. The following paragraphs will take a look into some recent research within deep learning models, and GANs in the context of self driving cars.

Traffic sign detection using a CNN model

Recognizing traffic signs is an important part of the perception system of a self driving car. The classification of these traffic signs are often used with CNN-based deep learning algorithms. Farag et al. proposed a CNN model called ”WAF-LeNet” for traffic sign recognition and detection, in the paper called *Recognition of traffic signs by convolutional neural nets for self-driving vehicles* [45]. The proposed CNN model is described as an upgraded version of the popular LeNet-architecture. Their model has a depth of 15 layers, which consist of convolutional, ReLU, pooling and fully connected layers. The learning process of the ”WAF-LeNet” - model was performed using the ”German Traffic Sign Dataset - GTSRB”, and an adam optimization algorithm was also used. The result of the training process was a 100% accuracy on the training data, 96.4 % accuracy on the validation data and 94.5% accuracy on the testing data. Several other papers have proposed CNN models with higher accuracies, but the CNN model in this paper distinguishes itself from the others due to its smaller size. The preprocessing steps consisted of normalization, converting the RGB images to grey-scale, further normalization on the newly created grey-scale images and a Gaussian blur was applied for filtering noise. Finally, the training data was shuffled to avoid pattern memorization. As a test for the robustness of the model, additional images were downloaded from the web. The fully trained WAF-LeNet was tested on these new images, and only achieved an accuracy of 50%. However, once the added images were preprocessed (using cropping, centering and rotation), they were all classified correctly by the model [45].

Reliability of GANs generated data for perception systems in self-driving cars

The advancements in deep learning models is a contributing factor in recent developments within perception tasks for self-driving cars. These deep learning models require a large amount of data to generate good and reliable predictions, which is especially important for self-driving cars. However, collecting data for self-driving cars is expensive and time consuming. Moreover, a self-driving car system must be able to tackle many real life scenarios that are rare, or do not exist at all in the training data. This is known as the long tail problem. GANs have emerged as a useful tool to tackle the challenges mentioned above [46].

The paper [46] aims to explore the reliability of GANs generated data for perception systems for self-driving cars. The authors of the paper, which is called *Reliability of GAN Generated Data to Train and Validate Perception Systems for Autonomous Vehicles*, conducted several experiments. One of the experiments performed in this paper was object detection using YOLOV5 on images that are data generated by a CycleGAN. The research team created a dataset by collecting day-time and night-time images on the highway. However, only 10% of the dataset were in the night-time domain, and the rest were taken during the day-time. The goal of this experiment was to generate night-time images using GANs, and explore its impact using the predictions made by the deep learning model. A pretrained CycleGANs is used, and is further trained on the dataset from this paper, for 200 epochs. The resulting GANs, created night-time images from the original day-time images [46].

The GANs generated images were added to the training set to understand the impact of GANs. In the experiment, the researchers started with an initial dataset with real night data.

The GAN generated night-time images are added to the training set, and the impact of the added images is monitored. It was shown through the various tests that the addition of more GANs generated images in the night time training set, increased the mean average precision of the night-time test set. GAN generated images were also added to the test data, as part of the experiment for object detection in this paper. The purpose of adding the images to the test data was to explore the safety and reliability of using GANs to evaluate models before deployment. For this smaller experiment, it was shown that the mean average precision values remain in the same range, but the class-wise average precision varies a lot [46].

Overall, it was shown that adding GAN generated images to the original dataset can substantially increase the performance of deep learning models [46].

Vision Transformers in self-driving cars

Vision Transformers have revolutionized the field of computer vision, and is emerging as an alternative to CNN based models in autonomous driving as well. The unique architecture of Vision Transformer models, with their global scene understanding, together with their ability to process data parallelly makes them ideal for systems in self-driving cars. The various Vision Transformer models are praised for their versatility in computer vision tasks, especially within self-driving cars [47]. Pedestrian detection is an important feature for self-driving cars. The current pedestrian detectors used for autonomous driving systems are shown to be bad at handling small domain shifts in cross-dataset evaluations, and this was discussed in this paper [48]. The authors of this paper explained the bad performance with a limited generalization, and the deep learning models being too tailored to the training dataset. In this paper, a performance comparison between a CNN model and a transformer model was conducted. The purpose of this experiment was to evaluate how these two models compare in the ability to generalize on a new different dataset, when pre-trained on an initial dataset. It was shown that transformer models outperform CNN based networks in cross-dataset evaluation. However, when the shift in domain is large between the original dataset, and the dataset used for evaluation, and when the size of the training dataset is increased, it is shown that the CNN models perform better and are more robust to the domain shift than the transformer models. This is a finding that differs from the findings of other similar experiments in the literature. Cascade R-CNN, HRNet and Swin Transformer were used to make this comparison [48].

2.2.1 Study aim

After a review of the existing literature on deep learning models and self driving cars, it is evident that there remains a gap in knowledge regarding the impact of data with different variations on deep learning models. This thesis aims to bridge this gap by focusing on five popular deep learning models, and perform image classification on three different datasets. As this is an exploratory study, the impact of original, augmented, noisy, and GAN-generated images on deep learning models will be explored to gain a better understanding of the topics. The performance and behaviour of these models on the three datasets will be discussed and compared later in Chapter 5. Furthermore, the resistance of these models against noise will also be discovered through one of the datasets. The following chapter will detail the methodology and findings of this study.

Chapter 3

Methodology

This chapter covers the material and methods used to answer the research questions of this thesis. It begins by covering the tools and libraries used in the experiments. Then, a detailed four-stage workflow will be described. In this workflow, the implementation of the deep learning models will be presented. These models have been trained, tuned and tested using three different datasets, which have been used to compare the different models.

3.1 Tools

3.1.1 Software

The main codes of this thesis, which includes both the deep learning models and the DCGANs were executed using Google Colab Pro. Python version 3.10.12 was used with the various GPUs provided by Google Colab Pro. TensorFlow together with Keras, and PyTorch were the main libraries used to build the deep learning models and DCGANs.

3.1.2 Hardware

Image augmentations on the initially collected images, along with image upsampling using bicubic spline interpolation for DCGAN images, were performed locally on a 13-inch MacBook Pro 2020. This was carried out using an Intel Core i5 processor without any GPUs. The procedures were implemented in a Jupyter Notebook environment using Python Version 3.9.7.

The image upsampling process is further elaborated in Section 3.3.2.

3.2 A2D2 dataset

The images used to create the three datasets of this thesis, were collected using a subset of the Audi Autonomous Driving Dataset (A2D2). This is a dataset created by the German car manufacturer Audi in 2020 and was made available under the CC BY-ND 4.0 license [21]. A more detailed explanation of the dataset itself and its structure will be given in Section 3.3.1.

3.3 Workflow

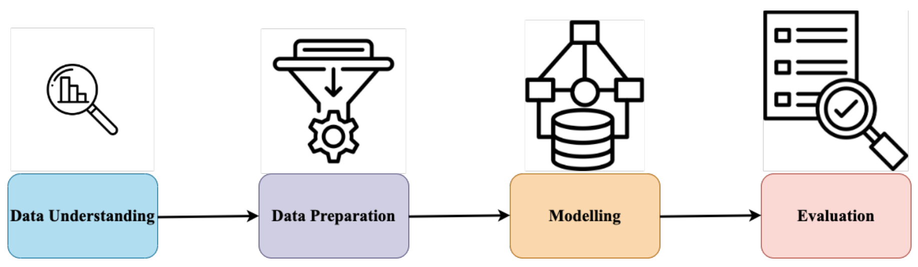


Figure 3.1: Illustration of the workflow used in this thesis. This consist of data understanding, data preparation, modelling and evaluation. This figure was inspired from a figure here [9].

A standardized workflow was followed throughout the entire project. The purpose of this workflow is to introduce structure, efficiency and reproducibility for other researchers who may wish to replicate the results of this thesis. The workflow of this thesis consist of four stages, which are: data understanding, data preparation, modelling and evaluation. These stages were selected because they are most relevant to and effectively address the research questions of this thesis. It is also important to note that each deep learning model used in this thesis was identically configured across all datasets it was used on. The following subsections will further elaborate on each of the four stages proposed in this workflow.

3.3.1 Data understanding

The first stage of this workflow is data understanding, which involves identifying, collecting, and exploring data. Previously, the A2D2 dataset was introduced as the initial source for image collection. The process of data understanding was carried out through these activities and the insights gathered will be detailed in the following paragraphs. A significant part of this stage involved reviewing and comprehending the details about the A2D2 dataset using the paper titled *A2D2: Audi Autonomous Driving Dataset*. This paper, authored by the researchers who developed the A2D2 dataset, covers essential details of the dataset [21].

The A2D2 dataset features data recorded from scenarios such as highways, country roads, and city streets in southern Germany, with images captured under various weather conditions including cloudy, rainy, and sunny. The research team at Audi created this dataset by equipping an Audi Q7 e-tron with camera sensors and LiDAR sensors, however only the images from the camera sensors were used for this thesis. This vehicle was equipped with six camera sensors, which are placed in the following areas of the vehicle: front-centre, left-centre, right-centre, side-right, side-left and rear-center. An overview of the camera setup for the vehicle used is shown in Figure 3.2. Together, these six cameras give a 360 degree surround view of the environment [21].

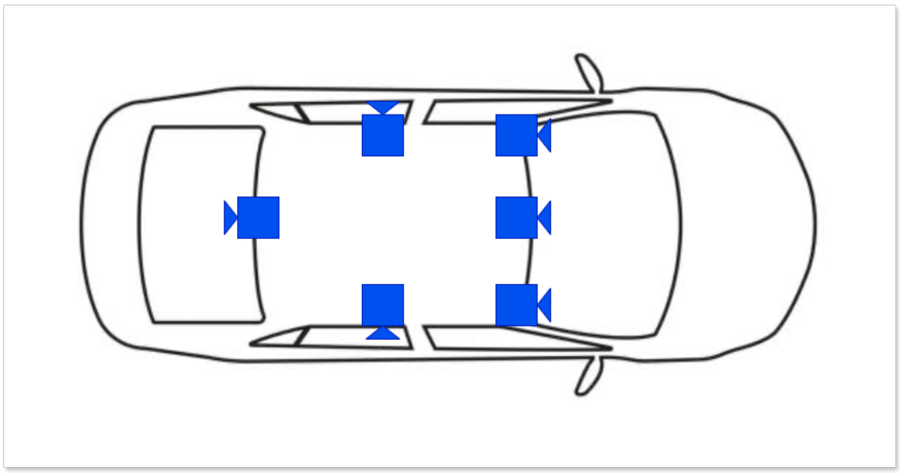


Figure 3.2: The camera setup used on to capture the data in the A2D2 dataset. The cameras are placed in the following locations on the cars: front-centre, left-centre, right-centre, side-right, side-left and rear-center.

The complete A2D2 dataset have several subsets suited for various tasks. This includes a dataset for semantic segmentation, 3D bounding boxes and point cloud segmentation. However, only the semantic segmentation dataset was used to create the three datasets in this thesis. This subset of data is divided into several data sequences taken from different scenery's. All the data sequences in the semantic segmentation dataset were used to collect the images for the datasets in this thesis. Each sequence have both camera and LiDAR data, but as mentioned, only the camera data was considered for this thesis [21].

3.3.2 Data preparation

Data preparation is a critical stage in any data-driven project, especially in deep learning and image classification. A well-structured and effective data preparation can serve as a foundation for enhanced performance of the deep learning models. The data preparation methods used in this thesis are divided into two parts: data collection and data preprocessing. The outcome of these steps is three different datasets, which are going to be used by the deep learning models. The techniques and methodologies used in these processes will be explained in the following paragraphs.

Data collection

The datasets for this thesis were manually collected and annotated. As noted earlier, the semantic segmentation sub-dataset from A2D2 served as the source of the images used to create the three datasets in this study. These three datasets were specifically tailored for the image classification task addressed in this thesis.

The images for the new datasets, suited for image classification tasks, were collected from the camera images of the A2D2 - *semantic segmentation dataset*. This new dataset consisted of six classes, which were: Car, trailer, cyclist, pedestrian, sidewalk and traffic light. These objects were chosen as classes, because of their importance in the traffic and their presence in the dataset. The images were chosen from the various data-sequence folders in the Semantic Segmentation dataset. Furthermore, images from all different cameras: Front-center, front-left, front-right, side-left, side-right, and rear-center were included to create a dataset that could lay the foundation for a model capable of generalizing well.

It´s important to properly define the class definitions before collecting the images for the six classes. During the data collection process, the following definitions were considered for each of the six classes:

Pedestrian: The *pedestrian* class is defined as images where a person is walking or is on foot in an area where vehicles operate. Images of two or more people were also included in this class.

Car: The *car* class is defined as images with one or more cars. Both moving and parked cars were considered as part of this class

Sidewalk: The *sidewalk* class is defined as images featuring a sidewalk, which is described as a path on the side of a road where pedestrians can walk.

Trailer: The *trailer* class is defined as images of large vehicles used for transportation. Trailers, both with and without the trailer compartment attached, were considered as part of this class. As with the previous classes, images featuring one or more trucks were included.

Traffic Light: The *traffic light* class is defined as a set of red, yellow, and green lights that control the movement of vehicles. Pedestrian traffic lights were also included in this class.

Cyclist: Lastly, the *cyclist* class is defined as images of a person who rides a bicycle, or walks with a bicycle. Images featuring one or more cyclists were also considered as part of this class.

After collecting all the images, these were cropped using an image editor. The motive behind this was that many of the images had several classes present in each image. By cropping, the main object in each image was mostly reduced to only one class per image. However, this was not possible for all of the images. The cropped images were of size 224 x 224, or larger. The intuition behind this was that many of the deep learning models used in this thesis had a recommended size of 224 x 224. Finally, the dataset after cropping the images had the following class distributions:

Class	Number of Images
Car	348
Cyclist	194
Pedestrian	315
Sidewalk	394
Traffic Light	257
Trailer	269

Table 3.1: Number of images per class, after image collection.

As seen, there is a class imbalance in the collected images, which is not ideal for use in deep learning tasks. Moreover, the amount of image samples for each class is not sufficient enough for large deep learning models, like the ones used in this thesis. Also, for a comprehensive comparison between the deep learning models for self-driving cars, experiments on various different datasets is desired. As a result of this, three datasets were created using the collected images described above. These three datasets are very similar to each other, and were created using data augmentation, manual cropping and synthetic images from a DCGAN. The process of creating these datasets will be further elaborated in the next few paragraphs.

Data preprocessing

Data preprocessing is an important step in machine learning. This section describes how the collected images were prepared to create three new datasets for the deep learning models. As mentioned, three methods were used to generate these datasets: data augmentation, manual cropping and GANs. These methods were essential in ensuring that the final three datasets are robust and reliable for use, and for creating datasets that would address the problem statement and research questions of this thesis.

Data augmentation

Data augmentation techniques were used to expand the size of the collected dataset, including both photometric and geometric transformations of the originally collected images. The photometric augmentation techniques included adjustments in brightness, contrast, saturation and hue. Moreover, cropping and horizontal flip was applied as geometric transformation techniques. These transformations were applied uniformly across all classes, ensuring that each class contained 600 images, thereby addressing issues of class imbalance and insufficient image data. The augmented images, along with the original images, were used to create the first dataset for this thesis. The number of normal, and augmented images for each class in the first dataset after this augmentation process is shown in Table 3.2. The images of each class were divided into train and test sets. This resulted in a per-class train set size of 480 images, and 120 for the test images. Also, this dataset acts as a base for the next two datasets in this thesis. The following two datasets will only add additional images on top of this first dataset.

Category	Normal	Augmented	Total number of images
Car	348	252	600
Cyclist	194	406	600
Pedestrian	315	285	600
Sidewalk	394	206	600
Traffic Light	257	343	600
Trailer	269	331	600

Table 3.2: Distribution of normal and augmented images in the first dataset. The 600 images for each class were further divided into 480 images for the train set and 120 images for the test set.

Manual cropping for noisy mixed-class images

Manual cropping of images was performed to introduce noise into the dataset. Earlier, the images were cropped to contain primarily a single class per image. However, to introduce noise, new images were cropped to have 2 or more classes per image. From the initial A2D2 dataset, 50 new images were collected, cropped, and added to the first dataset. The resulting dataset, which is the second dataset used in this thesis, consists of 650 images per class. These noisy images were included in the training set. After dividing the dataset into train and test sets, the training set for each class contained 530 images, while the test set contained 120 images per class. An overview of the distribution of normal, augmented and noisy images in the second dataset is shown in Table 3.3.

Class	Normal	Augmented	Noisy	Total number of images
Car	348	252	50	650
Cyclist	194	406	50	650
Pedestrian	315	285	50	650
Sidewalk	394	206	50	650
Traffic Light	257	343	50	650
Trailer	269	331	50	650

Table 3.3: Distribution of normal, augmented and noisy mixed-class images for the second dataset. The 650 images for each class were further divided into 530 images for the train set and 120 images for the test set.

GANs

A Deep Convolutional Generative Adversarial Network (DCGAN) was used to generate synthetic images for each class in this thesis. The initially collected images for each class from the A2D2 dataset were increased to 1000 images per class, using the same augmentations used to increase the first dataset. Additionally, all images were resized to 224 x 224 pixels. This approach is based on the premise that GANs typically require a substantial amount of data to generate high-quality images, as noted in Section 2.1.6. The DCGAN was trained separately for each of the six classes described earlier and underwent 1000 training epochs. Throughout the modelling process, several hyperparameters were adjusted to optimize results. An adaptive learning rate was employed for each class as an attempt to maintain balance between the discriminator and generator losses. At the end of the 1000-epoch training period, images were generated and saved. The synthetic images produced by the DCGANs were initially 64 x 64 in size. Although a DCGAN capable of producing 224 x 224 images was also tested, the quality of the images was inferior compared to the 64 x 64 images. Consequently, an upsampling technique using bicubic spline interpolation was applied to enhance the 64 x 64 resolution images up to 224 x 224, to match the rest of the images. The original dataset (first dataset), as mentioned previously, served as a baseline for creating the third dataset with the synthetic images. After the DCGAN training process, 64 images per class were generated, but only 15 images were selected for inclusion in the third dataset. These 15 images were added to the initial baseline dataset, resulting in a third dataset comprising 495 training images per class, while the test set remained at 120 images per class. An overview of the image type distribution is shown in Table 3.4 below.

Class	**Normal**	**Augmented**	**Synthetic**	**Total number of images**
Car	348	252	15	615
Cyclist	194	406	15	615
Pedestrian	315	285	15	615
Sidewalk	394	206	15	615
Traffic Light	257	343	15	615
Trailer	269	331	15	615

Table 3.4: Distribution of normal, augmented and synthetic images for the third dataset. The 615 images for each class were further divided into 495 images for the train set and 120 images for the test set.

Data summary

The three datasets described earlier in this thesis will from this point forward be referred to by shorter names. The first dataset, comprising normal and augmented images, will be known as *NAI* dataset (Normal and Augmented Image Dataset). The second dataset will be designated as *NANI* dataset (Normal, Augmented, and Noisy Image Dataset), while the third will be called *NASI* (Normal, Augmented, and Synthetic Image Dataset). Table 3.5 provides an overview of the image distribution for each dataset, along with their key characteristics. As previously mentioned, the *NAI* and *NANI* datasets have been utilized across all deep learning models, whereas the *NASI* dataset has been exclusively applied to VGG-16 and Vision Transformer models.

Dataset	Train (Images per class)	Test (Images per class)	Key characteristic
NAI dataset	480	120	Normal + augmented
NANI dataset	530	120	Normal, augmented, and noisy mixed class
NASI dataset	495	120	Normal, augmented and synthetic

Table 3.5: Overview of all three datasets used in this study. The train and test sizes for all datasets are also shown, together with the key characteristics for each dataset.

3.3.3 Modelling

Modelling is an important step in deep learning and represents the third stage of this workflow. The goal of this stage is to focus on the deep learning models used in this thesis and solve the given research problem. This subsection will go through the building process of the models, and how they were optimized and fine-tuned to reach the best possible performance. The modelling stage is crucial, as this is where the models are trained to extract information from the data.

Choice of deep learning model

Having a variety of different deep learning models to solve the research problem, is important for the comparison of the different models. VGG-16, ResNet-50, WideResNet-50, EfficientNetB0 and ViT were all used on the three datasets described earlier. These models were chosen for this task, due to their high performance in many benchmark tests as described in Section 2.1.4. All of the deep learning models can be categorized into convolutional neural networks, except the ViT model. This model was used in the thesis to compare the established CNNs to a state-of-the-art model in computer vision. The ViT model was compared against the best-performing CNN model across all three datasets, which was the VGG-16 model.

Normalization and resizing

All input data was normalized before being processed by the various deep learning models used in this thesis. Normalization was performed before the data was fed into the networks. RGB images have pixel values ranging from 0 to 255. Generally, neural networks struggle to handle input data with very large values, which can hinder the model´s ability to converge. To address this, the RGB pixel values are usually re-scaled, and this is a normalization technique used for image data [27]. The resulting pixels are adjusted to a range between 0 and 1.

Additionally, the images in the three datasets varied in size. Consequently, images in all datasets were resized to 224 x 224 before being used by the deep learning models. This standardization was crucial, providing consistent input across all models and facilitating more fair and accurate comparisons between them.

Model customization

Transfer learning was used for all deep learning models in this thesis. Although the size of the input data was increased using several techniques, the three datasets remain relatively small in the context of deep learning. Having a small dataset can lead to challenges in learning features, as the model may overfit [49]. As mentioned earlier in Section 2.1.5, transfer learning is an effective way to address some of the challenges involving smaller datasets. Therefore, it was used for all models, utilizing both Pytorch's and TensorFlow's implementations of this technique[50][51]. In this study, VGG-16, ResNet-50, EfficientNetB0 and the Vision Transformer model were executed using Keras, while WideResNet-50-2 was implemented using PyTorch.

The implementation of transfer learning for the deep learning models followed a general procedure that can be summarized in the following steps:

1. **Model initialization:** A pre-trained model is loaded. Here, the model is trained on the ImageNet dataset.

2. **Freezing layers:** The weights in the pre-trained model have been carefully trained for many epochs on the ImageNet dataset. If training begins with the entire network, it can disrupt these learned weights. To prevent this, all or part of the model is frozen, and training is focused only on the last few layers [52].

3. **Model modifications:** The original pre-trained model is trained on the ImageNet dataset, which consists of 1000 classes. Therefore, the original final layer(s) are replaced with one or more custom layers.

The primary variations in the transfer learning approaches for the deep learning models involve the number of layers frozen, the number of final layers removed, and the structure of the newly added output layers. Numerous adjustments were made to the transfer learning procedures of each model. These modifications included changes to the number of layers to be unfrozen, as well as the structure of the final few layers. Finally, it is important to mention that all the deep learning models in this thesis were trained for 100 epochs.

Optimization

Adjusting relevant parameters can enhance the model's performance. The model hyperparameters were primarily adjusted manually, using observations and the behavior of the models as indicators for the necessary adjustments. Additionally, a learning rate scheduler was experimented with for all the models and implemented if it resulted in increased performance. The purpose of the learning rate scheduler is to optimize the learning rate throughout all the epochs of the model. Only ResNet-50 and EfficientNetB0 benefited from the learning rate scheduler, and consequently, they were the only models to have it included in the final code. Moreover, categorical cross entropy was the loss function used for all of the models, and adam was used as an optimizer. However, the ViT model used the slightly different adamW optimizer, which yielded better performance compared to the regular adam optimizer.

3.3.4 Evaluation

Model evaluation is the final stage of this workflow and is performed using several techniques. This step is essential, as it allows for the conclusions to be drawn from the research questions through the evaluations and enables the comparison of different models.

Performance metric for deep learning models

The deep learning model´s ability to predict the images in the train and test set were both measured using a performance metric. The performance metric used to evaluate the deep learning models in this thesis was accuracy, and was described elaborately in Section 2.1.7.

DCGAN evaluation

The DCGAN used in this study was evaluated using several methods. First, the synthetic images generated at the end of the training process were visually assessed. Additionally, model loss plots were created to monitor the performance of the generator and the discriminator across the epochs. Finally, the inception score of the model was calculated, and an inception score plot was generated to statistically evaluate the diversity and quality of the images. The details of these evaluation techniques for GAN were covered earlier in Section 2.1.7 and 2.1.6.

Confusion matrix

Finally, a confusion matrix was generated for each deep learning model, after the training process. It was used as a tool to visualize the deep learning model´s performance, and to understand the types of errors the models make. These matrices will be shown in the next chapter.

Chapter 4

Results

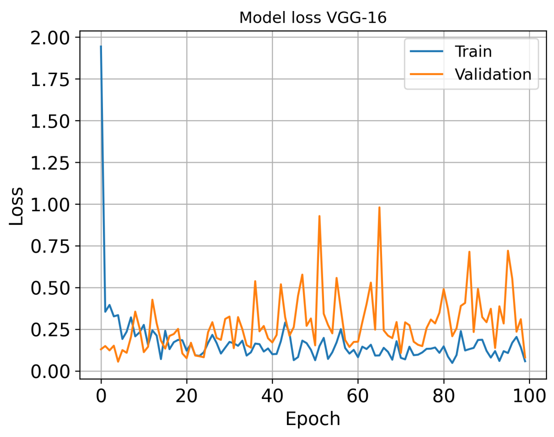
This chapter presents the results of the experiments conducted in thesis. It includes outcomes from the deep learning models on the datasets they were used on, as well as the performance of the DCGAN. The goal of this chapter is to provide the necessary information to answer the research questions introduced in Chapter 1. It should be noted that the configuration of the deep learning models remained consistent across the datasets; the same hyperparameters and configurations were used regardless of the dataset each model was applied to.

4.1 NAI dataset

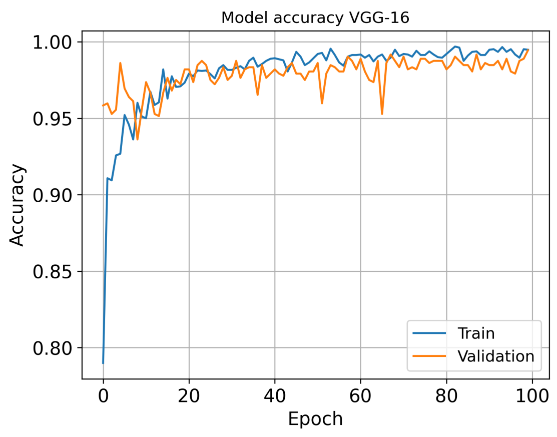
This section will present the results obtained from the deep learning models trained on the *NAI* dataset (dataset with normal and augmented images). Each model's performance will be visualized through accuracy plots, loss plots, and confusion matrices. Finally, the performance of each deep learning model on this dataset will be visualized in a single plot.

4.1.1 VGG-16

VGG-16 model was implemented as a pre-trained model, and further trained using the dataset with only augmented images, which was described in the previous chapter. When trained for 100 epochs, the VGG-16 network achieved a best validation accuracy of 99.44% on the last epoch, and the corresponding training accuracy equaled to 99.48% . The model accuracy and model loss plots are shown in Figure 4.1a and 4.1b.



(a) Model loss



(b) Model accuracy

Figure 4.1: Model loss and accuracy plots for VGG-16 model, for *NAI* dataset.

Additionally, a confusion matrix was created after the end of the training process, and showcases the types of predictions the model makes on the validation set, after training the model for 100 epochs. As one can see from the confusion matrix, four images were misclassified in the validation set.

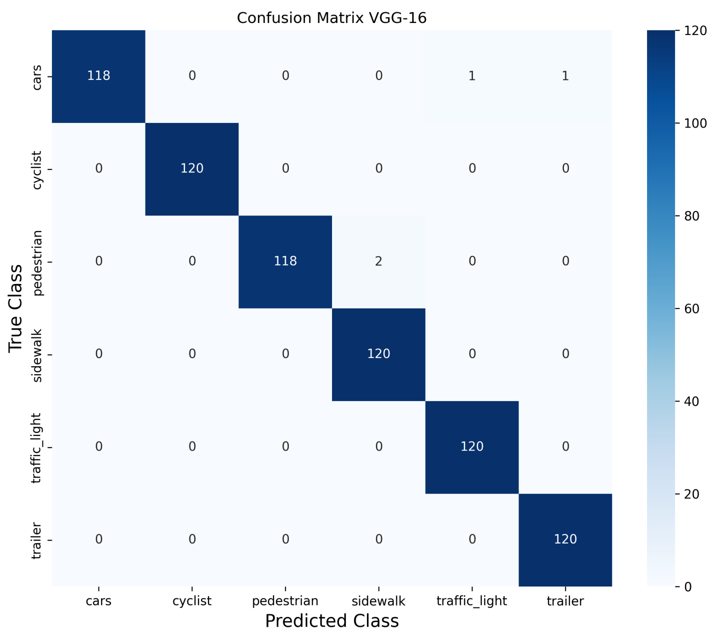
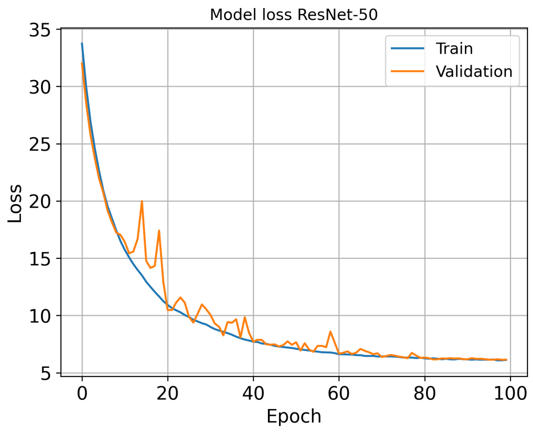


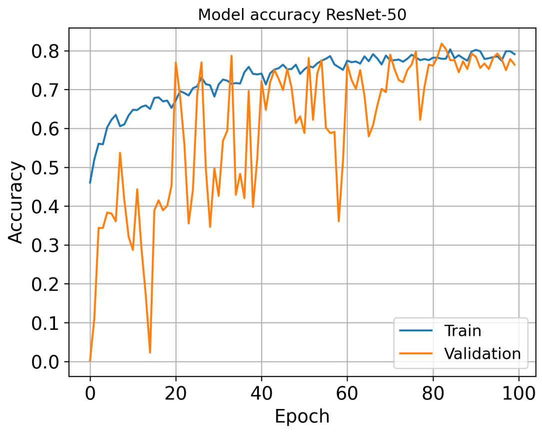
Figure 4.2: Confusion matrix for VGG-16 model, on the validation set of the *NAI* dataset.

4.1.2 ResNet-50

ResNet-50 model was implemented as a pre-trained model and further trained on the *NAI* dataset. When trained for 100 epochs, the best validation accuracy was 81.82%, and this was achieved on the 83$^{\text{rd}}$ epoch. The corresponding training accuracy for this epoch was 77.92%. The model accuracy and model loss plots are shown in Figure 4.3a and 4.3b.



(a) Model loss



(b) Model accuracy

Figure 4.3: Model loss and accuracy plots for ResNet-50 model, for *NAI* dataset.

The confusion matrix for this ResNet-50 model was also computed after the training process. This is shown in the figure below. From the confusion matrix, one can see that the ResNet-50 model misclassified 108 images on the dataset with only augmented images.

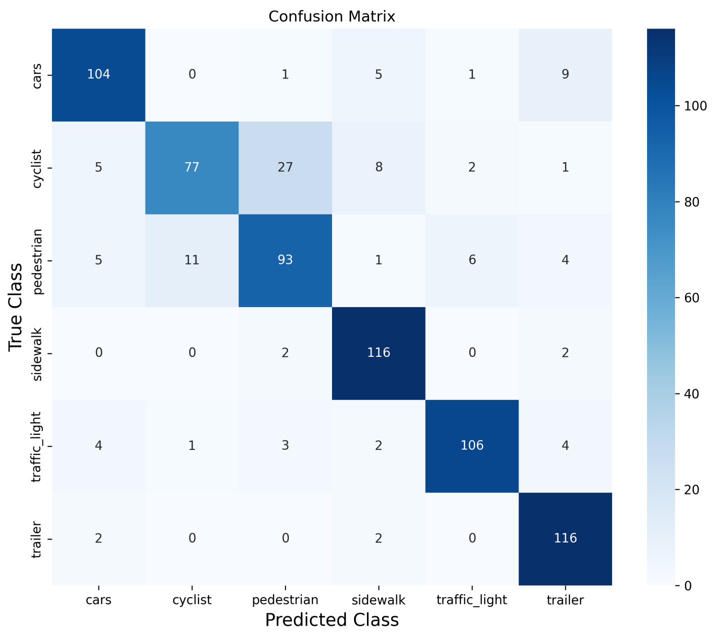
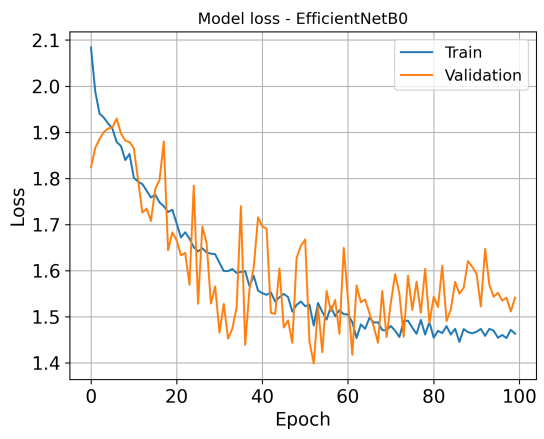


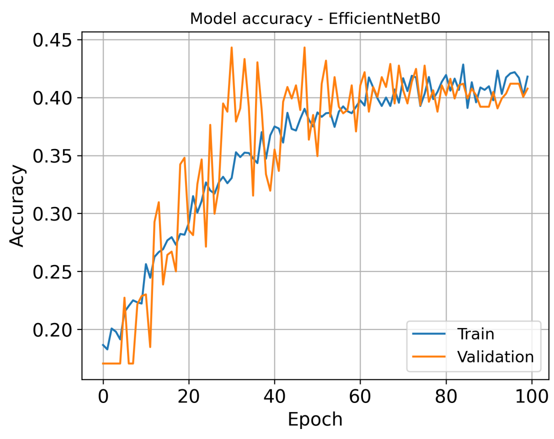
Figure 4.4: Confusion matrix for ResNet-50, on the validation set of the *NAI* dataset.

4.1.3 EfficientNetB0

EfficientNetB0 model was implemented as a pre-trained model, and further trained using the dataset with only augmented images, which was described in the previous chapter. When trained for 100 epochs, the highest validation accuracy of the model was 44.32%, which was achieved on the 48th epoch. The corresponding training accuracy was 39.03%.



(a) Model loss



(b) Model accuracy

Figure 4.5: Model loss and accuracy plots for EfficientNetB0 model, for *NAI* dataset.

The confusion matrix for the EfficientNetB0 was computed in a similar way as the previous models. This is shown in the Figure 4.6. As one can interpret from it, the EfficientNetB0 model misclassified 424 images, which means that the model misclassified more than half of the images. It can also be seen that the model classifies many of the samples as *sidewalk* class.

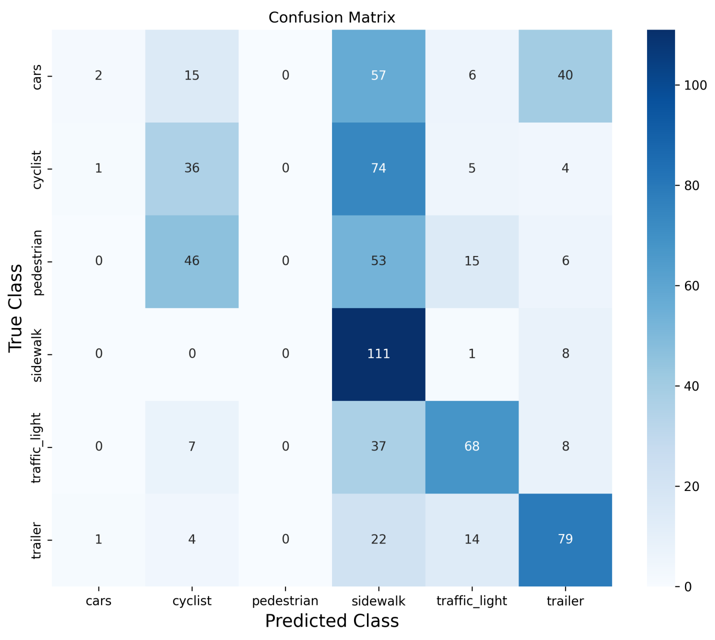
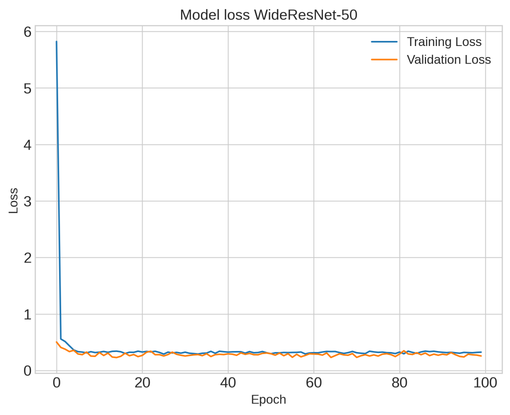


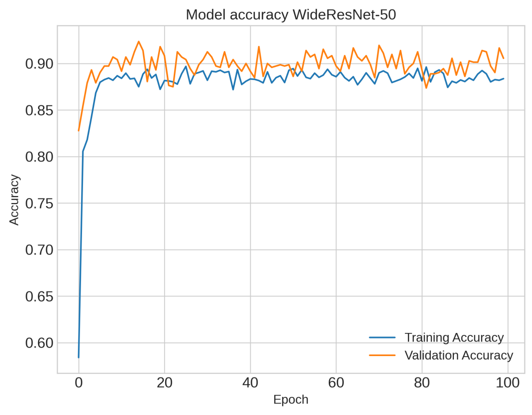
Figure 4.6: Confusion matrix for EfficientNetB0 model, on the validation set of the *NAI* dataset.

4.1.4 WideResNet-50-2

WideResNet-50-2 model was implemented as a pre-trained model and further trained using the *NAI* dataset. It was trained for 100 epochs, and the highest validation accuracy, 92.36%, was achieved at the 14th epoch, with the corresponding training accuracy at 87.50%. The model accuracy and loss plots below shows that the model reaches convergence very early in the training process. After this point, both the validation and training accuracies, as well as the loss, remain mostly unchanged throughout the remainder of the training. This behaviour will be further discussed in the next chapter.



(a) Model loss



(b) Model accuracy

Figure 4.7: Model Loss and Accuracy plots for WideResNet-50-2 model, for *NAI* dataset.

The confusion matrix for the WideResNet-50-2 network was computed similarly to the other models in this thesis. A total of 68 images were misclassified by the model.

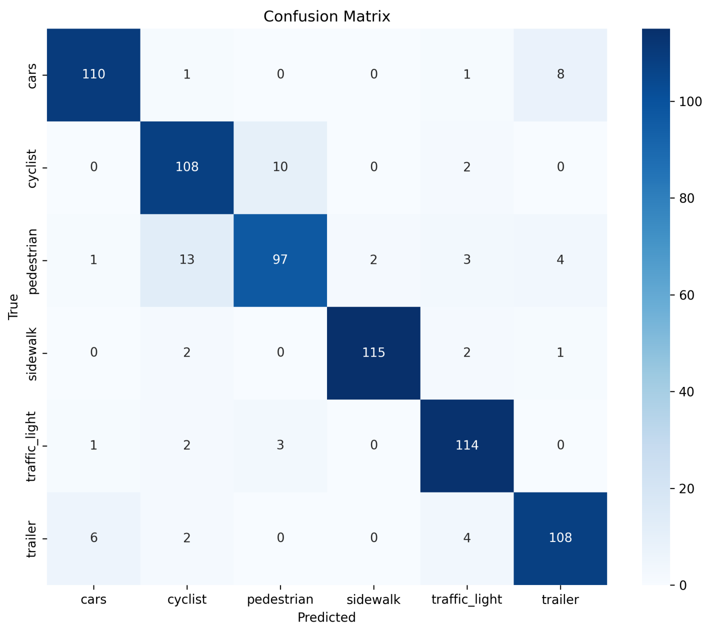
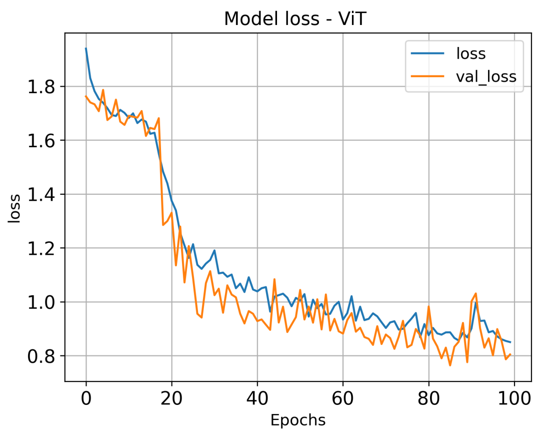


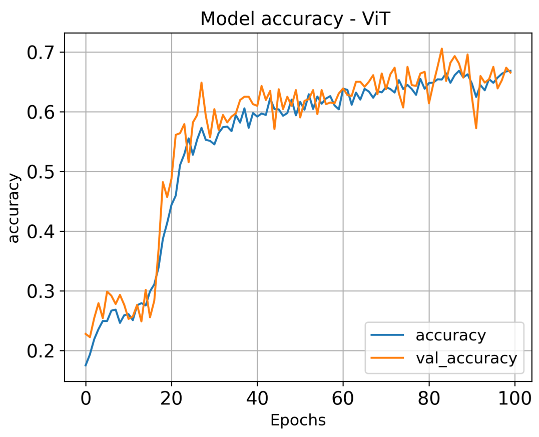
Figure 4.8: Confusion matrix for WideResNet-50-2 model, on the validation set of the *NAI* dataset.

4.1.5 Vision Transformer

Vision Transformer was implemented in this thesis as a pre-trained model and further trained on the *NAI* dataset, similar to the other models. It was trained for 100 epochs, achieving its best validation accuracy of 70.56% at the 84th epoch. The corresponding training accuracy at this epoch was 65.28%.



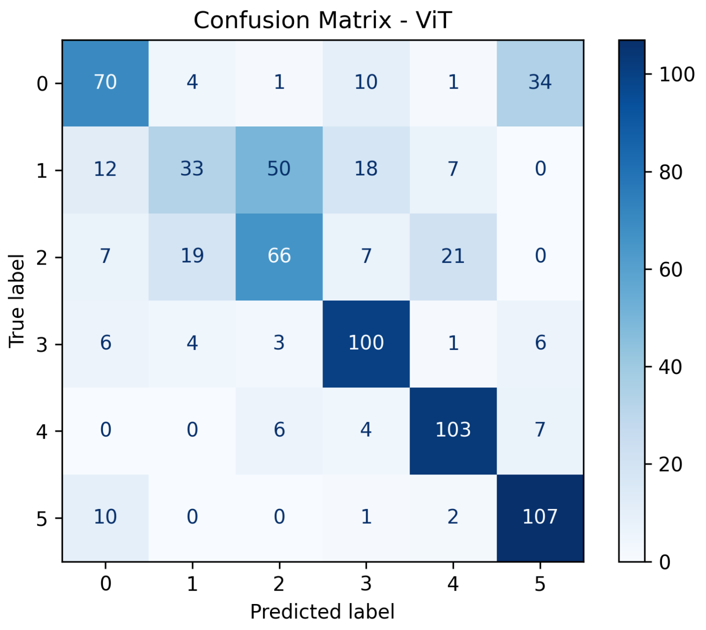
(a) Model loss



(b) Model accuracy

Figure 4.9: Model loss and accuracy plots for ViT model, for the *NAI* dataset.

Similar to the CNN models above, a confusion matrix was computed for the Vision Transformer model. Using this matrix, it can be seen that 241 images were misclassified by the model.



True \ Predicted	0	1	2	3	4	5
0	70	4	1	10	1	34
1	12	33	50	18	7	0
2	7	19	66	7	21	0
3	6	4	3	100	1	6
4	0	0	6	4	103	7
5	10	0	0	1	2	107

Figure 4.10: Confusion matrix for ViT model, on the validation set of dataset with *NAI* dataset.

4.1.6 Comparison of model performance

All five deep learning models were trained and evaluated on the *NAI* dataset. The validation accuracies of the models are visualized in Figure 4.11. From this, it is clear that the VGG-16 and WideResNet-50 are the two best performing models.

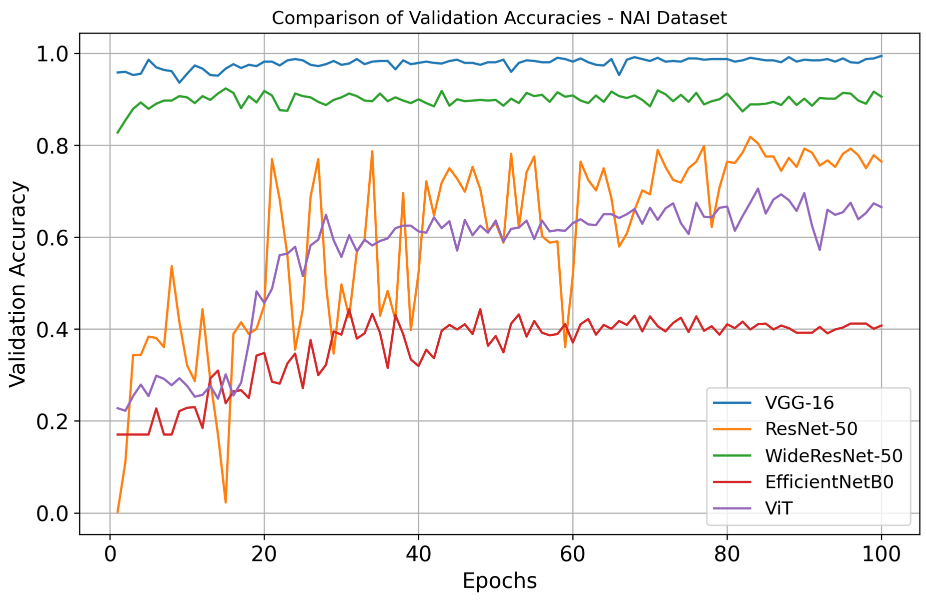


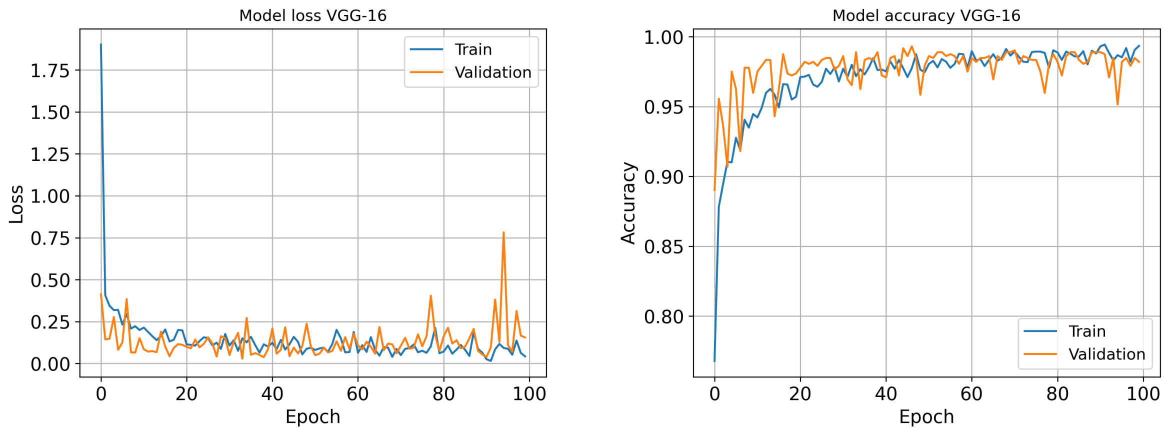
Figure 4.11: Validation accuracy of the deep learning models on *NAI* dataset.

4.2 NANI dataset

This section will describe the results from the deep learning models, using the *NANI* dataset (normal, augmented and noisy images). Each model´s results will be described using model accuracy plots, model loss plots and confusion matrices. At the end, the validation accuracies of these models will be compared in a single plot.

4.2.1 VGG-16

As mentioned earlier, the setup for the VGG-16 model for this dataset was identical to the one trained on the first dataset. This model was trained for 100 epochs on this dataset, and the best validation accuracy of this model was 99.31%, and this was achieved on the 47th epoch. The training accuracy on this epoch was 97.74%. An overview of the model accuracy and model loss over the 100 epochs are shown in Figure 4.12a and 4.12b.



(a) Model loss (b) Model accuracy

Figure 4.12: Model loss and accuracy plots for VGG-16 model, for *NANI* dataset.

A confusion matrix was also added for this model and shows the types of predictions that were made by the VGG-16 model on the validation dataset. The model misclassified 13 image samples.

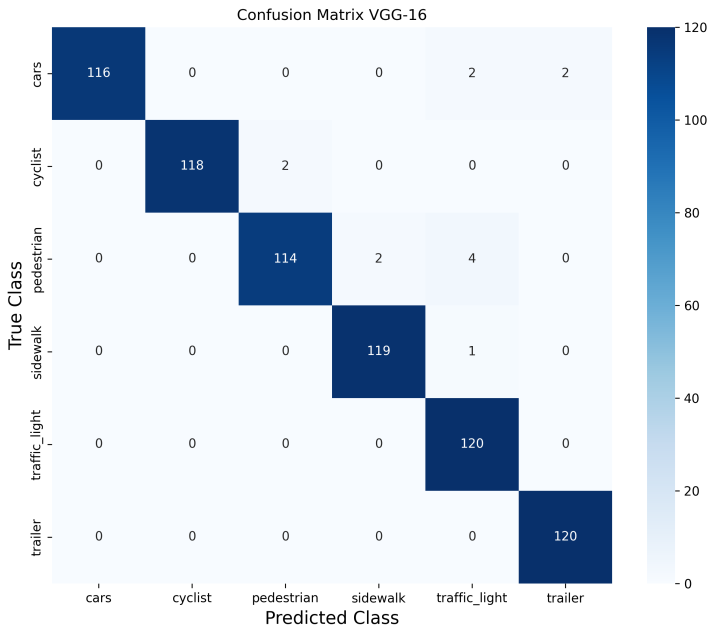
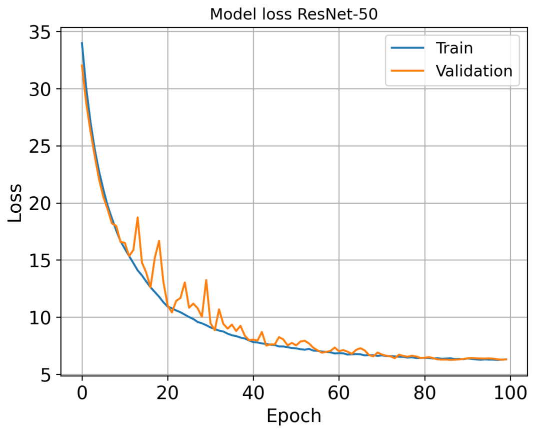


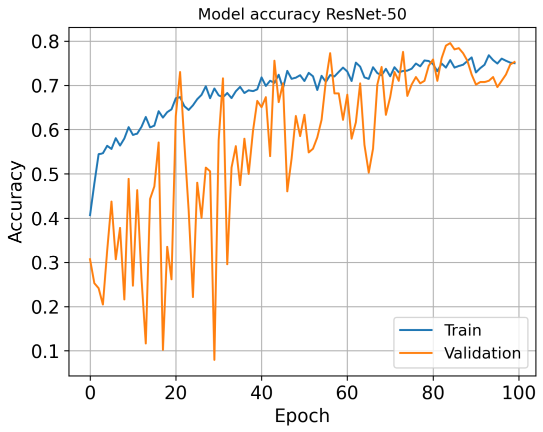
Figure 4.13: Confusion matrix for VGG-16 model, on the validation set of the *NANI* dataset.

4.2.2 ResNet-50

This model was trained for 100 epochs on this dataset, and the best validation accuracy of this model was 79.55%, and this was achieved on the 85th epoch. The training accuracy on this epoch was 75.70%. An overview of the model accuracy and model loss over the 100 epochs are shown in 4.14a and 4.14b. Also, the model loss plot shows that the loss is high. This will be discussed in the next chapter.



(a) Model loss



(b) Model accuracy

Figure 4.14: Model loss and accuracy plots for ResNet-50 model, for *NANI* dataset.

The confusion matrix for this model is shown below. One can see that the model misclassified 117 image samples on the validation set after the training process ended. The confusion matrix corresponding to this model is shown in Figure 4.15.

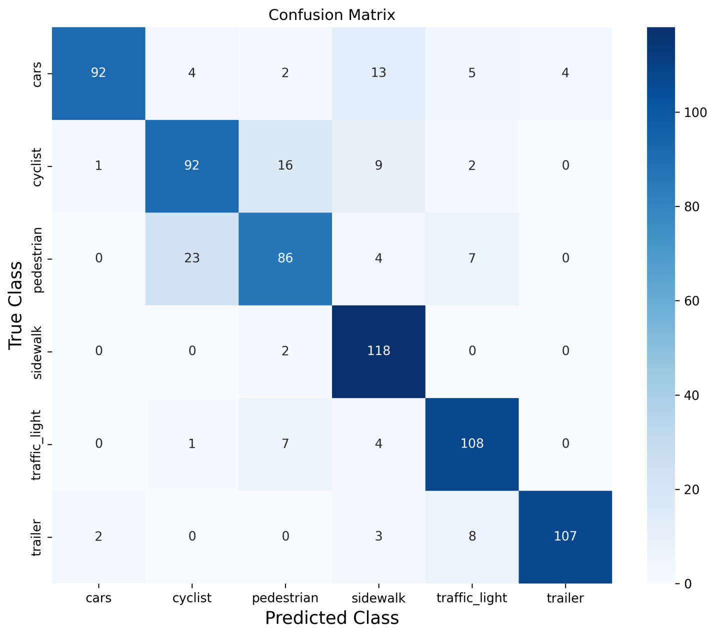
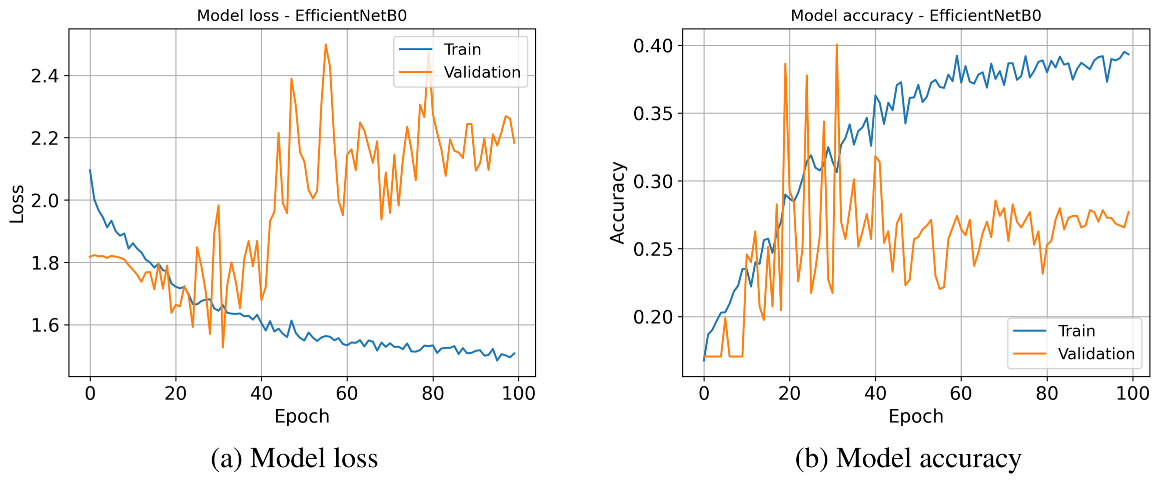


Figure 4.15: Confusion matrix for ResNet-50 model, on the validation set of the *NANI* dataset.

4.2.3 EfficientNetB0

The EfficientNetB0 model was also trained for 100 epochs on the *NANI* dataset. After training for 100 epochs, it achieved the best validation accuracy of 40.06% on the 32nd epoch. The corresponding training accuracy on this epoch was substantially lower, at 30.63%. The model accuracy and model loss plot of this model on this dataset is shown in Figure 4.16a and 4.16b.



(a) Model loss (b) Model accuracy

Figure 4.16: Model loss and accuracy plots for EfficientNetB0 model, for *NANI* dataset.

A confusion matrix was generated after the training process of this model. Again, it can be seen from the confusion matrix that the EfficientNetB0 model classifies many of the image samples as the *sidewalk* class.

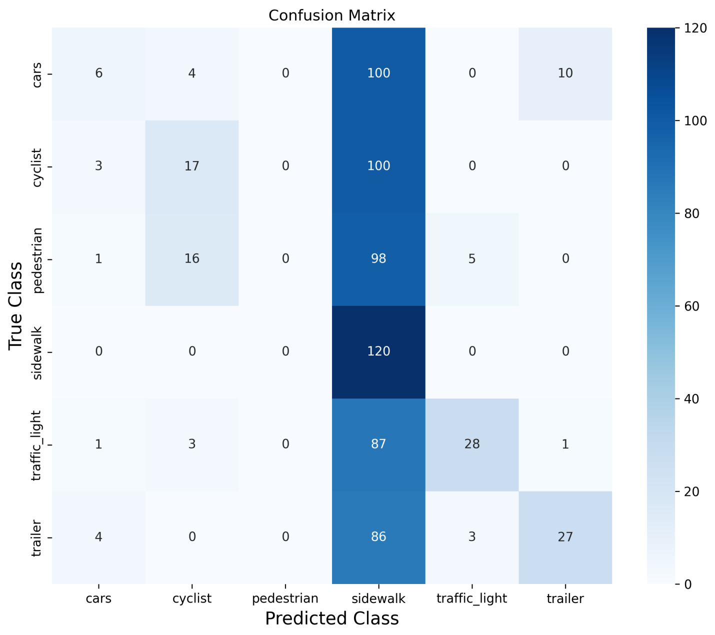
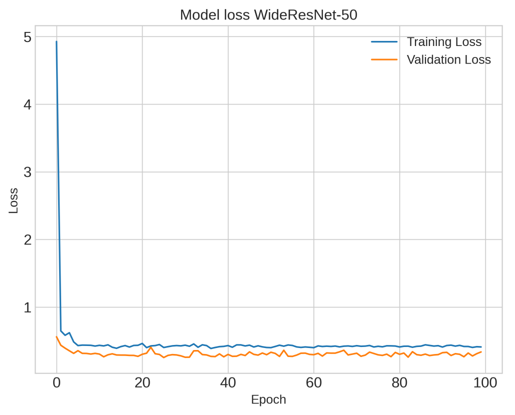


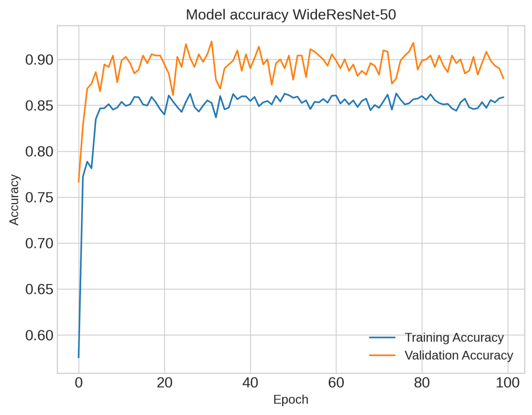
Figure 4.17: Confusion matrix for EfficientNetB0 model, on the validation set of the *NANI* dataset.

4.2.4 WideResNet-50

The pre-trained WideResNet-50 model was further trained on this dataset for 100 epochs, and achieved a best validation accuracy of 91.94% on the 31st epoch. Here, the corresponding training accuracy of the model was slightly lower at 85.25%. Figure 4.18a and 4.18b gives an overview of the model accuracy and model loss over the 100 epochs of training.



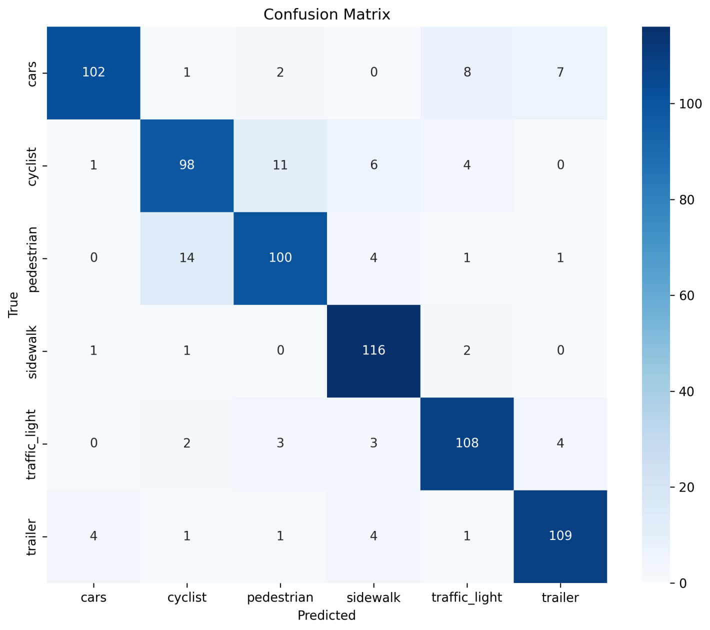
(a) Model loss



(b) Model accuracy

Figure 4.18: Model loss and accuracy plots for WideResNet-50-2 model, for *NANI* dataset.

The generated confusion matrix for this WideResNet-50-2 model is shown below. By looking at the matrix, 87 of the 720 images in the validation set were misclassified by the model.

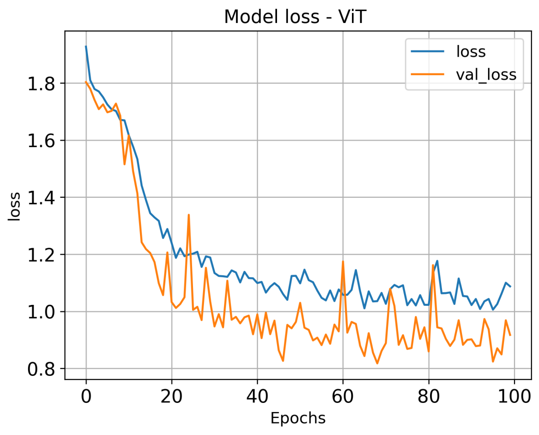


True \ Predicted	cars	cyclist	pedestrian	sidewalk	traffic_light	trailer
cars	102	1	2	0	8	7
cyclist	1	98	11	6	4	0
pedestrian	0	14	100	4	1	1
sidewalk	1	1	0	116	2	0
traffic_light	0	2	3	3	108	4
trailer	4	1	1	4	1	109

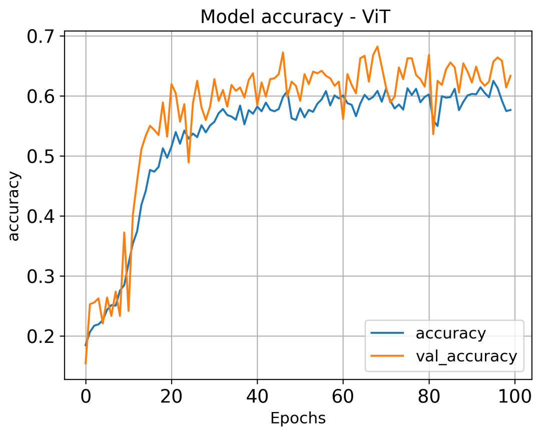
Figure 4.19: Confusion matrix for WideResNet-50-2 model, on the validation set of the *NANI* dataset.

4.2.5 Vision Transformer

The Vision Transformer model achieved a best validation accuracy of 68.19% on the 69$^{\text{th}}$ epoch in the training process. The corresponding training accuracy on this epoch was 60.82%. The model´s progress throughout the training process can be seen in the accuracy and loss plots in Figure 4.20a and 4.20b.



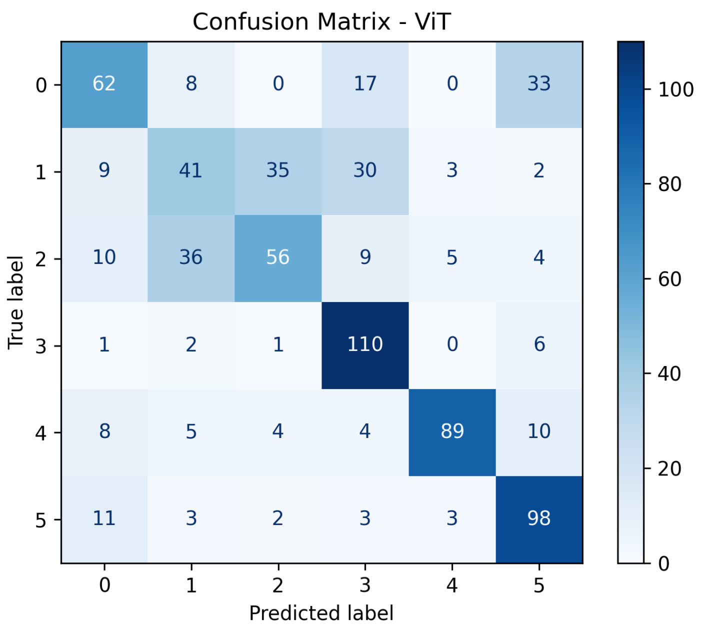
(a) Model loss



(b) Model accuracy

Figure 4.20: Model loss and accuracy plots for ViT model, for *NANI* dataset.

The confusion matrix is shown in Figure 4.21. It shows that a total of 264 images were misclassified by the ViT model on the validation set.



	0	1	2	3	4	5
0	62	8	0	17	0	33
1	9	41	35	30	3	2
2	10	36	56	9	5	4
3	1	2	1	110	0	6
4	8	5	4	4	89	10
5	11	3	2	3	3	98

Figure 4.21: Confusion matrix for Vision Transformer model, on the validation set of the *NANI* dataset.

4.2.6 Comparison of model performance

All five deep learning models were also trained and evaluated on the *NANI* dataset. The validation accuracies of the models are visualized in Figure 4.22. Also here, the VGG-16 and WideResNet-50-2 are the two best performing models.

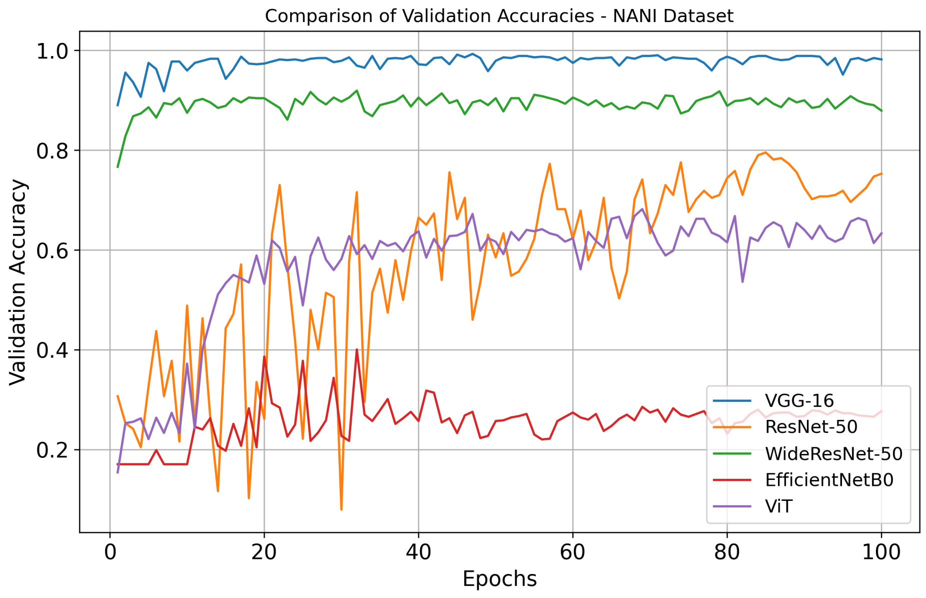


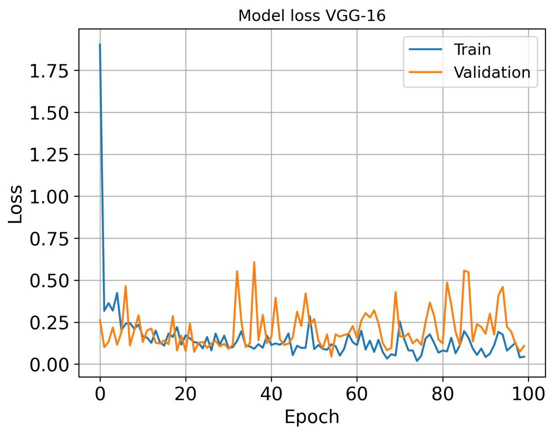
Figure 4.22: Validation accuracy of the deep learning models on *NANI* dataset.

4.3 NASI dataset

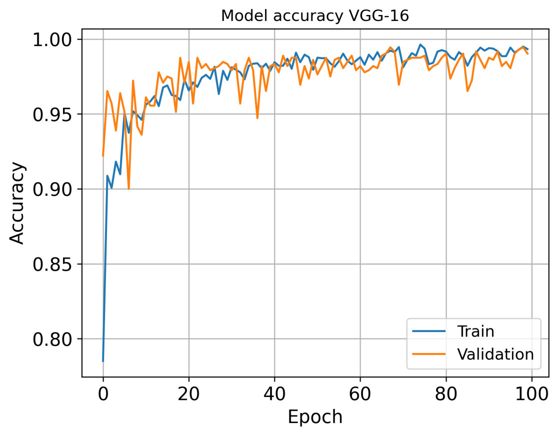
This section will describe the result of the VGG-16 and the Vision Transformer model, using the dataset with GANs generated images. These two models were selected as they represent the best-performing CNN model from two previous datasets and the newer, state-of-the-art Vision Transformer model. As in previous experiments, the model´s accuracy and loss will be plotted, and the predictions on the validation set will be visualized in the form of a confusion matrix. Finally, the validation accuracies of both models will be visualized in a single comparative plot.

4.3.1 VGG-16

The VGG-16 model achieved a best validation accuracy of 99.44% on 99th epoch. For this epoch, the training accuracy was 99.48%. This shows that this model achieved similar accuracies across all datasets. The model´s accuracy and loss plot over the 100 epochs of training is shown in Figure 4.23a and 4.23b below.



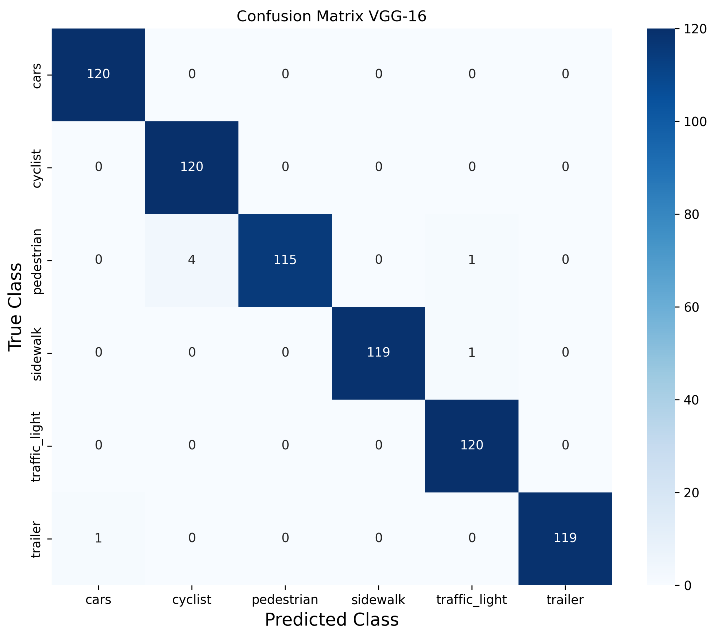
(a) Model loss



(b) Model accuracy

Figure 4.23: Model loss and accuracy plots for ViT model, for *NASI* dataset.

As one can interpret from the confusion matrix in Figure 4.24 below, most of the image samples in the validation set were classified correctly by the model. Only 7 image samples were misclassified.

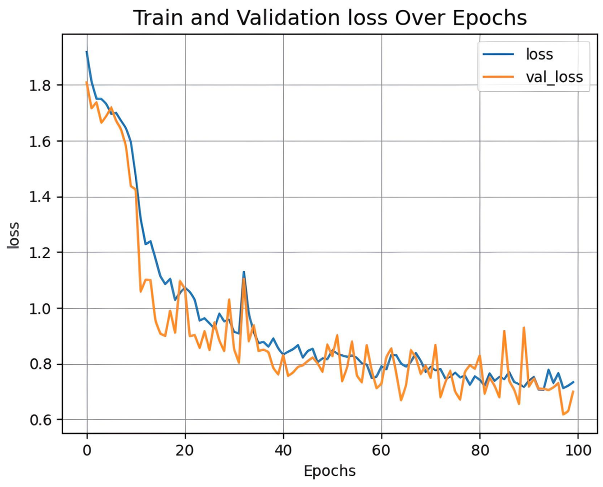


	cars	cyclist	pedestrian	sidewalk	traffic_light	trailer
cars	120	0	0	0	0	0
cyclist	0	120	0	0	0	0
pedestrian	0	4	115	0	1	0
sidewalk	0	0	0	119	1	0
traffic_light	0	0	0	0	120	0
trailer	1	0	0	0	0	119

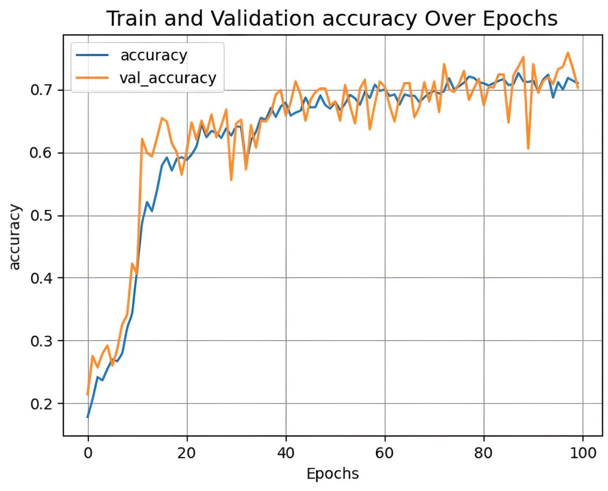
Figure 4.24: Confusion matrix for VGG-16 model, on the validation set of the *NASI* dataset.

4.3.2 Vision Transformer

Finally, the pre-trained Vision Transformer model was further trained on the dataset *NASI* dataset. After 100 epochs of training, it achieved a best validation accuracy of 75.83% on the 98th epoch. The corresponding training accuracy was 71.82%. Therefore, the ViT model in this study performs the best on the *NASI* dataset. The resulting model loss and model accuracy plots after the training process are shown in the Figure 4.25b and 4.25b below.



(a) Model loss



(b) Model accuracy

Figure 4.25: Model loss and accuracy plots for ViT model, for *NASI* dataset.

The final confusion matrix for the ViT model on the dataset with GANs images is shown in Figure 4.26. One can quickly see that the VIT model makes worse predictions than the VGG-16 model above for this dataset. Overall, the model misclassified 260 images on the validation set.

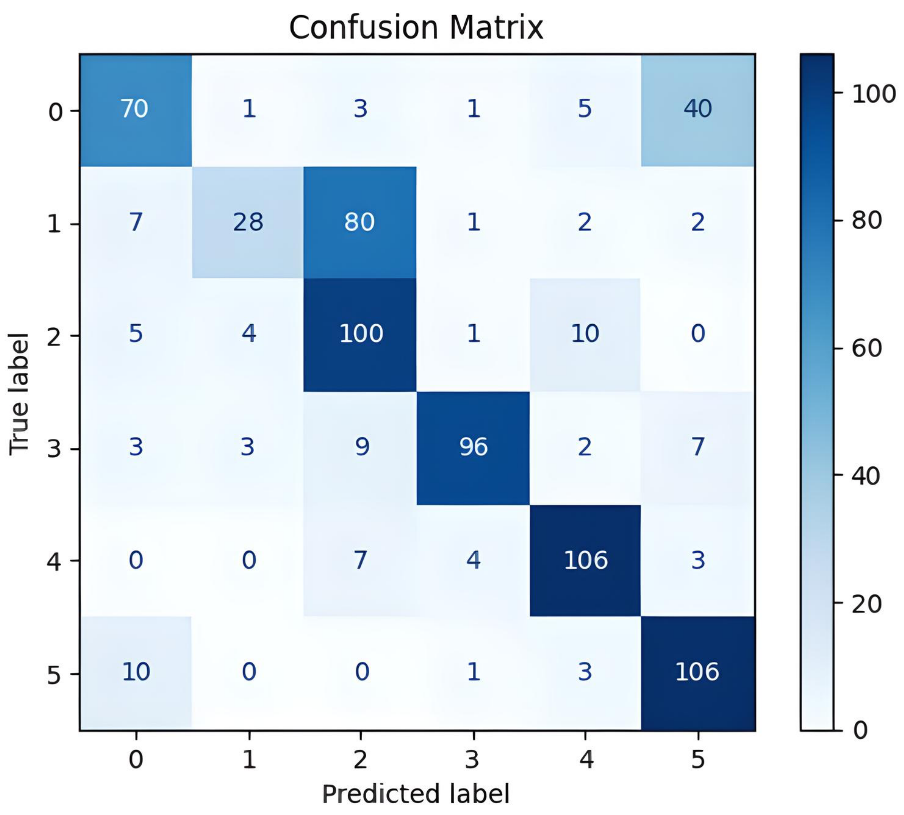


Figure 4.26: Confusion matrix for Vision Transformer model, on the validation set of the *NASI* dataset.

4.3.3 Comparison of model performance

As noted earlier, only the VGG-16 and Vision Transformer model were trained and evaluated on the *NASI* dataset with the synthetic images from DCGAN. As shown in Figure 4.27 below. The CNN-based VGG-16 model is achieving better results than the newer ViT model on this dataset as well.

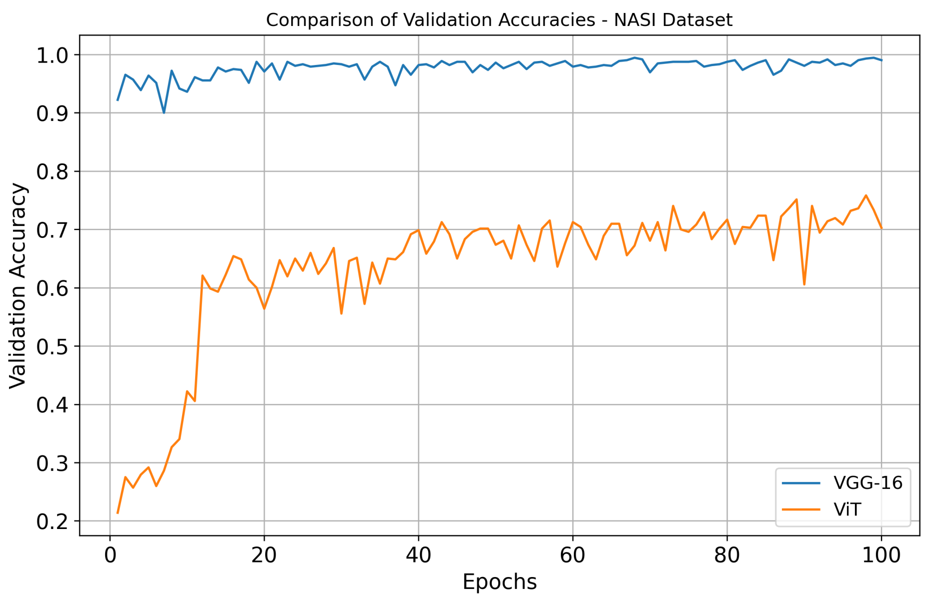


Figure 4.27: Validation accuracy of the deep learning models on *NASI* dataset.

4.4 DCGANs performance

Fake images were generated using a DCGAN trained from scratch, the results of which are visualized in this section. The structure and details of the DCGAN were outlined in the previous chapter. Images from each class were trained separately on the DCGAN, which then generated images for that specific class. This section will describe and visualize the results of this training process.

The generator and discriminator model loss plot, and inception score plot are only visualized for the *car* class in this section. This is because both the model loss plots and the inception score plots for all classes exhibited very similar performance and behaviour. These plots for the other classes in this thesis are shown in Appendix A.2. Nevertheless, this section includes visualizations of the synthetic images generated from all classes.

Through the loss plot for the discriminator and generator, which is shown in Figure 4.28, it is observed that the discriminator´s loss decreases over time, while the generator´s loss increases. Both losses, however, experience frequent jumps. This behaviour is consistent across the plots for the other classes, and will be further discussed in the next chapter.

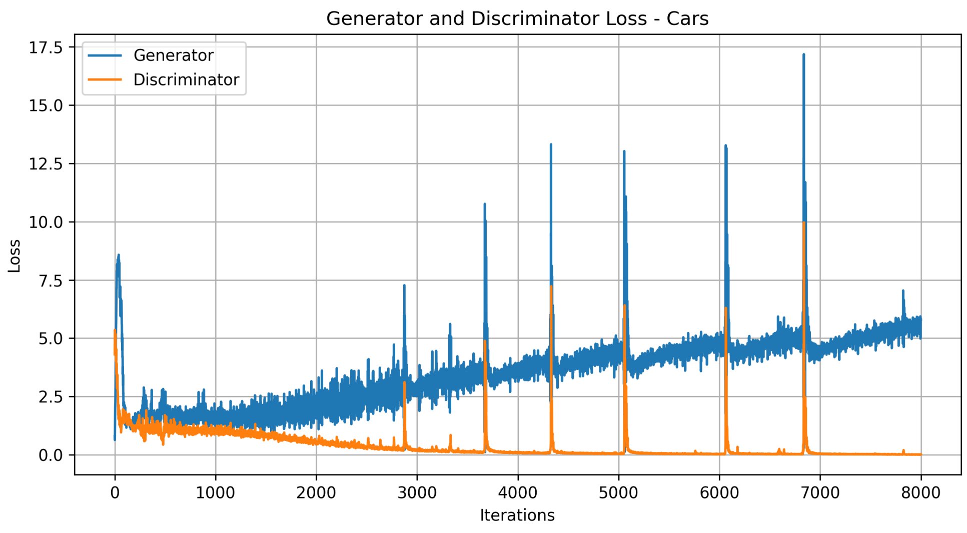


Figure 4.28: Model loss plot for generator (blue) and discriminator (orange), for *car* class.

The inception score plot for the *car* class is shown in Figure 4.29. The details of what inception score is, and its purpose were given earlier in Section 2.1.7. As can be seen from the plot below, the score appears to converge at around 1.5. This pattern is also observed for other classes, as shown in Appendix A.2.2.

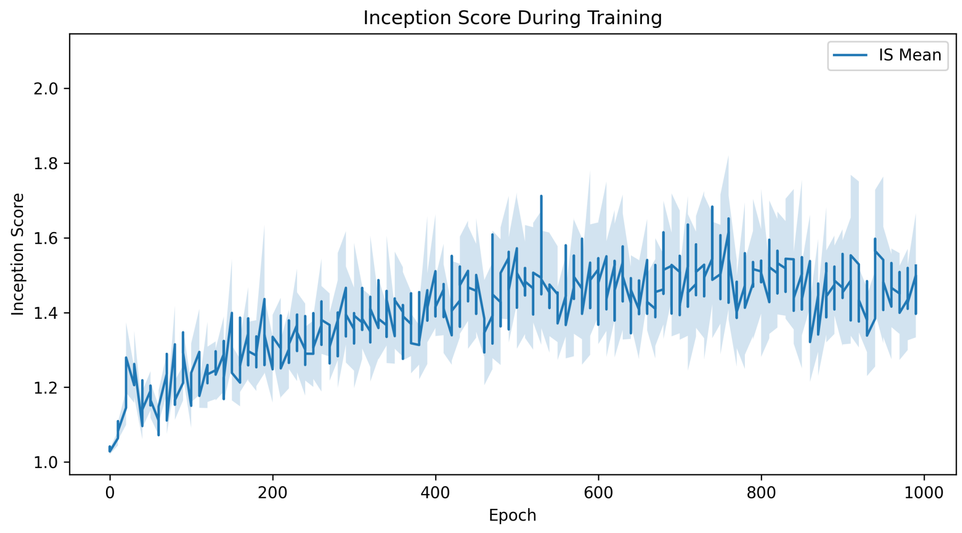


Figure 4.29: Inception score plot for *car* class. The inception score seems to converge around a value of 1.5.

Finally, the synthetic images that were generated after 1000 epoch for each class are shown below. As noted earlier, 64 synthetic images were generated for each class, but only 15 of these were chosen to be added to the third dataset. These visualizations of the generated images, together with the model loss plot and inception score-plot will be used in the discussion section to evaluate and discuss the results of the GANs.

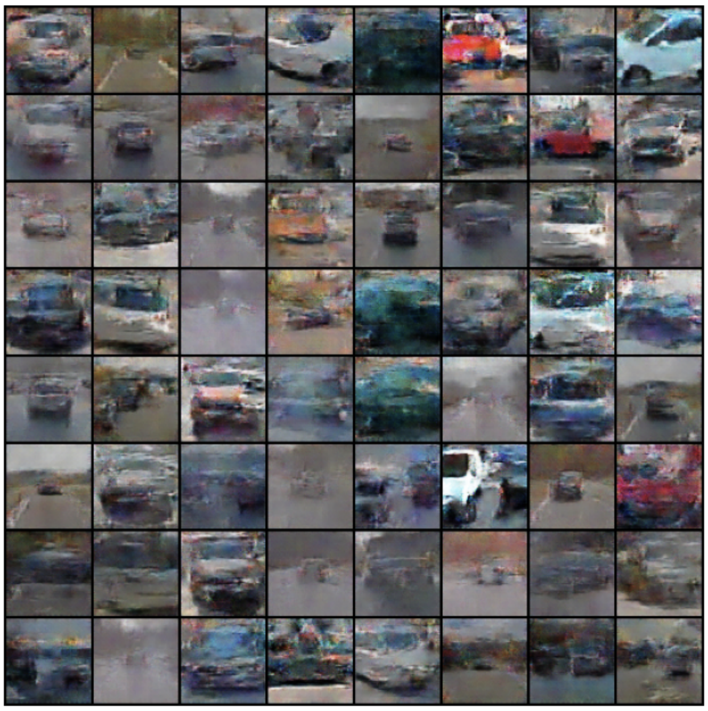


Figure 4.30: Synthetic images - *car*.

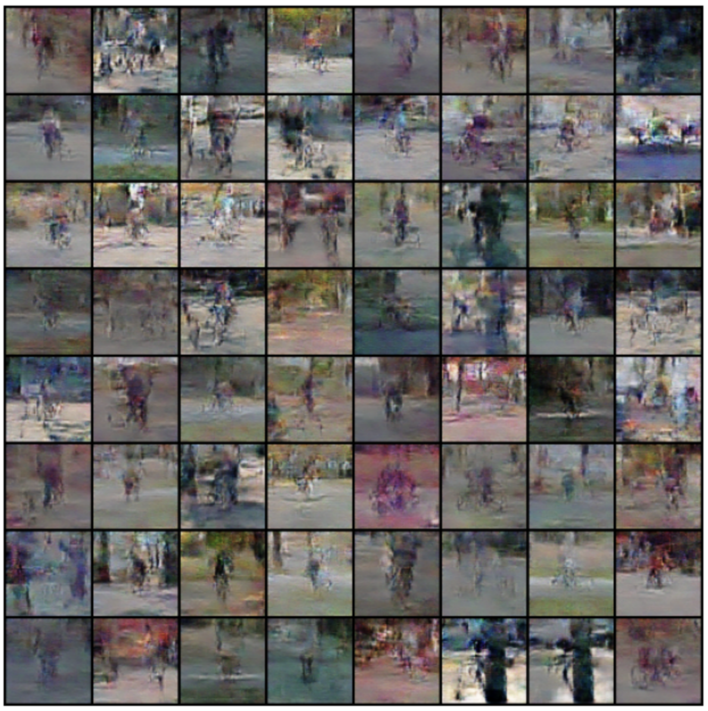


Figure 4.31: Synthetic images - *cyclist*.

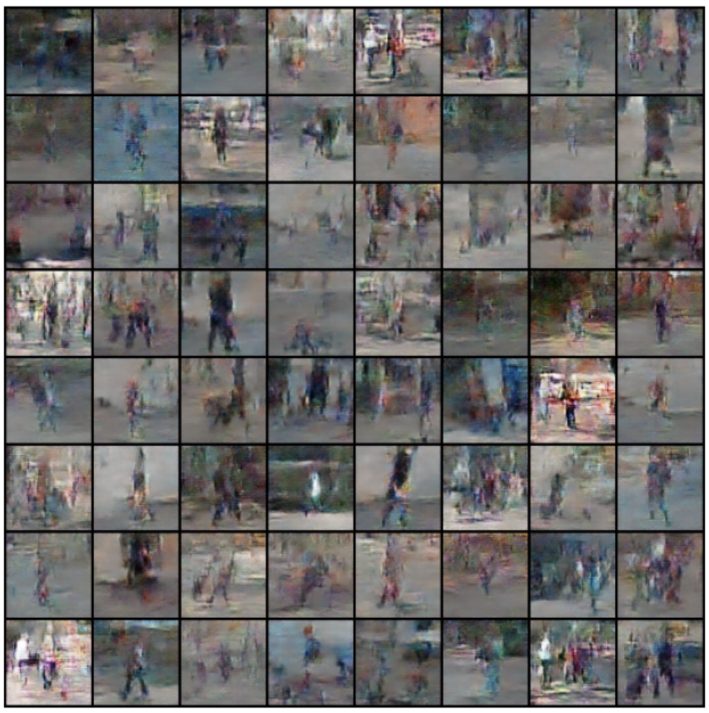


Figure 4.32: Synthetic images - *pedestrian*.

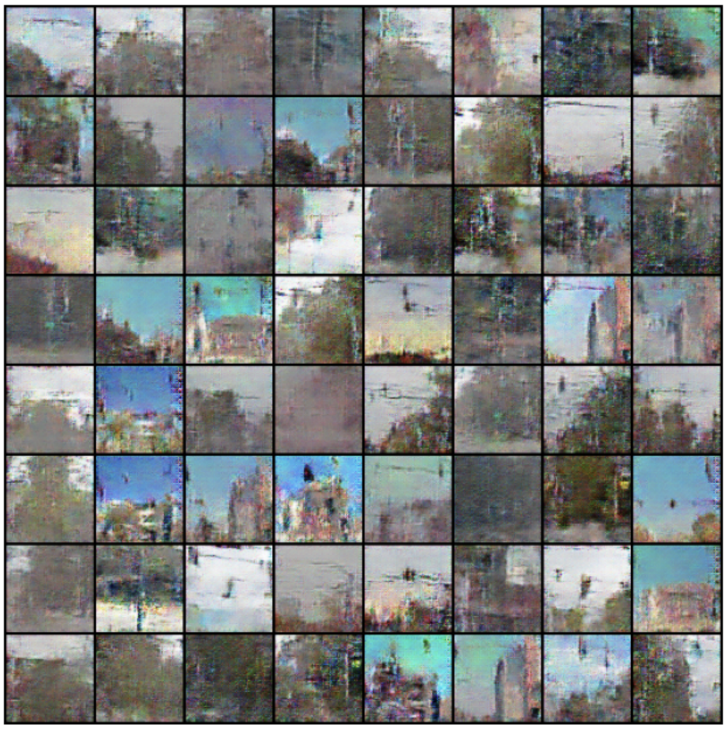


Figure 4.33: Synthetic images - *traffic light*.

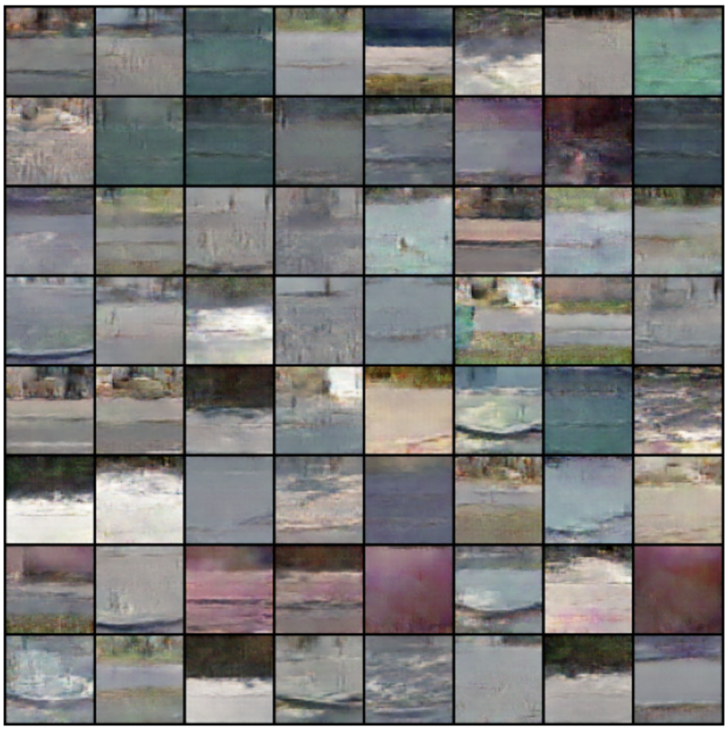


Figure 4.34: Synthetic images - *sidewalk*.

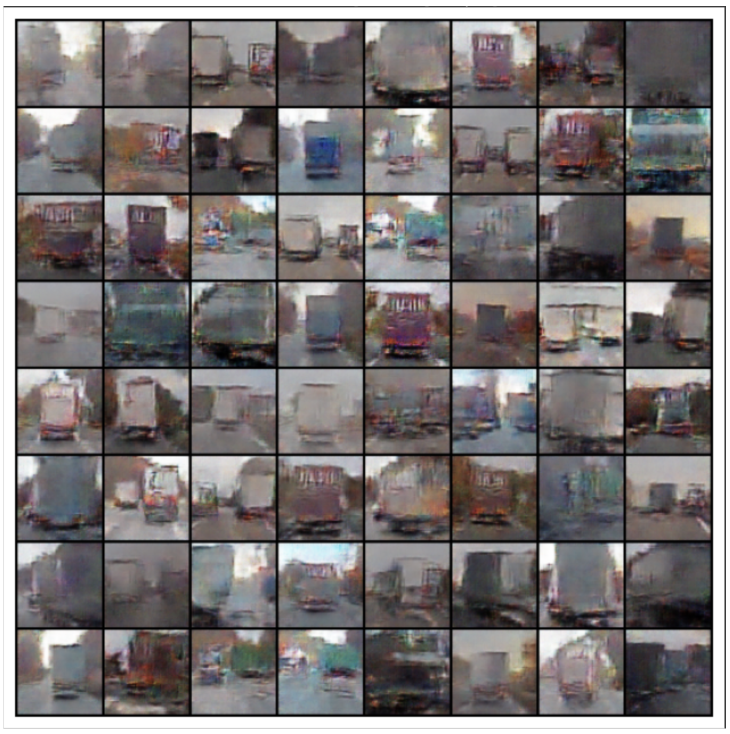


Figure 4.35: Synthetic images - *trailer*.

Chapter 5

Discussion

The primary purpose of this chapter is to discuss the findings from the previous chapter, keeping the research questions outlined in Chapter 1 in mind. It will include a discussion of the datasets used and provide some possible explanations for the results observed.

5.1 Dataset

Before delving into the specific results of the experiments in this study, it is crucial to consider several factors related to the three datasets used that might have influenced the outcomes.

5.1.1 Bias

The manual collection, annotation and cropping of the images makes the datasets vulnerable to human bias, given by the fact that all tasks were carried out by a single individual. This person´s subjective definitions of each class could significantly affect the dataset´s integrity, despite the class definitions being mentioned earlier in Section 3.3.2. For instance, if the majority of images in the *car* class mostly feature a limited selection of brands and models, the dataset might not accurately represent the diverse range of cars typically found on the road. This lack of representation can potentially affect the ability of deep learning models to generalize from this data.

Furthermore, the influence of human bias also affects the *NANI* dataset, which contains the noisy mixed-class images. Since these were also cropped manually, the subjective decisions made during this process could influence the model performances. Moreover, the *NASI* dataset is also affected by the ”weaknesses” of the first dataset, as the third dataset is based on the first dataset´s normal and augmented images, together with the synthetic images.

An alternative method could have reduced some of these biases. As mentioned in Section 3.3.1, the original A2D2 dataset comprised of several subsets of data, one of which was a *3D bounding box* dataset. Instead of the semantic segmentation dataset that was actually used, this bounding box subset could have been utilized. The use of the Bounding box dataset could have allowed for the automatic cropping of images using bounding box coordinates, which are included in this dataset together with the actual images [21]. This approach minimizes human intervention, thereby reducing the potential for bias, specifically human bias. Such a technique has been successfully implemented in other areas, as demonstrated by the following Keras example with the Caltech Birds Dataset [53]. This technique not only standardizes the data preparation process, but also enhances the replicability of the results. The technical details of such an approach have not been explored in this thesis, and this is merely a suggestion for an alternative method for cropping.

5.1.2 Generalization

The images in the A2D2 dataset were captured in various cities in Germany. Consequently, the classes represented in this dataset are typical of German settings. This geographic specificity suggests that while the deep learning models developed in this study effectively classify traffic-related objects in Germany, their performance may not generalize well to datasets from different domains. This potential limitation is crucial when evaluating the robustness of these models across diverse datasets. Although some models demonstrated robustness within the scope of this thesis, their performance might significantly differ when applied to datasets from other regions.

5.1.3 Class imbalance

Furthermore, as detailed in Section 3.3.2, the initial image collection for the six classes displayed significant class imbalance, with a difference of 200 images between the most and least represented classes. To address this issue, data augmentation techniques were applied to balance the distribution, ensuring that each class contained 600 images. While this strategy effectively solved the class imbalance within the datasets, it may have introduced redundancy, especially for smaller classes like *cyclist*. This was described earlier in Section 2.1.6.

5.1.4 DCGAN performance

The synthetic images generated from the DCGANs will be considered. The previous chapter visualized the model loss plot for the discriminator and the generator, together with the inception score plot for the *car* class. It can be seen from the loss plot that the generator and discriminator have loss values that are close in the early epochs. However, after around 2000 iterations, the discriminator loss decreases, and proceeds to reach values close to zero. On the other hand, the generator loss increases from this point and until the end of the training process. As mentioned in the last chapter, the observations made on the model loss plot and inception score plot for the *car* class can be applied and are consistent to all the other classes.

As stated, the inception scores of a GANs can be used as a statistical method to evaluate the image quality and diversity of the generated images. The inception score plot generated during the training process show what appears to be a convergence at a value around 1.5. Furthermore, the plot reveals that the inception score fluctuates significantly, which might indicate substantial variability in the quality of images generated throughout the learning process. Section 2.1.7 introduced the IS score of some GAN models in a research paper, where a well-performing DCGAN model had an IS score of around 6.5, while the real images of a benchmark dataset were measured to have an IS score of 8.43. Given these comparisons, the IS score of around 1.5 for the DCGAN model used in this thesis suggests that it produces relatively low-quality synthetic images. This becomes apparent when visually inspecting the images, where one can quickly observe the significant amount of noise present in all the synthetically generated images. This noise becomes more visually noticeable with the upsampling operation that was performed using the bicubic spline interpolation described earlier.

By looking at the various ways to evaluate DCGANs, it is clear that the training process is unstable. There are many possible reasons for this, as training GANs can be challenging.

Due to the increasing loss of the generator and the decreasing loss of the discriminator, it appears that the generator is creating poor-quality fake data, while the discriminator easily distinguishes between real and fake images. This issue could stem from insufficient adjustment of the hyperparameters, particularly the learning rate. The learning rate of the DCGAN model was adjusted several times during the modelling phase, which notably improved the quality of the generated images. Further adjustments could potentially enhance performance even more. Additionally, it is important to note that a learning rate adjuster was implemented. Although this reduced the loss for the generator and discriminator, the generated images were of lower quality. Consequently, the learning rate adjuster was not included in the final code for the DCGAN.

Occasional spikes in the generator and discriminator losses is seen, which further underlines the unstable nature of this DCGAN model. According to a paper by Hines [54], the reason behind these spikes could be due to the discriminator being over-trained on a small dataset. Consequently, the discriminator gets too powerful, leading the generator to fail in producing high-quality synthetic images. This issue highlights the competitive nature of the generator and discriminator, where the under-performance of one component results in the over-performance of the other.

Additionally, many of the generated images in certain classes were very similar, indicating a mode collapse during the training process. For instance, in the *trailer* class, most of the generated images are similar. This similarity could be attributed to the lack of variety in the training images, caused by the image augmentations used on the DCGAN's train set. The consequences of excessive use of image augmentations were described earlier in 2.1.6. A similar consequence of these augmentations can be seen on the DCGAN generated images on the *traffic light* class. Some of these images have features similar to the augmented images in its train set. This was described earlier in Section 2.1.6, and was referred to as augmentation ”leakage”.

The points mentioned above are important to consider when evaluating the performance of each deep learning model. They represent potential uncertainty factors regarding the dataset used in this thesis and the results of the various deep learning models.

5.2 Interpretation of results from deep learning models

This section will look into the results of all the deep learning models across the datasets they were used on, and give possible reasons for their performances.

5.2.1 VGG-16

The VGG-16 model achieved great results across all three datasets it was tested on, and was the best-performing model in this study. It performed well on all three datasets, and the validation accuracy remained consistently high across all datasets which is a sign of robustness. Moreover, the train and validation accuracy for the model remained mostly close, which indicates an absence of both overfitting and underfitting. Also, by looking at the confusion matrices of this model across all datasets, it is able to successfully classify most image samples.

5.2.2 ResNet-50

The ResNet-50 model was trained on the *NAI* and *NANI* dataset in this study. It achieved good validation accuracies on both datasets. However, it consistently showed higher training accuracies compared to validation accuracies throughout all epochs, and these training accuracies remained notably more stable. As explained, this is a clear sign of overfitting. However, it can be seen in the latter epochs that the validation accuracies approaches the train accuracies.

It is important to mention the high loss of the ResNet-50 model across the *NAI* and *NANI* dataset, compared to the other models in this study. The ResNet-50 model achieved much higher losses than the other models despite it having better accuracies than both the EfficientNetB0 and the ViT on these two datasets. A possible reason for this higher loss, could be explained with how the Categorical CrossEntropy loss is computed. As noted in Section 2.1.2, the Categorical Crossentropy loss, which is used for this ResNet-50 model and all the other DL models, computes the loss by comparing the probability distribution of the predicted output label and the true distribution of the labels. Therefore, an explanation of this abnormal loss of the ResNet-50 model could be down to the model predicting wrong classes with very high confidence.

5.2.3 WideResNet-50-2

The WideResNet-50-2 model achieved much higher performance than the model it is based on, ResNet-50. This very much confirms the statements of the WideResNet-50-2 model paper. From the results of this model across the two datasets it was trained on, it showed great results in both the accuracy and loss. As briefly mentioned in the *Result* chapter, the WideResNet-50-2 model converged very quickly across both datasets, and reached train and validation accuracies around 90% . This is similar to how the VGG-16 model behaved, but where the VGG-16 model´s accuracy increased gradually throughout the training process, the accuracy and loss of the WideResNet-50-2 model remained almost constant throughout the whole training process. A possible explanation for this could be that the initial learning rate of the model could have been set too high, which makes the model quickly converge to a local minimum, and not a global minimum. This can lead to the model not being able to escape and is stuck here for the rest of the training process.

Also, this model is experiencing a higher validation accuracy compared to the training accuracy. This is more notable for the *NANI* dataset, which contains the noisy mixed-class images in the train set. This is an unusual behaviour and might indicate a problem with how the images in the datasets are distributed. As described, augmentation was used to increase the number of images in the datasets. However, when splitting the dataset into train and test, the ratio between the normal and augmented images were not considered. In other words, the proportion of normal and augmented images is not constant between the train and test set.

5.2.4 EfficientNetB0

Already on the first dataset, this models has difficulties to learn. It does also show signs of overfitting on both of the datasets it was trained on, but this is especially evident when looking at its model loss and model accuracy graph for the *NANI* dataset. Moreover, by looking at the confusion matrix, one could see that the model classifies almost all classes as simply the *sidewalk* class. This low performance does not match with the performance of EfficientNetB0 on the ImageNet dataset, where it exceeded the performance of ResNet-50.

There could be many explanations to this poor performance. One possible reason could be due to the initial distribution of classes in the initially collected images. As seen in Section 3.3.2, the collected images ended up having an unbalanced distribution, and the *sidewalk* class had the most images. This unbalanced distribution was tackled by image augmentation, which increased the size of all classes to 600, for the first dataset. However, since the EfficientNetB0 model classifies most image samples as simply the *sidewalk* class, it might be that the image augmentations increase the samples in each class of the dataset, without adding meaningful diversity. This prohibits the model of learning a diverse set of features from the data, and was described earlier in Section 2.1.6.

Lastly, it is important to note that other deep learning models discussed in this thesis achieved significantly better results than the EfficientNetB0 model on the same datasets. This might indicate that the issue with this model's poor performance is not solely based on the dataset, but rather on the model's configurations. During the modelling process for the EfficientNetB0 model, several hyperparameters were manually tuned, including the learning rate, batch-size, choice of optimizer and regularization. These adjustments resulted in a slight increase in model accuracy, suggesting that further tuning could potentially yield even better results. As a final note, the notably poor performance of this model most likely stems from a combination of the points made above.

5.2.5 Vision Transformer

The ViT model performed good, but not state-of-the-art across the three datasets it was trained on. The model accuracy plot of this model signifies that it achieves a bit lower train and validation accuracies than the ResNet-50 model, but much lower loss. Furthermore, it is interesting to see that the ViT model yields the best validation accuracy on the last dataset with synthetic images, *NASI* dataset. This indicates that the addition of synthetic images in the train set improves the ViT model´s performance. This behaviour could be explained by the difference in how these two models see the images. As described in Section 2.1.4, ViT models differ from a traditional CNN models like VGG-16 in the sense that they have a global scene understanding, compared to CNNs which have a more local understanding of the scene. This difference might result in the ViT model to learn different features, and react differently to the synthetic images compared to the CNN-based VGG-16 model. Despite the adequate performance of the ViT model across the three datasets, its overall consistent performance across the datasets could be a sign of good robustness. It is also important to look at the performance of the ViT model in light of the data requirements needed for this model. As seen earlier in Section 2.1.4, the ViT model was only able to achieve state-of-the-art performance when pre-trained on a very large dataset such as the JFT-300M dataset, which has 300 million images. However, the ViT model in this thesis was pre-trained on the much smaller ImageNet dataset, which was shown to not achieve remarkable results compared to baseline CNN models.

5.2.6 GANs for self-driving cars

In the light of ViT's and VGG-16's strong performance on the *NASI* dataset, these results suggests that synthetic images created by GANs can be a viable option as an augmentation technique for tasks involving traffic participants and objects. The use of GANs can also be extended to computer-vision tasks in self-driving cars. As mentioned, computer vision tasks in self-driving cars, involving deep learning models, requires large datasets, where the training process can be expensive and time-consuming. Additionally, the process of acquiring the large datasets needed for these tasks can be expensive due to the many person-hours used for capturing the data, and inefficient due to numerous legal concerns [20]. The improvement in model accuracy for deep learning models was seen in the previously described paper *Reliability of GAN generated data to Train and Validate Perception Systems for Autonomous Vehicles* in Section 2.2. Likewise, the results of the experiments done in this study suggests that synthetic images using GANs could be used to improve the performance of deep learning models, and thus potentially reduce costs and increase efficiency in computer vision tasks related to self-driving cars. However, it is important to note that the implementation of GANs generated images were performed in a small scale for this study, and further research must be done to understand more about the impact of these synthetic images, in the context of this study and the models used in this study.

5.3 Limitations

The primary constraint encountered in this thesis was the size of the datasets. The selected deep learning models, and DCGAN, require large amounts of data to perform well and reliably, which is especially important for self-driving cars. Even though some models, such as VGG-16, achieved great results with the datasets used in this thesis, evaluating these models' performance on larger datasets, similar to those used for training deep learning models in self-driving cars, would be beneficial.

Furthermore, it is important to note that the results of this study are specific to the datasets utilized and may not necessarily predict behavior in real-world scenarios. This study serves primarily as an exploratory investigation of selected deep learning models on datasets related to traffic object and participants, and does not replicate any specific real-world task of a self-driving car.

5.4 Further work

Besides overcoming the previously described limitations, there are several directions one could pursue moving forward. An exciting direction could involve expanding the dataset to check the performance of DCGANs if trained on a larger scale. It would also be interesting to see how the performance of DCGANs changes with more hyperparameter tuning, given their sensitivity to these parameters.

Additionally, another approach could involve replicating the experiments in this thesis using more thoughtfully designed augmentations, especially for the images used to train the DCGAN. This was addressed in the paper *Training Generative Adversarial Networks with Limited Data*, where the researchers tackled the problem of DCGAN discriminator overfitting through more carefully considered augmentations [43].

It was observed from the experiments in this thesis that the ViT model achieved promising results on the three datasets, particularly in terms of robustness. An exciting future approach would be to further adjust the hyperparameters of the ViT model, such as the patch size, batch size, transformer units, and transformer layers.

Chapter 6

Conclusion

We cannot deny the crucial role of deep learning models in computer vision tasks. Further improvements and understanding of these models can significantly accelerate the development of self-driving cars. This study compares four CNN-based deep learning models (VGG-16, ResNet-50, WideResNet-50-2 and EfficientNetB0), and a state-of-the-art ViT model on three datasets relevant to self-driving cars, aiming to solve an image classification task. All models used in this study were pre-trained on the ImageNet dataset and trained further on the *NAI* (normal and augmented images) and the *NANI* (normal, augmented, and noisy mixed-class images). Only the VGG-16 and ViT models were trained on the *NASI*, which included normal, augmented, and synthetic images generated using a DCGAN model trained from scratch.

The experiments in this study demonstrates that the VGG-16 model performed the best across all three datasets. This model consistently achieved validation accuracies of 99% or higher for each dataset. The performance of the other models varied; some showed better results on the second dataset than on the first, while others declined in accuracy. Due to these mixed outcomes, it is challenging to make definitive conclusions about the impact of noisy images on the models across all datasets. However, the strong performance of the VGG-16 model indicates that deep learning models can effectively handle datasets with noisy mixed-class images. The Vision Transformer model used in this thesis showed good performance across all three dataset. Interestingly, the ViT model achieved its best validation accuracy on the third dataset with the synthetic images.

In conclusion, the VGG-16 model provided the best performance to identify traffic participants and objects on the three datasets relevant to self-driving cars. Moreover, the positive performance of both the VGG-16 and ViT model on the third dataset with synthetic images indicates that deep learning models are able to successfully classify DCGAN generated images.

Bibliography

[1] S. Singh, R. Kumar, S. Payra, and S. Kumar Singh, "Artificial intelligence and machine learning in pharmacological research: Bridging the gap between data and drug discovery," *Cureus*, vol. 15, 08 2023.

[2] J. Patterson and A. Gibson, *Deep learning: A practitioner's approach.* " O'Reilly Media, Inc.", 2017.

[3] M. Yani, S. Irawan, and C. Setianingsih, "Application of transfer learning using convolutional neural network method for early detection of terry's nail," *Journal of Physics: Conference Series*, vol. 1201, p. 012052, 05 2019.

[4] P. Khan, M. F. Kader, S. M. R. Islam, A. B. Rahman, M. Kamal, M. Toha, and K. Kwak, "Machine learning and deep learning approaches for brain disease diagnosis: Principles and recent advances," *IEEE Access*, vol. 9, pp. 37622 – 37655, 02 2021.

[5] B. Shi, R. Hou, M. Mazurowski, L. Grimm, Y. Ren, J. Marks, L. King, C. Maley, and J. Lo, "Learning better deep features for the prediction of occult invasive disease in ductal carcinoma in situ through transfer learning," p. 98, February 2018.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[7] PyTorch, "Wide resnet by sergey zagoruyko." `https://pytorch.org/hub/pytorch_vision_wide_resnet/`, 2022. Accessed: 12-05-2024.

[8] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[9] AUDI AG, "Sensor setup - a2d2." `https://www.a2d2.audi/a2d2/en/sensor-setup.html`, 2023. Accessed: 12-05-2024.

[10] Statistics Norway (Statistisk sentralbyrå), "Innenlandsk transport." `https://www.ssb.no/transport-og-reiseliv/landtransport/statistikk/innenlandsk-transport`, 2023. Accessed: 13-03-2024.

[11] Statistics Norway (Statistisk sentralbyrå), "Trafikkulykker med personskade." `https://www.ssb.no/transport-og-reiseliv/landtransport/statistikk/trafikkulykker-med-personskade`, 2024. Accessed: 13-03-2024.

[12] Norwegian Government, "Meld. st. 40 (2015–2016) - trafikksikkerhetsarbeidet – samordning og organisering." `https://www.regjeringen.no/no/dokumenter/meld.-st.-40-20152016/id2513038/?ch=3`, 2016. Accessed: 13-03-2024.

[13] Statens Vegvesen, "Dybdeanalyser av døds- ulykker i vegtrafikken 2022," Rapport 936, Statens Vegvesen, 2023. Accessed: 01-03-2024.

[14] Norwegian Public Roads Administration (Statens vegvesen), "Best på trafikksikkerhet – 179 tiltak skal gjøre oss enda bedre." `https://www.vegvesen.no/om-oss/presse/aktuelt/2022/03/trafikksikkerhetsplanen/`, March 2022. Accessed: 13-03-2024.

[15] S. M. Hegner, A. D. Beldad, and G. J. Brunswick, "In automatic we trust: investigating the impact of trust, control, personality characteristics, and extrinsic and intrinsic motivations on the acceptance of autonomous vehicles," *International Journal of Human–Computer Interaction*, vol. 35, no. 19, pp. 1769–1780, 2019.

[16] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. M. Paixao, F. Mutz, *et al.*, "Self-driving cars: A survey," *Expert systems with applications*, vol. 165, p. 113816, 2021.

[17] B. B. Elallid, N. Benamar, A. S. Hafid, T. Rachidi, and N. Mrani, “A comprehensive survey on the application of deep and reinforcement learning approaches in autonomous driving,” *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 9, pp. 7366–7390, 2022.

[18] J. Ni, Y. Chen, Y. Chen, J. Zhu, D. Ali, and W. Cao, “A survey on theories and applications for self-driving cars based on deep learning methods,” *Applied Sciences*, vol. 10, no. 8, p. 2749, 2020.

[19] A. Gupta, A. Anpalagan, L. Guan, and A. S. Khwaja, “Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues,” *Array*, vol. 10, p. 100057, 2021.

[20] M. Ivanovs, K. Ozols, A. Dobrajs, and R. Kadikis, “Improving semantic segmentation of urban scenes for self-driving cars with synthetic images,” *Sensors*, vol. 22, no. 6, p. 2252, 2022.

[21] J. Geyer, Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A. S. Chung, L. Hauswald, V. H. Pham, M. Mühlegg, S. Dorn, *et al.*, “A2d2: Audi autonomous driving dataset,” *arXiv preprint arXiv:2004.06320*, 2020.

[22] S. Raschka and V. Mirjalili, *Python Machine Learning, 3rd Ed.* Birmingham, UK: Packt Publishing, 2019.

[23] Y. Baştanlar and M. Özuysal, “Introduction to machine learning,” *miRNomics: MicroRNA biology and computational analysis*, pp. 105–128, 2014.

[24] E. F. Morales and H. J. Escalante, “A brief introduction to supervised, unsupervised, and reinforcement learning,” in *Biosignal processing and classification using computational learning and intelligence*, pp. 111–129, Elsevier, 2022.

[25] W. Heyden, H. Ullah, M. S. Siddiqui, and F. Al Machot, “An integral projection-based semantic autoencoder for zero-shot learning,” *IEEE Access*, 2023.

[26] P. Cunningham, M. Cord, and S. J. Delany, “Supervised learning,” in *Machine learning techniques for multimedia: case studies on organization and retrieval*, pp. 21–49, Springer, 2008.

[27] F. Chollet, *Deep learning with Python.* Manning Publications Co., 2017.

[28] E. Stevens, L. Antiga, and T. Viehmann, *Deep learning with PyTorch.* Manning Publications, 2020.

[29] B. Ghojogh and M. Crowley, “The theory behind overfitting, cross validation, regularization, bagging, and boosting: tutorial,” *arXiv preprint arXiv:1905.12787*, 2019.

[30] J. Gu and D. Oelke, “Understanding bias in machine learning,” *arXiv preprint arXiv:1909.01866*, 2019.

[31] A. Krenker, J. Bešter, and A. Kos, “Introduction to the artificial neural networks,” *Artificial Neural Networks: Methodological Advances and Biomedical Applications. InTech*, pp. 1–18, 2011.

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

[33] R. Adrian, *Deep learning for computer vision with python.* PyImageSearch.com, 2017.

[34] S. H. Hasanpour, M. Rouhani, M. Fayyaz, and M. Sabokrou, “Lets keep it simple, using simple architectures to outperform deeper and more complex architectures,” *arXiv preprint arXiv:1608.06037*, 2016.

[35] S. Zagoruyko and N. Komodakis, “Wide residual networks,” *arXiv preprint arXiv:1605.07146*, 2016.

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

[37] G. Boesch, “Vision Transformers (ViT) in Image Recognition – 2024 Guide.” `https://viso.ai/deep-learning/vision-transformer-vit/`. Accessed: 04-04-2024.

[38] M. Raghu, T. Unterthiner, S. Kornblith, C. Zhang, and A. Dosovitskiy, “Do vision transformers see like convolutional neural networks?,” *Advances in neural information processing systems*, vol. 34, pp. 12116–12128, 2021.

[39] C. Li and C. Zhang, “Toward a deeper understanding: Retnet viewed through convolution,” *Available at SSRN 4637493*, 2023.

[40] C. Shorten and T. M. Khoshgoftaar, “A survey on image data augmentation for deep learning,” *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.

[41] K. Alomar, H. I. Aysel, and X. Cai, “Data augmentation in classification and segmentation: A survey and new strategies,” *Journal of Imaging*, vol. 9, no. 2, p. 46, 2023.

[42] V. Bok and J. Langr, *GANs in action: deep learning with generative adversarial networks*. Simon and Schuster, 2019.

[43] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, “Training generative adversarial networks with limited data,” *Advances in neural information processing systems*, vol. 33, pp. 12104–12114, 2020.

[44] K. Shmelkov, C. Schmid, and K. Alahari, “How good is my gan?,” in *Proceedings of the European conference on computer vision (ECCV)*, pp. 213–229, 2018.

[45] W. Farag, “Recognition of traffic signs by convolutional neural nets for self-driving vehicles,” *International Journal of Knowledge-based and Intelligent Engineering Systems*, vol. 22, no. 3, pp. 205–214, 2018.

[46] W. Xu, N. Souly, and P. P. Brahma, “Reliability of gan generated data to train and validate perception systems for autonomous vehicles,” in *Proceedings of the ieee/cvf winter conference on applications of computer vision*, pp. 171–180, 2021.

[47] Q.-V. Lai-Dang, “A survey of vision transformers in autonomous driving: Current trends and future directions,” *arXiv preprint arXiv:2403.07542*, 2024.

[48] I. Hasan, S. Liao, J. Li, S. U. Akram, and L. Shao, “Pedestrian detection: Domain generalization, cnns, transformers and beyond,” *arXiv preprint arXiv:2201.03176*, 2022.

[49] R. Barman, S. Deshpande, S. Agarwal, U. Inamdar, M. Devare, and A. Patil, "Transfer learning for small dataset," in *Proceedings of the National Conference on Machine Learning*, vol. 26, ResearchGate Berlin, Germany, 2019.

[50] PyTorch, "Transfer learning for computer vision tutorial." `https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html`, 2023. Accessed: 03-04-2024.

[51] TensorFlow Team, "Transfer learning and fine-tuning." `https://www.tensorflow.org/tutorials/images/transfer_learning`, 2023. Accessed: 03-04-2024.

[52] M. Ekman, *Learning deep learning: Theory and practice of neural networks, computer vision, natural language processing, and transformers using TensorFlow*. Addison-Wesley Professional, 2021.

[53] Keras, "Adaptive discriminator augmentation (ada) for training generative adversarial networks." `https://keras.io/examples/generative/gan_ada/`, 2022. Accessed: 28-04-2024.

[54] S. Hines, "Generative adversarial networks take on hand drawn sketches: An application to louisiana culture and mardi gras fashion," 2022.

[55] Norwegian University of Life Sciences, "Artificial intelligence at the faculty of science and technology." `https://www.nmbu.no/en/faculties/faculty-science-and-technology/kunstig-intelligens-ved-realtek`, 2023. Accessed: 26-04-2024.

Appendix A

Additional Material

A.1 Use of artificial tool - ChatGPT

ChatGPT was used as an artificial intelligence tool for this study, assisting with the following tasks: debugging code, check for grammar mistakes and synonyms in sentences. More specifically, ChatGPT was used to resolve error messages from the scripts for the deep learning models and the DCGAN model. Additionally, it was used as a tool to check for grammatical mistakes in sentences, and recommending synonyms to words. The NMBU REALTEK Guidelines for the use of Artificial Intelligence [55] was read and understood. The usage of ChatGPT in this thesis are used in accordance with the given guidelines. Some examples of the prompts I have asked ChatGPT, that reflects my usage of it in this thesis includes:

- *Hi! I think there are some grammar mistakes in this sentence. Can you check if there are any grammar mistakes in the sentence, and correct them?*
- *Hi! I get this [”An error message”] error message when running this [...] piece of code. What could be the reason for this, and how can I solve it?*
- *I have used ”Furthermore” several times in my text. What is a good alternative phrase to use, instead of ”Furthermore”?*

A.2 DCGAN plots

This section of the appendix visualizes the model loss and inception score plots for the classes not covered earlier in Section 4.4. As mentioned, the model loss plots and inception score plots showed very similar behaviour across all classes.

A.2.1 Model loss plots

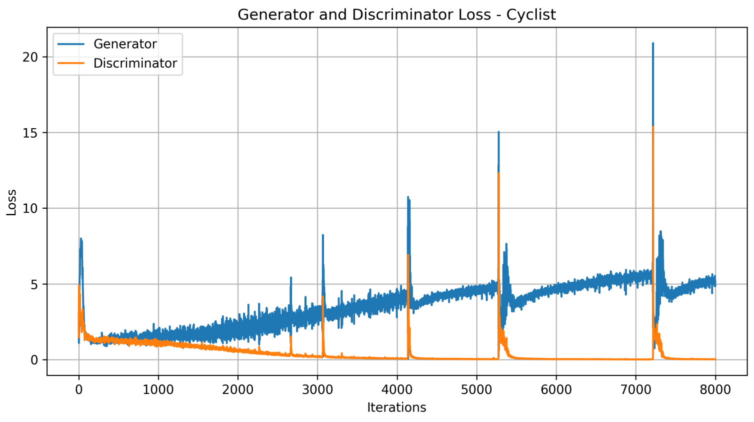


Figure A.1: Model loss plot for generator (blue) and discriminator (orange), for *cyclist* class.

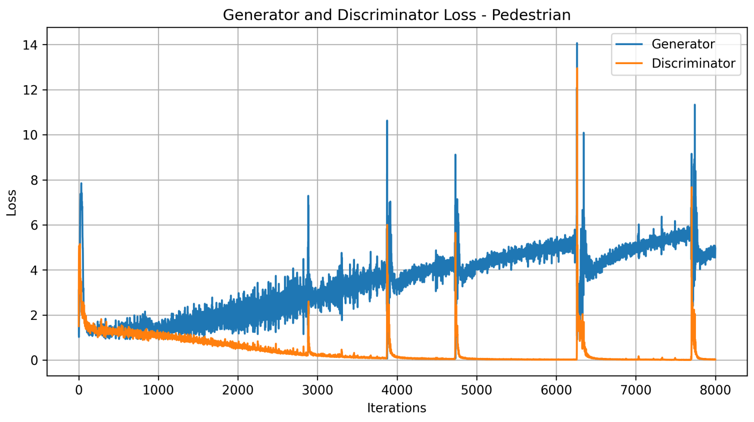


Figure A.2: Model loss plot for generator (blue) and discriminator (orange), for *pedestrian* class.

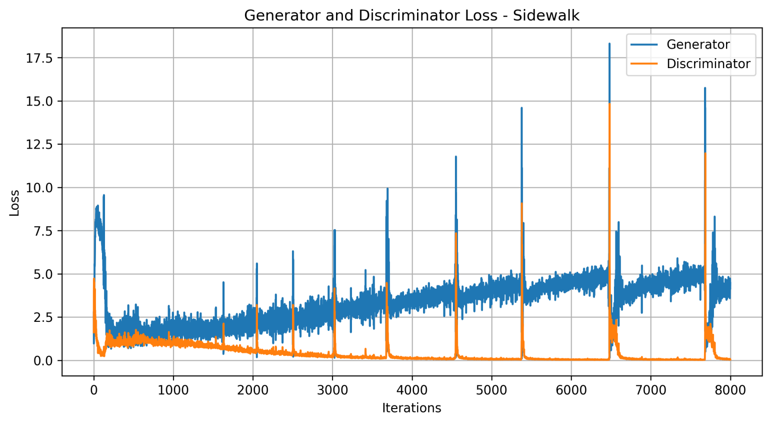


Figure A.3: Model loss plot for generator (blue) and discriminator (orange), for *sidewalk* class.

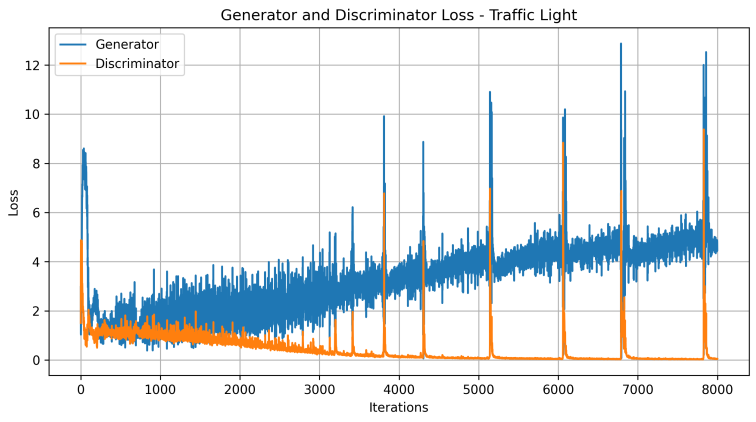


Figure A.4: Model loss plot for generator (blue) and discriminator (orange), for *traffic light* class.

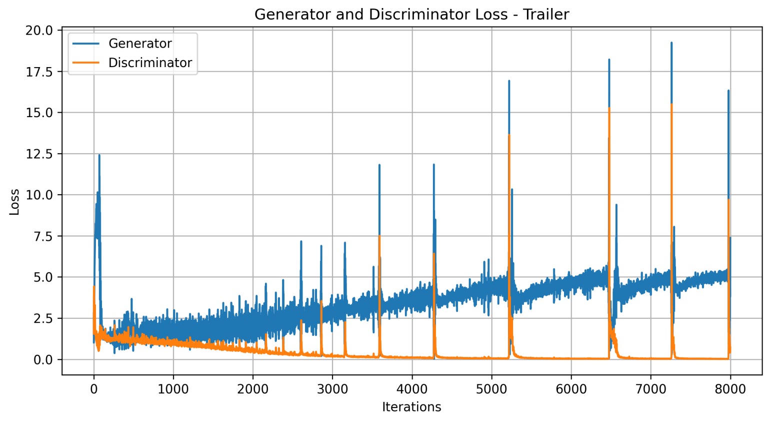


Figure A.5: Model loss plot for generator(blue) and discriminator(orange), for *trailer* class.

A.2.2 Inception score plots

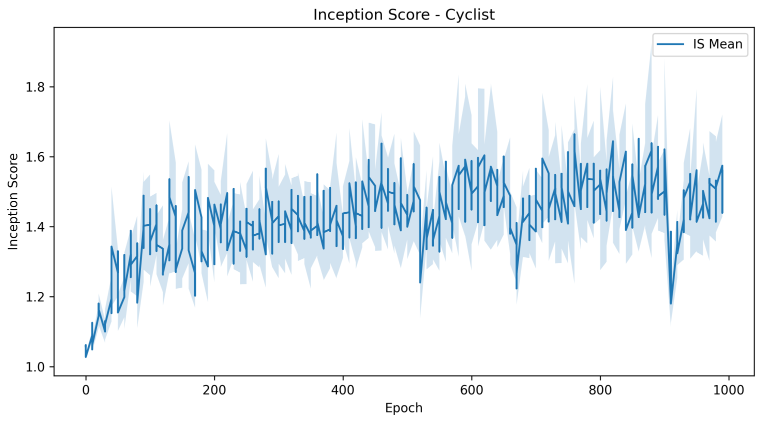


Figure A.6: Inception score plot for *cyclist* class.

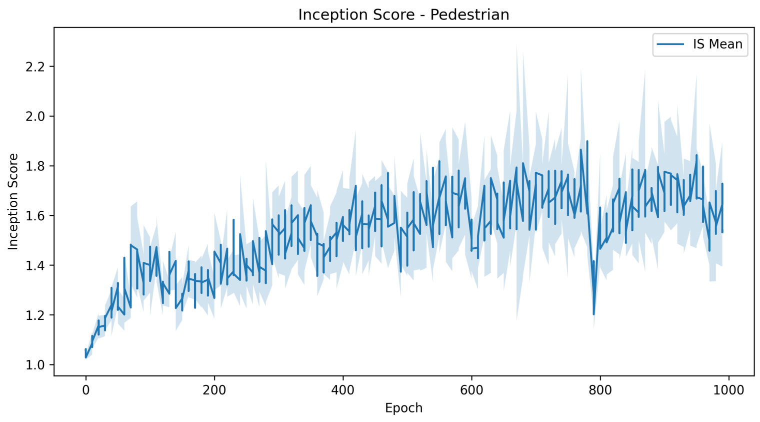


Figure A.7: Inception score plot for *pedestrian* class.

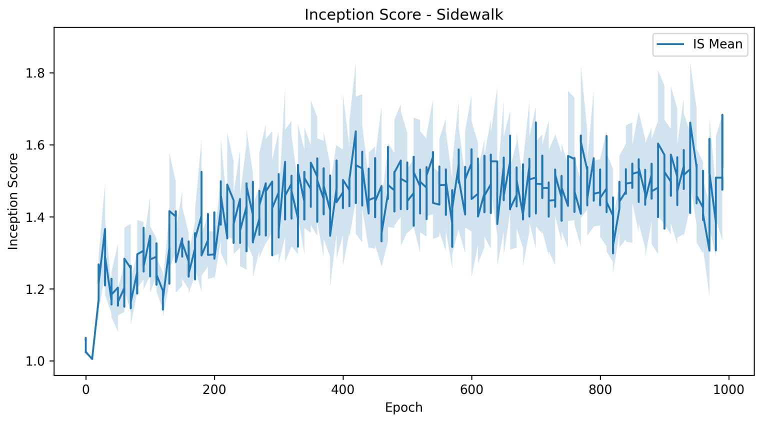


Figure A.8: Inception score for *sidewalk* class.

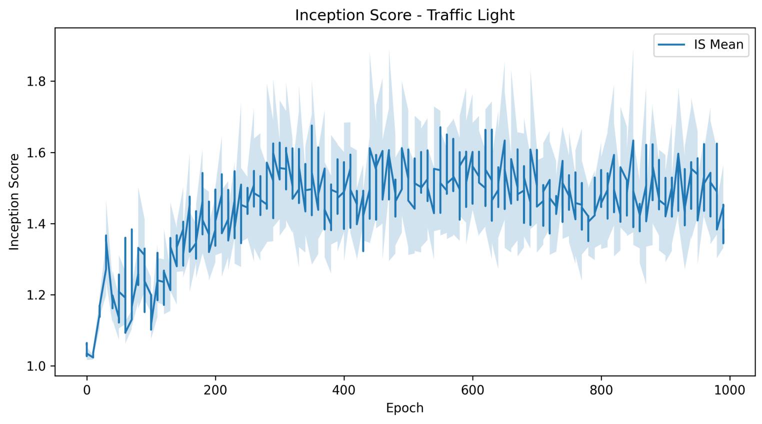


Figure A.9: Inception score plot for *traffic light* class.

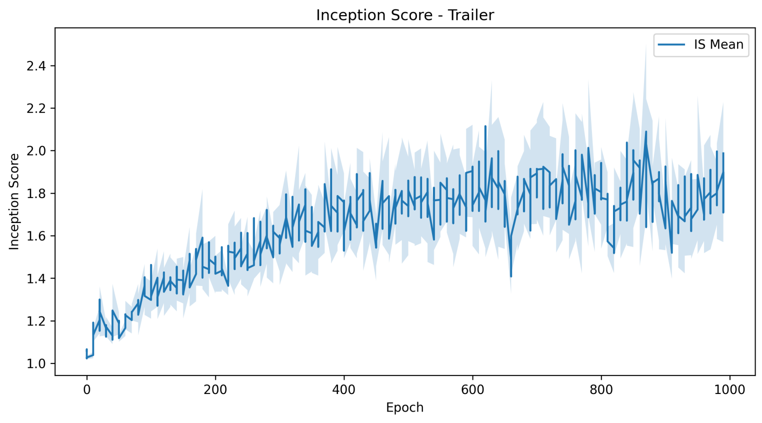


Figure A.10: Inception score plot for *trailer* class.